Guacho 3D

V1.1

Generated by Doxygen 1.8.9.1

Thu Apr 2 2015 12:50:14

Contents

Chapter 1

GUACHO-3D Documentation

Authors

Alejandro Esquivel et al.

1.1 Introduction

Documentation of the Guacho code

1.2 release.notes

1.3 requirements

Fortran 90/95 compiler with C preprocessor, Message Passing Interface (optional), gmake.

Chapter 2

Modules Index

2.1 Modules List

Here is a list of all documented modules with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Module Documentation

4.1 boundaries Module Reference

Boundary conditions.

Functions/Subroutines

- subroutine boundaryi (time, dt)

  Boundary conditions for 1st order half timestep.
- subroutine boundaryii (time, dt)

  Boundary conditions for 2nd order half timestep.

4.1.1 Detailed Description

Sets boundary conditions, the type of boundaries is set in the Makefile

4.1.2 Function/Subroutine Documentation

4.1.2.1 subroutine boundaries::boundaryi (real, intent(in), optional *time*, real, intent(in), optional *dt*)

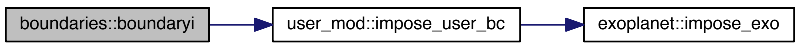
Boundary conditions for 1st order half timestep
The conditions only are imposed at the innermost ghost cell, on the u (unstepped) variables

Parameters

real	[in] optional, time : integration time
real	[in] optional, dt : timestep

Definition at line 50 of file boundaries.f90.

Here is the call graph for this function:



4.1.2.2 subroutine boundaries::boundaryii (real, intent(in), optional *time,* real, intent(in), optional *dt*)

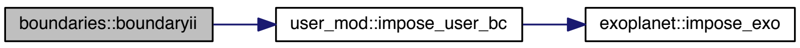
Boundary conditions for 2nd order half timestep
The conditions only are imposed in two ghost cells on the up (stepped) variables

Parameters

real	[in] optional, time : integration time
real	[in] optional, dt : timestep

Definition at line 264 of file boundaries.f90.

Here is the call graph for this function:



4.2 coldens_utilities Module Reference

Column densirt projection.

Functions/Subroutines

- subroutine init_coldens ()

  Initializes data.
- subroutine read_data (u, itprint, filepath)

  reads data from file
- subroutine getxyz (i, j, k, x, y, z)

  gets position of a cell
- subroutine rotation_x (theta, x, y, z, xn, yn, zn)

  Rotation around the X axis.
- subroutine rotation_y (theta, x, y, z, xn, yn, zn)

  Rotation around the Y axis.
- subroutine rotation_z (theta, x, y, z, xn, yn, zn)

  Rotation around the Z axis.
- subroutine fill_map (nxmap, nymap, u, map, dxT, dyT, theta_x, theta_y, theta_z)

  Fill target map.
- subroutine write_map (fileout, nxmap, nymap, map)

  Writes projection to file.

4.2.1 Detailed Description

Utilities to compute a column density map

4.2.2 Function/Subroutine Documentation

4.2.2.1 subroutine coldens_utilities::fill_map (integer, intent(in) *nxmap,* integer, intent(in) *nymap,* real, dimension(neq,nxmin:nxmax,nymin:nymax, nzmin:nzmax), intent(in) *u,* real, dimension(nxmap,nymap), intent(out) *map,* real, intent(in) *dxT,* real, intent(in) *dyT,* real, intent(in) *theta_x,* real, intent(in) *theta_y,* real, intent(in) *theta_z*)

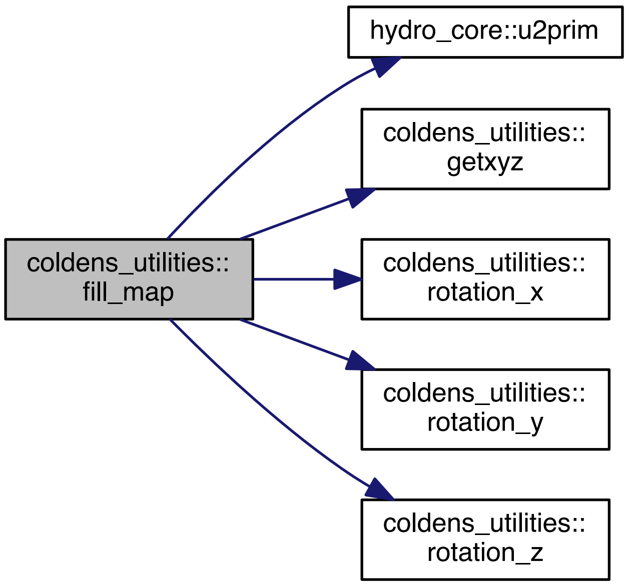
Fills the target map of one MPI block

Parameters

integer	[in] nxmap : Number of X cells in target
integer	[in] nymap : Number of Y cells in target
real	[in] u(neq,nxmin:nxmax,nymin:nymax, nzmin:nzmax) : conserved variables
real	[out] map(nxmap,mymap) : Target map
real	[in] dxT : target pixel width
real	[in] dyT : target pixel height
real	[in] thetax : Rotation around X
real	[in] thetay : Rotation around Y
real	[in] thetaz : Rotation around Z

Definition at line 286 of file coldens.f90.

Here is the call graph for this function:


hydro_core::u2prim
coldens_utilities:: getxyz
coldens_utilities:: fill_map
coldens_utilities:: rotation_x
coldens_utilities:: rotation_y
coldens_utilities:: rotation_z

4.2.2.2 subroutine coldens_utilities::getxyz (integer, intent(in) *i*, integer, intent(in) *j*, integer, intent(in) *k*, real, intent(out) *x*, real, intent(out) *y*, real, intent(out) *z*)

Returns the position and spherical radius calculated with respect to the center of the grid

Parameters

integer	[in] i : cell index in the x direction
integer	[in] j : cell index in the y direction
integer	[in] k : cell index in the z direction
real	[in] x : x position in the grid

real	[in] y : y position in the grid
real	[in] z : z position in the grid

Definition at line 188 of file coldens.f90.

4.2.2.3 subroutine coldens_utilities::init_coldens ( )

Initializes data, MPI and other stuff

Definition at line 36 of file coldens.f90.

4.2.2.4 subroutine coldens_utilities::read_data (real, dimension(neq,nxmin:nxmax,nymin:nymax,nzmin:nzmax), intent(out) *u,* integer, intent(in) *itprint,* character (len=128), intent(in) *filepath*)

reads data from file

Parameters

real	[out] u(neq,nxmin:nxmax,nymin:nymax,nzmin:nzmax) : conserved variables
integer	[in] itprint : number of output
string	[in] filepath : path where the output is

Definition at line 135 of file coldens.f90.

4.2.2.5 subroutine coldens_utilities::rotation_x (real, intent(in) *theta,* real, intent(in) *x,* real, intent(in) *y,* real, intent(in) *z,* real, intent(out) *xn,* real, intent(out) *yn,* real, intent(out) *zn*)

Does a rotation around the x axis

Parameters

real	[in], theta : Angle of rotation (in radians)
real	[in], x : original x position in the grid
real	[in], y : original y position in the grid
real	[in], x : original z position in the grid
real	[out], x : final x position in the grid
real	[out], y : final y position in the grid
real	[out], x : final z position in the grid

Definition at line 214 of file coldens.f90.

4.2.2.6 subroutine coldens_utilities::rotation_y (real, intent(in) *theta,* real, intent(in) *x,* real, intent(in) *y,* real, intent(in) *z,* real, intent(out) *xn,* real, intent(out) *yn,* real, intent(out) *zn*)

Does a rotation around the x axis

Parameters

real	[in], theta : Angle of rotation (in radians)
real	[in], x : original x position in the grid
real	[in], y : original y position in the grid
real	[in], x : original z position in the grid
real	[out], x : final x position in the grid
real	[out], y : final y position in the grid

real	[out], x : final z position in the grid

Definition at line 238 of file coldens.f90.

4.2.2.7 subroutine coldens_utilities::rotation_z (real, intent(in) *theta,* real, intent(in) *x,* real, intent(in) *y,* real, intent(in) *z,* real, intent(out) *xn,* real, intent(out) *yn,* real, intent(out) *zn*)

Does a rotation around the x axis

Parameters

real	[in], theta : Angle of rotation (in radians)
real	[in], x : original x position in the grid
real	[in], y : original y position in the grid
real	[in], x : original z position in the grid
real	[out], x : final x position in the grid
real	[out], y : final y position in the grid
real	[out], x : final z position in the grid

Definition at line 260 of file coldens.f90.

4.2.2.8 subroutine coldens_utilities::write_map (character (len=128), intent(in) *fileout,* integer, intent(in) *nxmap,* integer, intent(in) *nymap,* real, dimension(nxmap,nymap), intent(in) *map*)

Writes projection to file

Parameters

integer	[in] itprint : number of output
string	[in] fileout : file where to write
integer	[in] nxmap : Number of X cells in target
integer	[in] nymap : Number of Y cells in target
real	[in] map(nxmap,mymap) : Target map

Definition at line 340 of file coldens.f90.

4.3 constants Module Reference

Module containing physical and asronomical constants.

Variables

- real, parameter pi =acos(-1.)

  π
- real, parameter amh =1.66e-24

  hydrogen mass
- real, parameter mu =1.3

  mean atomic mass
- real, parameter kb =1.38e-16

  Boltzmann constant (cgs)
- real, parameter rg =8.3145e7

  Gas constant (cgs)
- real, parameter ggrav =6.67259e-8

  Gravitational constant (cgs)
- real, parameter clight =2.99E10

speed of light in vacuum (cgs)

- real, parameter msun =1.99E33

  solar radius (cgs)
- real, parameter rsun =6.955e10

  solar mass (cgs)
- real, parameter mjup =1.898E30

  Jupiter mass (cgs)
- real, parameter rjup =7.1492E9

  Jupiter radius (cgs)
- real, parameter au =1.496e13

  1AU in cm
- real, parameter pc =3.0857E18

  1pc in cm
- real, parameter kpc =3.0857E21

  1Kpc in cm
- real, parameter hr =3600.

  1hr in seconds
- real, parameter day =86400.

  1day in seconds
- real, parameter yr =3.1536E7

  1yr in seconds
- real, parameter myr =3.1536E13

  1Myr in seconds

4.4 cooling_chi Module Reference

Cooling module with CHIANTI generated cooling curves.

Functions/Subroutines

- subroutine read_table ()

  Reads the cooling curve table.
- real(kind=8) function coolchi (T)

  Returns the cooling coefficient interpolating the table.
- subroutine coolingchi (dt)

  High level wrapper to apply cooling with CHIANTI tables.

Variables

- real(kind=8), dimension(2, 41) **cooltab**

4.4.1 Detailed Description

Cooling module with CHIANTI generated cooling curves
The location of the tables is assumed to be in src/CHIANTIlib/coolingCHIANTI.tab

4.4.2 Function/Subroutine Documentation

4.4.2.1 real (kind=8) function cooling_chi::coolchi (real, intent(in) *T*)

Parameters

real	[in] T : Temperature K

Definition at line 75 of file cooling_chi.f90.

4.4.2.2 subroutine cooling_chi::coolingchi (real, intent(in) *dt*)

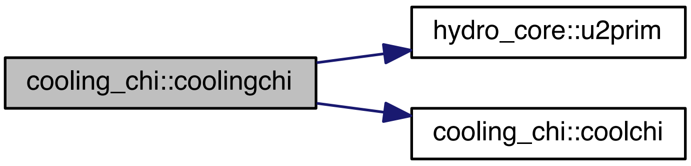
High level wrapper to apply cooling with CHIANTI tables
cooling is applied in the entire domain and updates both the conserved and primitive variables

Parameters

real	[in] dt : timestep (in seconds)

Definition at line 103 of file cooling_chi.f90.

Here is the call graph for this function:



4.4.2.3 subroutine cooling_chi::read_table ( )

Reads the cooling curve table generated by CHUANTI, the location is assumed in /src/CHIANTIlib/coolingCHIAN↩
TI.tab

Definition at line 44 of file cooling_chi.f90.

4.5 cooling_dmc Module Reference

Cooling module with Dalgarno McCray coronal cooling curve.

Functions/Subroutines

- subroutine read_table ()
  Reads the cooling curve table.
- real(kind=8) function cooldmc (T)
  Returns the cooling coefficient interpolating the table.
- subroutine coolingdmc (dt)
  High level wrapper to apply cooling with DMC table.

Variables

- real(kind=8), dimension(2, 41) **cooltab**

4.5.1 Detailed Description

Cooling module with Dalgarno McCray coronal cooling curve
The location of the tables is assumed to be in src/DMClib/coolingDMC.tab, it is read by init subroutine

4.5.2 Function/Subroutine Documentation

4.5.2.1 real (kind=8) function cooling_dmc::cooldmc (real, intent(in) *T*)

Parameters

real	[in] T : Temperature K

Definition at line 77 of file cooling_dmc.f90.

4.5.2.2 subroutine cooling_dmc::coolingdmc (real, intent(in) *dt*)

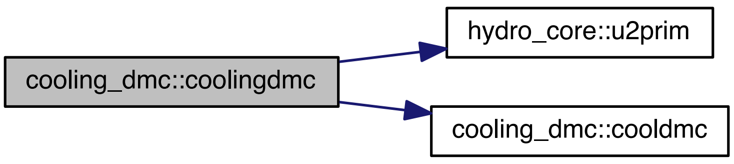
High level wrapper to apply cooling with DMC table
cooling is applied in the entire domain and updates both the conserved and primitive variables

Parameters

real	[in] dt : timestep (in seconds)

Definition at line 104 of file cooling_dmc.f90.

Here is the call graph for this function:



4.5.2.3 subroutine cooling_dmc::read_table ( )

Reads the Dalgarno McCray cooling courve the location is assumed in src/DMClib/coolingDMC.tab, it is read by init subroutine

Definition at line 45 of file cooling_dmc.f90.

4.6 cooling_h Module Reference

Cooling with parametrized cooling and H rate equation.

Functions/Subroutines

- subroutine coolingh (dt)

High level wrapper to apply cooling.

- real(kind=8) function alpha (T)

  calculates the recombination rate (case B)
- real(kind=8) function alpha1 (T)

  calculates the recombination rate to level 1
- real(kind=8) function colf (T)

  calculates the collisional ionization rate
- real(kind=8) function betah (T)

  betaH(T)
- real(kind=8) function aloss (X1, X2, DT, DEN, DH0, TE0)

  Non equilibrium cooling.
- subroutine atomic (dt, uu, tau, radphi)

  Updates the ionization fraction and applpies cooling.

4.6.1 Detailed Description

Cooling with parametrized cooling and H rate equation

4.6.2 Function/Subroutine Documentation

4.6.2.1 real (kind=8) function cooling_h::aloss (real (kind=8), intent(in) *X1,* real (kind=8), intent(in) *X2,* real, intent(in) *DT,* real (kind=8), intent(in) *DEN,* real (kind=8), intent(in) *DH0,* real (kind=8), intent(in) *TE0*)

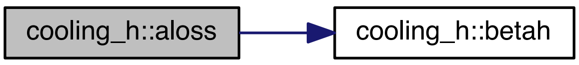
Non-equilibrium energy loss for low temperatures considering the collisional excitation of [O I] and [O II] lines and radiative recombination of H. This cooling rate is multiplied by a factor of 7.033 so that it has the same value as the "coronal equilibrium" cooling rate at a temperature of 44770 K (at temperatures higher than this value, the equilibrium cooling rate is used). The collisional ionization of H and excitation of Lyman-alpha are computed separately, and added to the cooling rate.

Parameters

real8	[in] x1 : initial H ionization fraction
real8	[in] x2 : final H ionization fraction
real	[in] dt : timestep
real8	[in] den : total density of hydrogen
real8	[in] dh0 : density of neutral hydrogen
real8	[in] Te0 : Temperature

Definition at line 163 of file cooling_h.f90.

Here is the call graph for this function:



4.6.2.2 real (kind=8) function cooling_h::alpha (real (kind=8), intent(in) *T*)

calculates the recombination rate (case B)

Parameters

real8	[in] T : Temperature K

Definition at line 79 of file cooling_h.f90.

4.6.2.3 real (kind=8) function cooling_h::alpha1 (real (kind=8), intent(in) *T*)

calculates the recombination rate to level 1

Parameters

real8	[in] T : Temperature K

Definition at line 96 of file cooling_h.f90.

4.6.2.4 subroutine cooling_h::atomic (real, intent(in) *dt,* real, dimension(neq), intent(out) *uu,* real, intent(in) *tau,* real, intent(in) *radphi*)

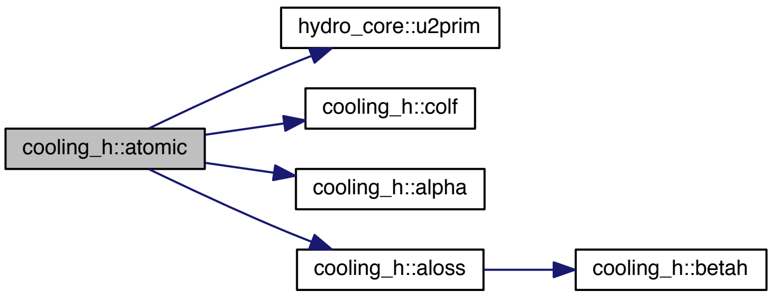
Calculates the new ionization state and energy density using a time dependent ionization calculation and an approximate time dependent cooling calculation

Parameters

real	[in] dt : timestep (seconds)
real	[in] uu(neq) : conserved variablas in one cell
real	[in] tau : optical depth (not in use)
real	[in] radphi : photoionizing rate

Definition at line 260 of file cooling_h.f90.

Here is the call graph for this function:



4.6.2.5 real (kind=8) function cooling_h::betah (real (kind=8), intent(in) *T*)

$\beta_H(T)$

Parameters

real	8[in] T : Temperature K

Definition at line 129 of file cooling_h.f90.

4.6.2.6 real (kind=8) function cooling_h::colf (real (kind=8), intent(in) *T*)

calculates the collisional ionization rate

Parameters

real8[in]	T : Temperature K

Definition at line 112 of file cooling_h.f90.

4.6.2.7 subroutine cooling_h::coolingh (real, intent(in) *dt*)

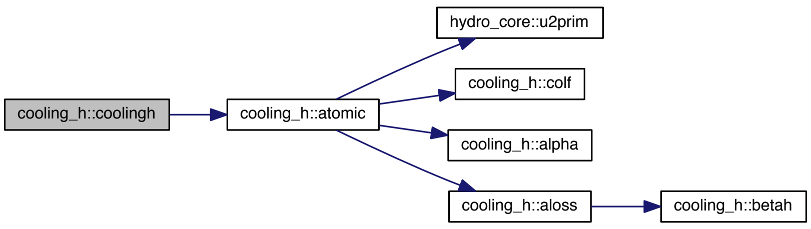
High level wrapper to apply cooling
parametrized cooling curve, uses the ionization state of hydrogen and ties the O I and II to it

Parameters

real	[in] dt : timestep in seconds

Definition at line 43 of file cooling_h.f90.

Here is the call graph for this function:



4.7 difrad Module Reference

Ray tracing Radiative Trasnport.

Functions/Subroutines

- subroutine init_rand ()

  initializes random number generation
- subroutine emdiff (emax)

  calculates the diffuse fotoionization emissivity
- subroutine random_versor (xd, yd, zd)

  returns the 3 components of a random versor
- subroutine starsource (srad, x0, y0, z0, x, y, z, xd, yd, zd)

  Place photon packets at a "star" surface.
- subroutine photons (xl0, yl0, zl0, xd, yd, zd, f)

  Photon trajectories.

- subroutine radbounds ()

  follows the rays across MPI boundaries
- subroutine progress (j, tot)

  Progress bar.
- subroutine diffuse_rad ()

  Diffuse radiation driver.

Variables

- real, parameter a0 =6.3e-18

  Fotoionization cross section.
- integer, parameter nrays =1000000

  Number of rays.
- real, dimension(:,:,:), allocatable ph

  Photoionizing rate.
- real, dimension(:,:,:), allocatable em

  Photoionizing emissivity.
- real, dimension(:,:,:), allocatable photl

  Auxiliary buffer for MPI.
- real, dimension(:,:,:), allocatable photr

  Auxiliary buffer for MPI.
- real, dimension(:,:,:), allocatable photb

  Auxiliary buffer for MPI.
- real, dimension(:,:,:), allocatable phott

  Auxiliary buffer for MPI.
- real, dimension(:,:,:), allocatable photo

  Auxiliary buffer for MPI.
- real, dimension(:,:,:), allocatable photi

  Auxiliary buffer for MPI.
- integer, dimension(6) buffersize

  Auxiliary buffer for MPI.

4.7.1 Detailed Description

Ray tracing Radiative Trasnport

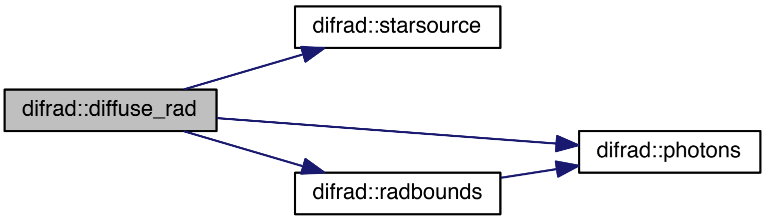
4.7.2 Function/Subroutine Documentation

4.7.2.1 subroutine difrad::diffuse_rad ( )

Upper level wrapper to compute the diffuse photoionization rate

Definition at line 655 of file difrad.f90.

Here is the call graph for this function:



4.7.2.2 subroutine difrad::emdiff (real, intent(out) *emax*)

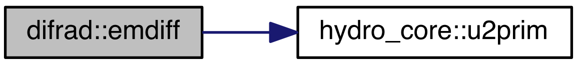
calculates the diffuse fotoionization emissivity in the entire domain

Parameters

real	[out] emax : maximum emissivity in the entire grid

Definition at line 98 of file difrad.f90.

Here is the call graph for this function:



4.7.2.3 subroutine difrad::init_rand ( )

initializes random number generation

Definition at line 56 of file difrad.f90.

4.7.2.4 subroutine difrad::photons (real, intent(in) *xl0*, real, intent(in) *yl0*, real, intent(in) *zl0*, real, intent(in) *xd*, real, intent(in) *yd*, real, intent(in) *zd*, real, intent(inout) *f*)

Launches a photon from cell (xc,yc,zc) in the (xd,yd,zd) direction, with f and ionizing photons, and updates the photoionizing rate

Parameters

real	[in] xl0 : Initial X position
real	[in] yl0 : Initial Y position
real	[in] zl0 : Initial Z position
real	[in] xd : Direction in X
real	[in] yd : Direction in Y
real	[in] zd : Direction in Z
real	[in] f : NUmber of photoionizong photons

Definition at line 252 of file difrad.f90.

4.7.2.5 subroutine difrad::progress (integer(kind=4) *j*, integer(kind=4), intent(in) *tot*)

Progress bar (only tested with Fortran conmpiler) takes a number between 1 and tot

Parameters

integer	[in] j : current iteration
integer	[in] tot : total number of iterartions

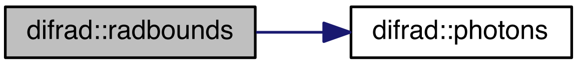
Definition at line 633 of file difrad.f90.

4.7.2.6 subroutine difrad::radbounds ( )

follows the rays across MPI boundaries

Definition at line 455 of file difrad.f90.

Here is the call graph for this function:



4.7.2.7 subroutine difrad::random_versor (real, intent(out) *xd*, real, intent(out) *yd*, real, intent(out) *zd*)

returns the 3 components of a random versor (unit magnitude)

Parameters

real	[out] xd : x component
real	[out] yd : y component
real	[out] zd : z component

Definition at line 149 of file difrad.f90.

4.7.2.8 subroutine difrad::starsource (real, intent(in) *srad*, real, intent(in) *x0*, real, intent(in) *y0*, real, intent(in) *z0*, real, intent(out) *x*, real, intent(out) *y*, real, intent(out) *z*, real, intent(out) *xd*, real, intent(out) *yd*, real, intent(out) *zd*)

returns the random location and direction at a star surface, if the direction goes into the star, the direction is inverted

Parameters

real	[in] Srad : radius of the "star"
real	[in] x0 : X position of the center of the star
real	[in] y0 : Y position of the center of the star
real	[in] y0 : Z position of the center of the star
real	[out] x : random X position at the star surface
real	[out] y : random Y position at the star surface
real	[out] z : random Z position at the star surface
real	[out] xd : random X direction
real	[out] yd : random Y direction
real	[out] zd : random Z direction

Definition at line 187 of file difrad.f90.

4.8 exoplanet Module Reference

Exoplanet module.

Functions/Subroutines

- subroutine init_exo ()

  Module initialization.
- subroutine impose_exo (u, time)

  Inject sources of wind.

Variables

- real rsw

  Stellar radius.
- real tsw

  Stellar wind temperature.
- real vsw

  Stellar wind velocity.
- real dsw

  Stellar Wind Density.
- real b0

  Magnetic Field.
- real rpw

  Planetary radius.
- real tpw

  Planetary wind temperature.
- real vpw

  Planetary wind velocity.
- real **dpw**
- real torb

  planet: orbital period
- real rorb

  orbital radius
- real masss

  Mass of the Star.

- real massp

  Mass of the Planet.
- real xp

  X position of the planet.
- real yp

  Y position of the planet.
- real zp

  Z position of the planet.

4.8.1 Detailed Description

Problem Module for exoplanet

4.8.2 Function/Subroutine Documentation

4.8.2.1 subroutine exoplanet::impose_exo (real, dimension(neq,nxmin:nxmax,nymin:nymax,nzmin:nzmax), intent(out) *u*, real, intent(in) *time*)

Imposes the sources of wond from the star and planet

Parameters

real	[out] u(neq,nxmin:nxmax,nymin:nymax,nzmin:nzmax) : conserver variables
real	[time] time : current integration timr

Definition at line 135 of file exoplanet.f90.

4.8.2.2 subroutine exoplanet::init_exo ( )

Here the parameters of the Star are initialized, and scaled to code units

Definition at line 60 of file exoplanet.f90.

4.9 globals Module Reference

Module containing global variables.

Variables

- real, dimension(:,:,:,:), allocatable u

  conserved varibles
- real, dimension(:,:,:,:), allocatable up

  conserved varibles after 1/2 timestep
- real, dimension(:,:,:,:), allocatable primit

  primitive varibles
- real, dimension(:,:,:,:), allocatable f

  X fluxes.
- real, dimension(:,:,:,:), allocatable g

  Y fluxes.
- real, dimension(:,:,:,:), allocatable h

  Z fluxes.
- real dx

grid spacing in X

- real dy

  grid spacing in Y
- real dz

  grid spacing in Z
- integer, dimension(0:2) coords

  position of neighboring MPI blocks
- integer left

  MPI neighbor in the -x direction.
- integer right

  MPI neighbor in the +x direction.
- integer top

  MPI neighbor in the -y direction.
- integer bottom

  MPI neighbor in the +y direction.
- integer out

  MPI neighbor in the -z direction.
- integer in

  MPI neighbor in the +z direction.
- integer rank

  MPI rank.
- integer comm3d

  Cartesian MPI comunicator.

4.9.1 Detailed Description

This mudules contains variables that are treated as global in the code

4.10 h_alpha_utilities Module Reference

H alpha projection.

Functions/Subroutines

- subroutine init_ha ()

  Initializes data.
- subroutine read_data (u, itprint, filepath)

  reads data from file
- subroutine getxyz (i, j, k, x, y, z)

  gets position of a cell
- subroutine rotation_x (theta, x, y, z, xn, yn, zn)

  Rotation around the X axis.
- subroutine rotation_y (theta, x, y, z, xn, yn, zn)

  Rotation around the Y axis.
- subroutine rotation_z (theta, x, y, z, xn, yn, zn)

  Rotation around the Z axis.
- subroutine fill_map (nxmap, nymap, u, map, dxT, dyT, theta_x, theta_y, theta_z)

  Fill target map.
- subroutine write_ha (fileout, nxmap, nymap, map)

  Writes projection to file.
- subroutine write_rg (fileout, nxmap, nymap, map)

  Writes projection to file in rg format.

4.10.1 Detailed Description

Utilities to compute an H alpha map

4.10.2 Function/Subroutine Documentation

4.10.2.1 subroutine h_alpha_utilities::fill_map (integer, intent(in) *nxmap,* integer, intent(in) *nymap,* real, dimension(neq,nxmin:nxmax,nymin:nymax, nzmin:nzmax), intent(in) *u,* real, dimension(nxmap,nymap), intent(out) *map,* real, intent(in) *dxT,* real, intent(in) *dyT,* real, intent(in) *theta_x,* real, intent(in) *theta_y,* real, intent(in) *theta_z*)

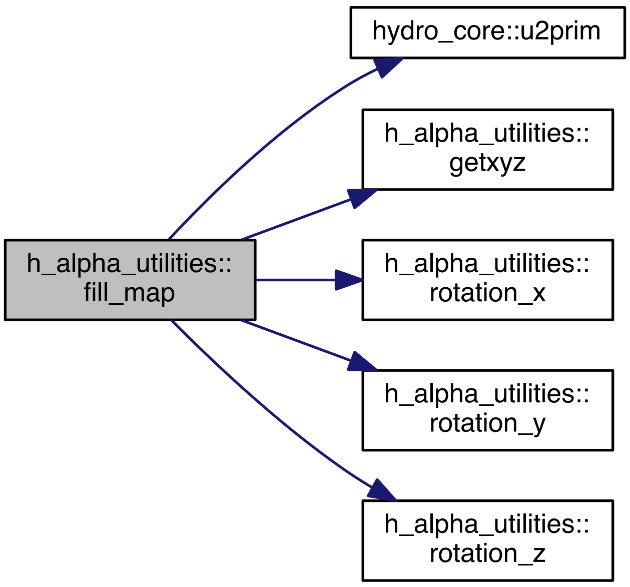
Fills the target map of one MPI block

Parameters

integer	[in] nxmap : Number of X cells in target
integer	[in] nymap : Number of Y cells in target
real	[in] u(neq,nxmin:nxmax,nymin:nymax, nzmin:nzmax) : conserved variables
real	[out] map(nxmap,mymap) : Target map
real	[in] dxT : target pixel width
real	[in] dyT : target pixel height
real	[in] thetax : Rotation around X
real	[in] thetay : Rotation around Y
real	[in] thetaz : Rotation around Z

Definition at line 285 of file h_alpha_proj.f90.

Here is the call graph for this function:


hydro_core::u2prim
h_alpha_utilities:: getxyz
h_alpha_utilities:: fill_map
h_alpha_utilities:: rotation_x
h_alpha_utilities:: rotation_y
h_alpha_utilities:: rotation_z

4.10.2.2 subroutine h_alpha_utilities::getxyz (integer, intent(in) *i*, integer, intent(in) *j*, integer, intent(in) *k*, real, intent(out) *x*, real, intent(out) *y*, real, intent(out) *z*)

Returns the position and spherical radius calculated with respect to the center of the grid

Parameters

integer	[in] i : cell index in the x direction
integer	[in] j : cell index in the y direction
integer	[in] k : cell index in the z direction
real	[in] x : x position in the grid
real	[in] y : y position in the grid
real	[in] z : z position in the grid

Definition at line 187 of file h_alpha_proj.f90.

4.10.2.3 subroutine h_alpha_utilities::init_ha ( )

Initializes data, MPI and other stuff

Definition at line 35 of file h_alpha_proj.f90.

4.10.2.4 subroutine h_alpha_utilities::read_data (real, dimension(neq,nxmin:nxmax,nymin:nymax,nzmin:nzmax), intent(out) *u*, integer, intent(in) *itprint*, character (len=128), intent(in) *filepath*)

reads data from file

Parameters

real	[out] u(neq,nxmin:nxmax,nymin:nymax,nzmin:nzmax) : conserved variables
integer	[in] itprint : number of output
string	[in] filepath : path where the output is

Definition at line 134 of file h_alpha_proj.f90.

4.10.2.5 subroutine h_alpha_utilities::rotation_x (real, intent(in) *theta*, real, intent(in) *x*, real, intent(in) *y*, real, intent(in) *z*, real, intent(out) *xn*, real, intent(out) *yn*, real, intent(out) *zn*)

Does a rotation around the x axis

Parameters

real	[in], theta : Angle of rotation (in radians)
real	[in], x : original x position in the grid
real	[in], y : original y position in the grid
real	[in], x : original z position in the grid
real	[out], x : final x position in the grid
real	[out], y : final y position in the grid
real	[out], x : final z position in the grid

Definition at line 213 of file h_alpha_proj.f90.

4.10.2.6 subroutine h_alpha_utilities::rotation_y (real, intent(in) *theta*, real, intent(in) *x*, real, intent(in) *y*, real, intent(in) *z*, real, intent(out) *xn*, real, intent(out) *yn*, real, intent(out) *zn*)

Does a rotation around the x axis

Parameters

real	[in], theta : Angle of rotation (in radians)

real	[in], x : original x position in the grid
real	[in], y : original y position in the grid
real	[in], x : original z position in the grid
real	[out], x : final x position in the grid
real	[out], y : final y position in the grid
real	[out], x : final z position in the grid

Definition at line 237 of file h_alpha_proj.f90.

4.10.2.7 subroutine h_alpha_utilities::rotation_z (real, intent(in) *theta,* real, intent(in) *x,* real, intent(in) *y,* real, intent(in) *z,* real, intent(out) *xn,* real, intent(out) *yn,* real, intent(out) *zn*)

Does a rotation around the x axis

Parameters

real	[in], theta : Angle of rotation (in radians)
real	[in], x : original x position in the grid
real	[in], y : original y position in the grid
real	[in], x : original z position in the grid
real	[out], x : final x position in the grid
real	[out], y : final y position in the grid
real	[out], x : final z position in the grid

Definition at line 259 of file h_alpha_proj.f90.

4.10.2.8 subroutine h_alpha_utilities::write_ha (character (len=128), intent(in) *fileout,* integer, intent(in) *nxmap,* integer, intent(in) *nymap,* real, dimension(nxmap,nymap), intent(in) *map*)

Writes projection to file

Parameters

integer	[in] itprint : number of output
string	[in] fileout : file where to write
integer	[in] nxmap : Number of X cells in target
integer	[in] nymap : Number of Y cells in target
real	[in] map(nxmap,mymap) : Target map

Definition at line 362 of file h_alpha_proj.f90.

4.10.2.9 subroutine h_alpha_utilities::write_rg (character (len=128), intent(in) *fileout,* integer, intent(in) *nxmap,* integer, intent(in) *nymap,* real, dimension(nxmap,nymap), intent(in) *map*)

Writes projection to file

Parameters

integer	[in] itprint : number of output
string	[in] fileout : file where to write
integer	[in] nxmap : Number of X cells in target
integer	[in] nymap : Number of Y cells in target
real	[in] map(nxmap,mymap) : Target map

Definition at line 391 of file h_alpha_proj.f90.

4.11 hll Module Reference

HLL approximate Riemann solver module.

Functions/Subroutines

- subroutine prim2fhll (priml, primr, ff)

  Solves the Riemann problem at the interface PL,PR using the HLL solver.
- subroutine hllfluxes (choice)

  Calculates HLL fluxes from the primitive variables on all the domain.

4.11.1 Detailed Description

The module contains the routines needed to Solve the Riemann problem in the entire domain and return the physical fluxes in x,y,z with the HLL solver

4.11.2 Function/Subroutine Documentation

4.11.2.1 subroutine hll::hllfluxes (integer, intent(in) *choice*)

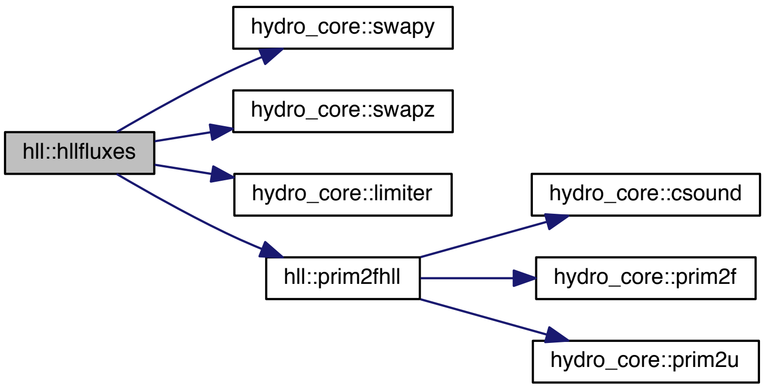
Calculates HLL fluxes from the primitive variables on all the domain

Parameters

integer	[in] choice : 1, uses primit for the 1st half of timestep (first order) 2 uses primit for second order timestep

Definition at line 93 of file hll.f90.

Here is the call graph for this function:



4.11.2.2 subroutine hll::prim2fhll (real, dimension(neq), intent(in) *priml*, real, dimension(neq), intent(in) *primr*, real, dimension(neq), intent(inout) *ff*)

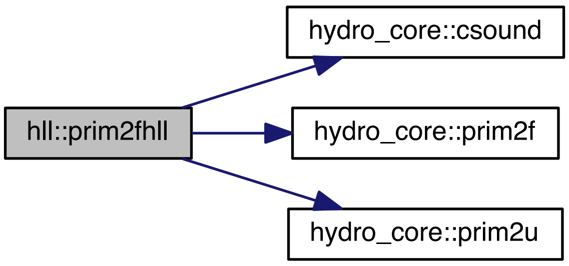
Solves the Riemann problem at the interface betweem PL and PR using the HLL solver
The fluxes are computed in the X direction, to obtain the y ans z directions a swap is performed

Parameters

real	[in] primL : primitives at the Left state
real	[in] primR : primitives at the Right state
real	[out] ff : fluxes at the interface ($F_{i+1/2}$)

Definition at line 48 of file hll.f90.

Here is the call graph for this function:



4.12 hllc Module Reference

HLLC approximate Riemann solver module.

Functions/Subroutines

- subroutine prim2fhllc (priml, primr, ff)

  Solves the Riemann problem at the interface PL,PR using the HLLC solver.
- subroutine hllcfluxes (choice)

  Calculates HLLC fluxes from the primitive variables on all the domain.

4.12.1 Detailed Description

The module contains the routines needed to Solve the Riemann problem in the entire domain and return the physical fluxes in x,y,z with the HLLC solver

4.12.2 Function/Subroutine Documentation

4.12.2.1 subroutine hllc::hllcfluxes (integer, intent(in) *choice*)

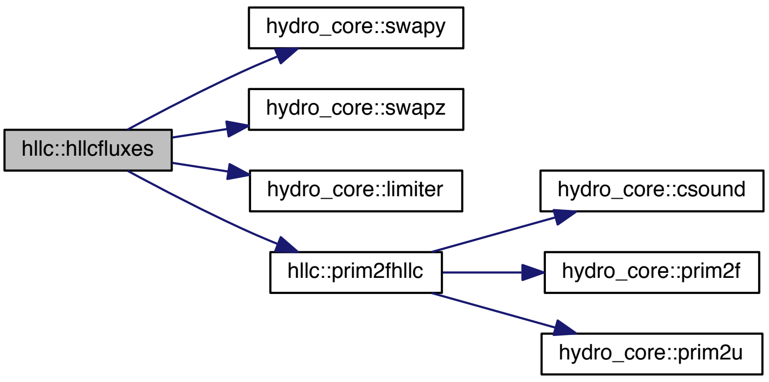
Calculates HLLC fluxes from the primitive variables on all the domain

Parameters

integer	[in] choice : 1, uses primit for the 1st half of timestep (first order) 2 uses primit for second order timestep

Definition at line 145 of file hllc.f90.

Here is the call graph for this function:



4.12.2.2 subroutine hllc::prim2fhllc (real, dimension(neq), intent(in) *priml,* real, dimension(neq), intent(in) *primr,* real, dimension(neq), intent(inout) *ff*)

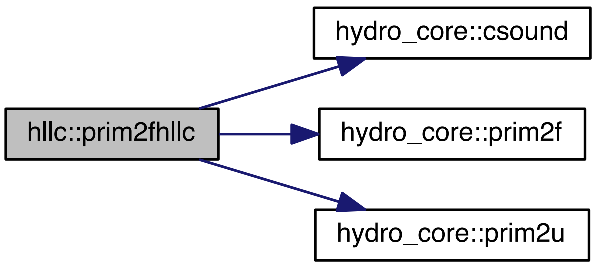
Solves the Riemann problem at the interface betweem PL and PR using the HLLC solver
The fluxes are computed in the X direction, to obtain the y ans z directions a swap is performed

Parameters

real	[in] primL : primitives at the Left state
real	[in] primR : primitives at the Right state
real	[out] ff : fluxes at the interface ($F_{i+1/2}$)

Definition at line 47 of file hllc.f90.

Here is the call graph for this function:



4.13 hlld Module Reference

HLLD approximate Riemann solver module.

Functions/Subroutines

- subroutine prim2fhlld (priml, primr, ff)

  Solves the Riemann problem at the interface PL,PR using the HLLD solver.
- subroutine hlldfluxes (choice)

  Calculates HLLD fluxes from the primitive variables on all the domain.

4.13.1 Detailed Description

The module contains the routines needed to Solve the Riemann problem in the entire domain and return the physical fluxes in x,y,z with the HLLD solver

4.13.2 Function/Subroutine Documentation

4.13.2.1 subroutine hlld::hlldfluxes (integer, intent(in) *choice*)

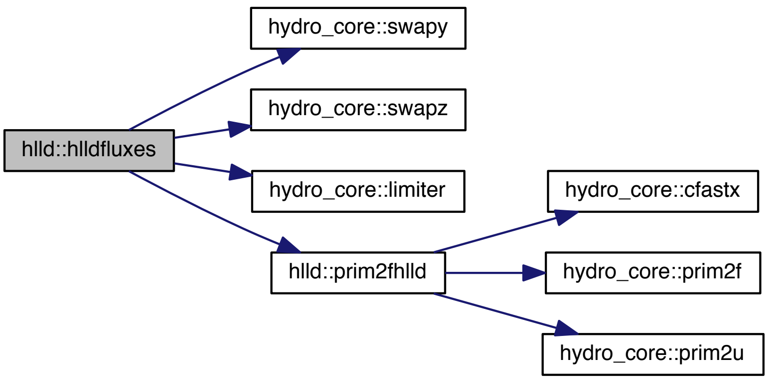
Calculates HLLD fluxes from the primitive variables on all the domain

Parameters

integer	[in] choice : 1, uses primit for the 1st half of timestep (first order) 2 uses primit for second order timestep

Definition at line 319 of file hlld.f90.

Here is the call graph for this function:



4.13.2.2 subroutine hlld::prim2fhlld (real, dimension(neq), intent(in) *priml,* real, dimension(neq), intent(in) *primr,* real, dimension(neq), intent(inout) *ff*)

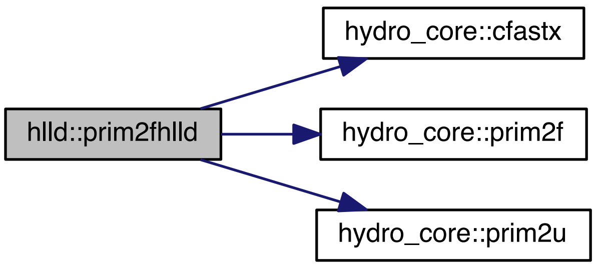
Solves the Riemann problem at the interface betweem PL and PR using the HLLD solver
The fluxes are computed in the X direction, to obtain the y ans z directions a swap is performed

Parameters

real	[in] primL : primitives at the Left state
real	[in] primR : primitives at the Right state
real	[out] ff : fluxes at the interface ($F_{i+1/2}$)

Definition at line 49 of file hlld.f90.

Here is the call graph for this function:



4.14 hlle Module Reference

HLLE approximate Riemann solver module.

Functions/Subroutines

- subroutine prim2fhlle (priml, primr, ff)

  Solves the Riemann problem at the interface PL,PR using the HLLE solver.
- subroutine hllefluxes (choice)

  Calculates HLLE fluxes from the primitive variables on all the domain.

4.14.1 Detailed Description

The module contains the routines needed to Solve the Riemann problem in the entire domain and return the physical fluxes in x,y,z with the HLLE solver

4.14.2 Function/Subroutine Documentation

4.14.2.1 subroutine hlle::hllefluxes (integer, intent(in) *choice*)

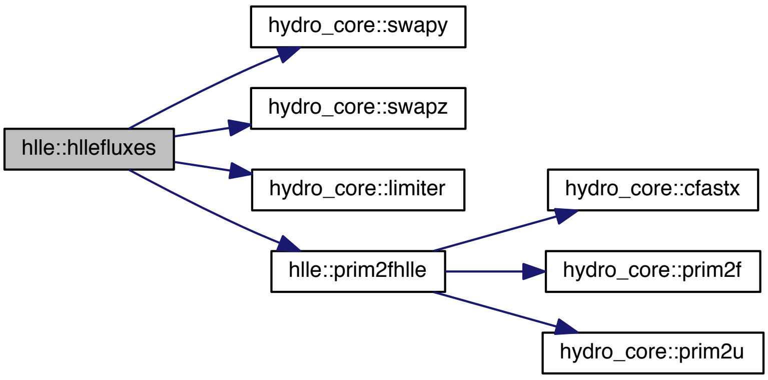
Calculates HLLE fluxes from the primitive variables on all the domain

Parameters

integer	[in] choice : 1, uses primit for the 1st half of timestep (first order) 2 uses primit for second order timestep

Definition at line 96 of file hlle.f90.

Here is the call graph for this function:



4.14.2.2 subroutine hlle::prim2fhlle (real, dimension(neq), intent(in) *priml,* real, dimension(neq), intent(in) *primr,* real, dimension(neq), intent(inout) *ff*)

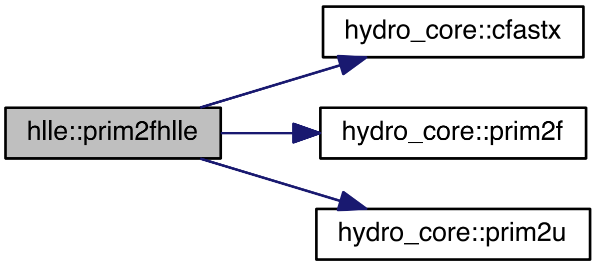
Solves the Riemann problem at the interface betweem PL and PR using the HLLE solver
The fluxes are computed in the X direction, to obtain the y ans z directions a swap is performed

Parameters

real	[in] primL : primitives at the Left state
real	[in] primR : primitives at the Right state
real	[out] ff : fluxes at the interface ($F_{i+1/2}$)

Definition at line 51 of file hlle.f90.

Here is the call graph for this function:



4.15 hydro_core Module Reference

Basic hydro (and MHD) subroutines utilities.

Functions/Subroutines

- subroutine u2prim (uu, prim, T)

  Computes the primitive variables and temperature from conserved variables on a single cell.
- subroutine calcprim (u, primit)

  Updated the primitives, using the conserved variables in the entire domain.
- subroutine prim2u (prim, uu)

  Computes the conserved conserved variables from the primitives in a single cell.
- subroutine prim2f (prim, ff)

  Computes the Euler Fluxes in one cell.
- subroutine swapy (var, neq)

  Swaps the x and y components in a cell.
- subroutine swapz (var, neq)

  Swaps the x and z components in a cell.
- subroutine csound (p, d, cs)

  Computes the sound speed.
- subroutine cfast (p, d, bx, by, bz, cfx, cfy, cfz)

  Computes the fast magnetosonic speeds in the 3 coordinates.
- subroutine cfastx (prim, cfX)

  Computes the fast magnetosonic speed in the x direction.
- subroutine get_timestep (dt)

  Otains the timestep allowed by the CFL condition in the entire.
- subroutine limiter (PLL, PL, PR, PRR, neq)

  Performs a linear reconstruction of the primitive variables.

4.15.1 Detailed Description

This module contains subroutines and utilities that are the core of the hydro (and MHD) that are common to most implementations and will be used for the different specific solvers

4.15.2 Function/Subroutine Documentation

4.15.2.1 subroutine hydro_core::calcprim (real, dimension(neq,nxmin:nxmax,nymin:nymax,nzmin:nzmax), intent(in) *u*, real, dimension(neq,nxmin:nxmax,nymin:nymax,nzmin:nzmax), intent(out) *primit*)

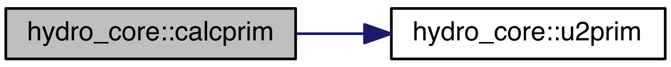
Updated the primitives, using the conserved variables in the entire domain

Parameters

real	[in] u(neq,nxmin:nxmax,nymin:nymax,nzmin:nzmax) : conserved variables
real	[out] prim(neq,nxmin:nxmax,nymin:nymax,nzmin:nzmax) : primitive variables

Definition at line 116 of file hydro_core.f90.

Here is the call graph for this function:



4.15.2.2 subroutine hydro_core::cfast (real, intent(in) *p*, real, intent(in) *d*, real, intent(in) *bx*, real, intent(in) *by*, real, intent(in) *bz*, real, intent(out) *cfx*, real, intent(out) *cfy*, real, intent(out) *cfz*)

Computes the fast magnetosonic speeds in the 3 coordinates

Parameters

real	[in] p : value of pressure
real	[in] d : value of density
real	[in] Bx : value of the x component of the magnetic field
real	[in] By : value of the y component of the magnetic field
real	[in] Bz : value of the z component of the magnetic field
real	[out] csx : fast magnetisonic speed in x
real	[out] csy : fast magnetisonic speed in y
real	[out] csz : fast magnetisonic speed in z

Definition at line 325 of file hydro_core.f90.

4.15.2.3 subroutine hydro_core::cfastx (real, dimension(neq), intent(in) *prim*, real, intent(out) *cfX*)

Computes the fast magnetosonic speed in the x direction

Parameters

real	[in] prim(neq) : vector with the primitives in one cell

Definition at line 350 of file hydro_core.f90.

4.15.2.4 subroutine hydro_core::csound (real, intent(in) *p*, real, intent(in) *d*, real, intent(out) *cs*)

Computes the sound speed

Parameters

real	[in] p : value of pressure
real	[in] d : value of density
real	[out] cs : sound speed

Definition at line 299 of file hydro_core.f90.

4.15.2.5 subroutine hydro_core::get_timestep (real, intent(out) *dt*)

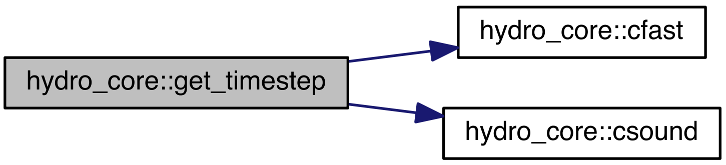
Otains the timestep allowed by the CFL condition in the entire domain using the global primitives

Parameters

real	[out] : Δt allowed by the CFL condition

Definition at line 373 of file hydro_core.f90.

Here is the call graph for this function:



4.15.2.6 subroutine hydro_core::limiter (real, dimension(neq), intent(in) *PLL*, real, dimension(neq), intent(inout) *PL*, real, dimension(neq), intent(inout) *PR*, real, dimension(neq), intent(in) *PRR*, integer, intent(in) *neq*)

returns a linear reconstruction of the variables at the interface beteen the primitives PLL, PL, PR, PRR
The reconstruction is made with a slope limiter chosen at compilation time (i.e. set on the Makefile)

Parameters

real	[in] : primitives at the left of the left state
real	[inout] : primitives at the left state
real	[inout] : primitives at the right state
real	[in] : primitives at the right of the right state
real	[in] : number of equations

Definition at line 437 of file hydro_core.f90.

4.15.2.7 subroutine hydro_core::prim2f (real, dimension(neq), intent(in) *prim*, real, dimension(neq), intent(out) *ff*)

Computes the Euler Fluxes in one cell, using the primitices
It returns the flux in the x direction (i.e. F), the y and z fluxes can be obtained swaping the respective entries (see swapy and swapz subroutines)

Parameters

real	[in] prim(neq) : primitives in one cell
real	[out] ff(neq) : Euler Fluxes (x direction)

Definition at line 196 of file hydro_core.f90.

4.15.2.8 subroutine hydro_core::prim2u (real, dimension(neq), intent(in) *prim*, real, dimension(neq), intent(out) *uu*)

Computes the conserved conserved variables from the primitives in a single cell

Parameters

real	[in] prim(neq) : primitives in one cell
real	[out] uu(neq) : conserved varibles in one cell

Definition at line 155 of file hydro_core.f90.

4.15.2.9 subroutine hydro_core::swapy (real, dimension(neq), intent(inout) *var,* integer, intent(in) *neq*)

Swaps the x and y components in a cell.

Parameters

real	[inout] var(neq) : variable to be swapped
real	[in] neq : number of equations in the code

Definition at line 247 of file hydro_core.f90.

4.15.2.10 subroutine hydro_core::swapz (real, dimension(neq), intent(inout) *var,* integer, intent(in) *neq*)

Swaps the x and z components in a cell.

Parameters

real	[inout] var(neq) : variable to be swapped
real	[in] neq : number of equations in the code

Definition at line 273 of file hydro_core.f90.

4.15.2.11 subroutine hydro_core::u2prim (real, dimension(neq), intent(in) *uu,* real, dimension(neq), intent(out) *prim,* real, intent(out) *T*)

Computes the primitive variables and temperature from conserved variables on a single cell

Parameters

real	[in] uu(neq) : conserved variables in one cell
real	[out] prim(neq) : primitives in one cell
real	[out] T : Temperature [K]

Definition at line 44 of file hydro_core.f90.

4.16 hydro_solver Module Reference

Advances the simulation one timestep.

Functions/Subroutines

- subroutine viscosity ()

  Adds artificial viscosity to the conserved variables.
- subroutine step (dt)

  Upwind timestep.
- subroutine tstep (time, dt)

  High level wrapper to advancce the simulation.

4.16.1 Detailed Description

Advances the solution from t to $t + \Delta t$

4.16.2 Function/Subroutine Documentation

4.16.2.1 subroutine hydro_solver::step (real, intent(in) *dt*)

Performs the upwind timestep according to

$$U_i^{n+1} = U_i^n - \frac{\Delta t}{\Delta x}\left[F_{i+1/2}^{n+1/2} - F_{i-1/2}^{n+1/2}\right]$$

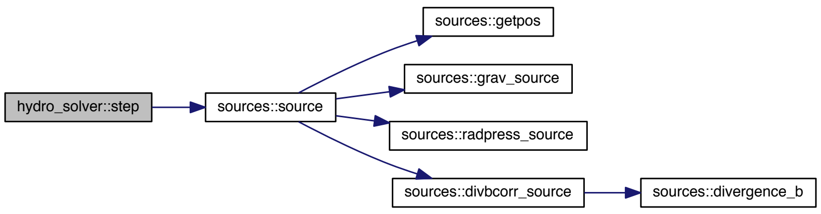
(in 3D), it takes U^{n+1}=up from the global variables and U^{n}=u

Parameters

real	[in] dt : timestep

Definition at line 82 of file hydro_solver.f90.

Here is the call graph for this function:



4.16.2.2 subroutine hydro_solver::tstep (real, intent(in) *time,* real, intent(in) *dt*)

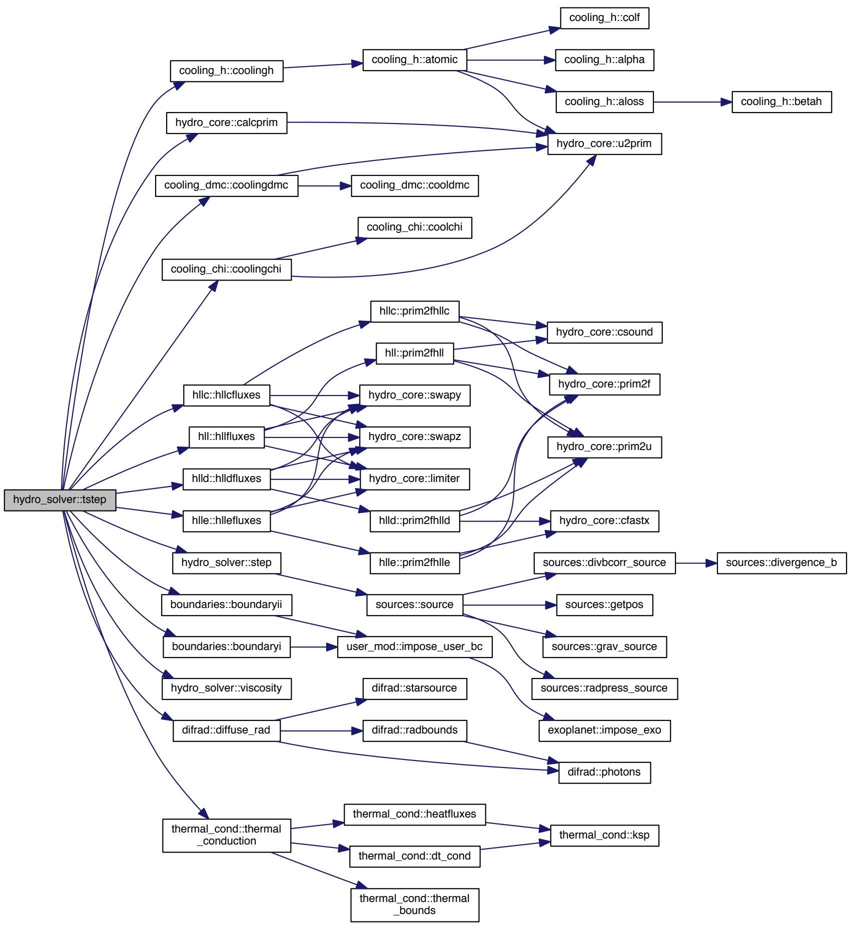
High level wrapper to advancce the simulation
The variables are taken from the globals module.

Parameters

real	[in] time : integration time
real	[in] dt : timestep

Definition at line 126 of file hydro_solver.f90.

Here is the call graph for this function:



4.16.2.3 subroutine hydro_solver::viscosity ( )

Adds artificial viscosity to the conserved variables
Takes the variables from the globals module and it assumes that the up are the stepped variables, while u are unstepped

Definition at line 52 of file hydro_solver.f90.

4.17 init Module Reference

Guacho-3D initialization.

Functions/Subroutines

- subroutine initmain (time, tprint, itprint)

  Main initialization routine.
- subroutine initflow (itprint)

  Initializes the conserved variables, in the globals module.

4.17.1 Detailed Description

This module contains the routines needed to initializa the code, it also initiaizes all the modules set by the user.

4.17.2 Function/Subroutine Documentation

4.17.2.1 subroutine init::initflow (integer, intent(inout) *itprint*)

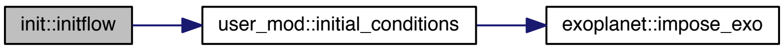
Initializes the conserved variables, in the globals module

Parameters

real	[inout] itprint : number of current output

Definition at line 417 of file init.f90.

Here is the call graph for this function:



4.17.2.2 subroutine init::initmain (real, intent(out) *time,* real, intent(out) *tprint,* integer, intent(out) *itprint*)

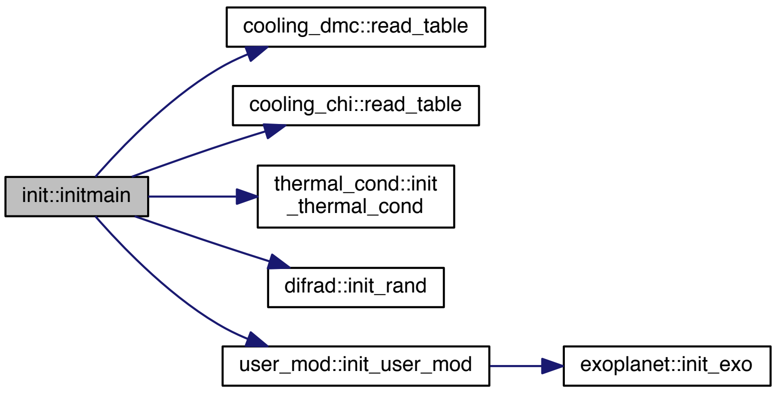
This subsroutine initializes all the variables in the globals module, MPI, cooling and user_mod routines; and outputs to screen the main parameters used in the run

Parameters

real	[out] time : time (code units)
real	[out] tprint : time of next output
integer	[out] itprint : number of next output

Definition at line 42 of file init.f90.

Here is the call graph for this function:



4.18 lyman_alpha_utilities Module Reference

Lyman_alpha_utilities.

Functions/Subroutines

- subroutine init_la ()

  Initializes data.
- subroutine read_data (u, itprint, filepath)

  reads data from file
- subroutine getxyz (i, j, k, x, y, z)

  gets position of a cell
- subroutine rotation_x (theta, x, y, z, xn, yn, zn)

  Rotation around the X axis.
- subroutine rotation_y (theta, x, y, z, xn, yn, zn)

  Rotation around the Y axis.
- subroutine rotation_z (theta, x, y, z, xn, yn, zn)

  Rotation around the Z axis.
- subroutine fill_map (nxmap, nymap, nvmap, vmin, vmax, u, map, dxT, dyT, theta_x, theta_y, theta_z)

  Fill target map.
- subroutine write_la (itprint, filepath, nxmap, nymap, nvmap, map)

  Writes projection to file.
- subroutine phigauss (T, vzn, vmin, vmax, nvmap, profile)

  This routine computes a gaussian line profile.

4.18.1 Detailed Description

Utilities to compute the Lyman-

4.18.2 Function/Subroutine Documentation

4.18.2.1 subroutine lyman_alpha_utilities::fill_map (integer, intent(in) *nxmap,* integer, intent(in) *nymap,* integer, intent(in) *nvmap,* real, intent(in) *vmin,* real, intent(in) *vmax,* real, dimension(neq,nxmin:nxmax,nymin:nymax, nzmin:nzmax), intent(in) *u,* real, dimension(nxmap,nymap,nvmap), intent(out) *map,* real, intent(in) *dxT,* real, intent(in) *dyT,* real, intent(in) *theta_x,* real, intent(in) *theta_y,* real, intent(in) *theta_z*)

Fills the target map of one MPI block

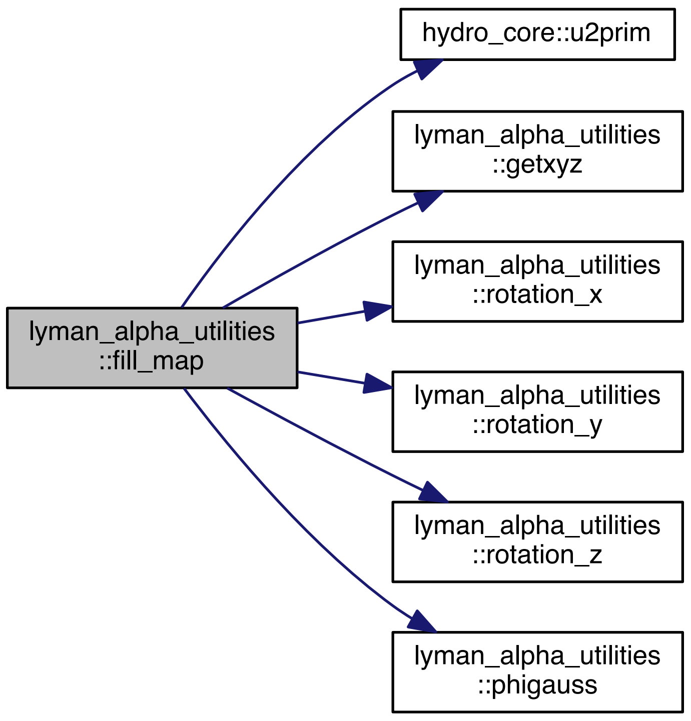
Parameters

integer	[in] nxmap : Number of X cells in target
integer	[in] nymap : Number of Y cells in target
real	[in] u(neq,nxmin:nxmax,nymin:nymax, nzmin:nzmax) : conserved variables
real	[out] map(nxmap,mymap) : Target map
real	[in] dxT : target pixel width
real	[in] dyT : target pixel height
real	[in] thetax : Rotation around X
real	[in] thetay : Rotation around Y

real	[in] thetaz : Rotation around Z

Definition at line 284 of file lyman_alpha_tau.f90.

Here is the call graph for this function:


hydro_core::u2prim
lyman_alpha_utilities ::getxyz
lyman_alpha_utilities ::rotation_x
lyman_alpha_utilities ::fill_map
lyman_alpha_utilities ::rotation_y
lyman_alpha_utilities ::rotation_z
lyman_alpha_utilities ::phigauss

4.18.2.2 subroutine lyman_alpha_utilities::getxyz (integer, intent(in) *i*, integer, intent(in) *j*, integer, intent(in) *k*, real, intent(out) *x*, real, intent(out) *y*, real, intent(out) *z*)

Returns the position and spherical radius calculated with respect to the center of the grid

Parameters

integer	[in] i : cell index in the x direction
integer	[in] j : cell index in the y direction
integer	[in] k : cell index in the z direction
real	[in] x : x position in the grid
real	[in] y : y position in the grid
real	[in] z : z position in the grid

Definition at line 185 of file lyman_alpha_tau.f90.

4.18.2.3 subroutine lyman_alpha_utilities::init_la ( )

Initializes data, MPI and other stuff

Definition at line 36 of file lyman_alpha_tau.f90.

4.18.2.4 subroutine lyman_alpha_utilities::phigauss (real, intent(in) *T,* real, intent(in) *vzn,* real, intent(in) *vmin,* real, intent(in) *vmax,* integer, intent(in) *nvmap,* real, dimension(nvmap), intent(out) *profile*)

This routine computes a gaussian line profile

Definition at line 385 of file lyman_alpha_tau.f90.

4.18.2.5 subroutine lyman_alpha_utilities::read_data (real, dimension(neq,nxmin:nxmax,nymin:nymax,nzmin:nzmax), intent(out) *u,* integer, intent(in) *itprint,* character (len=128), intent(in) *filepath*)

reads data from file

Parameters

real	[out] u(neq,nxmin:nxmax,nymin:nymax,nzmin:nzmax) : conserved variables
integer	[in] itprint : number of output
string	[in] filepath : path where the output is

Definition at line 135 of file lyman_alpha_tau.f90.

4.18.2.6 subroutine lyman_alpha_utilities::rotation_x (real, intent(in) *theta,* real, intent(in) *x,* real, intent(in) *y,* real, intent(in) *z,* real, intent(out) *xn,* real, intent(out) *yn,* real, intent(out) *zn*)

Does a rotation around the x axis

Parameters

real	[in], theta : Angle of rotation (in radians)
real	[in], x : original x position in the grid
real	[in], y : original y position in the grid
real	[in], x : original z position in the grid
real	[out], x : final x position in the grid
real	[out], y : final y position in the grid
real	[out], x : final z position in the grid

Definition at line 211 of file lyman_alpha_tau.f90.

4.18.2.7 subroutine lyman_alpha_utilities::rotation_y (real, intent(in) *theta,* real, intent(in) *x,* real, intent(in) *y,* real, intent(in) *z,* real, intent(out) *xn,* real, intent(out) *yn,* real, intent(out) *zn*)

Does a rotation around the x axis

Parameters

real	[in], theta : Angle of rotation (in radians)
real	[in], x : original x position in the grid
real	[in], y : original y position in the grid
real	[in], x : original z position in the grid
real	[out], x : final x position in the grid
real	[out], y : final y position in the grid
real	[out], x : final z position in the grid

Definition at line 235 of file lyman_alpha_tau.f90.

4.18.2.8 subroutine lyman_alpha_utilities::rotation_z (real, intent(in) *theta,* real, intent(in) *x,* real, intent(in) *y,* real, intent(in) *z,* real, intent(out) *xn,* real, intent(out) *yn,* real, intent(out) *zn*)

Does a rotation around the x axis

Parameters

real	[in], theta : Angle of rotation (in radians)
real	[in], x : original x position in the grid
real	[in], y : original y position in the grid
real	[in], x : original z position in the grid
real	[out], x : final x position in the grid
real	[out], y : final y position in the grid
real	[out], x : final z position in the grid

Definition at line 257 of file lyman_alpha_tau.f90.

4.18.2.9 subroutine lyman_alpha_utilities::write_la (integer, intent(in) *itprint,* character (len=128), intent(in) *filepath,* integer, intent(in) *nxmap,* integer, intent(in) *nymap,* integer, intent(in) *nvmap,* real, dimension(nxmap,nymap,nvmap), intent(in) *map*)

Writes projection to file

Parameters

integer	[in] itprint : number of output
string	[in] filepath : path where to write
integer	[in] nxmap : Number of X cells in target
integer	[in] nymap : Number of Y cells in target
integer	[in] nvmap : Number of velocity channels
real	[in] map(nxmap,mymap) : Target map

Definition at line 360 of file lyman_alpha_tau.f90.

4.19 out_silo_module Module Reference

Output in Silo (+HDF5) Format.

Functions/Subroutines

- subroutine writeblocks (itprint)

  Writes Data, one file per processor.
- subroutine writemaster (itprint)

  Writes the Master File.
- subroutine outputsilo (itprint)

  Upper level wrapper.

4.19.1 Detailed Description

This module writes the ouput in SILO (HDF5) format

4.19.2 Function/Subroutine Documentation

4.19.2.1 subroutine out_silo_module::outputsilo (integer, intent(in) *itprint*)

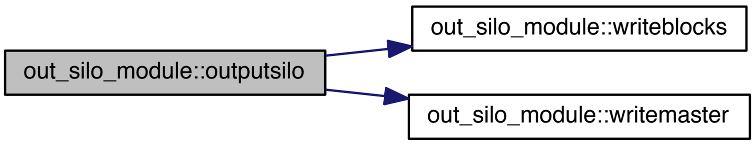
Upper level wrapper for the SILO output

Parameters

integer	[in] itprint : number of output

Definition at line 347 of file Out_Silo_Module.f90.

Here is the call graph for this function:


out_silo_module::outputsilo
out_silo_module::writeblocks
out_silo_module::writemaster

4.19.2.2 subroutine out_silo_module::writeblocks (integer, intent(in) *itprint*)

Writes Data in silo format one file per processor

Parameters

integer	[in] itprint : number of output

Definition at line 44 of file Out_Silo_Module.f90.

4.19.2.3 subroutine out_silo_module::writemaster (integer, intent(in) *itprint*)

Writes the master file with the metadata and multivars

Parameters

integer	[in] itprint : number of output

Definition at line 198 of file Out_Silo_Module.f90.

4.20 output Module Reference

Writes output.

Functions/Subroutines

- subroutine write_output (itprint)

  Writes output.

4.20.1 Detailed Description

This module writes the ouput in the formats specified in the makefile

4.20.2 Function/Subroutine Documentation

4.20.2.1 subroutine output::write_output (integer, intent(in) *itprint*)

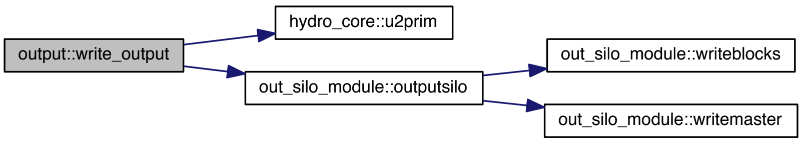
Writes output, the format is chosen in makefile
Supported formats are ∗.dat (ASCII), ∗.bin and VTK (both BINARY), Silo (+hdf5)

Parameters

integer	[in] itprint : number of output

Definition at line 41 of file output.f90.

Here is the call graph for this function:


output::write_output
hydro_core::u2prim
out_silo_module::outputsilo
out_silo_module::writeblocks
out_silo_module::writemaster

4.21 parameters Module Reference

Parameters module.

Variables

- character(len=128), parameter outputpath ='/datos/esquivel/EXO-GUACHO/P5/'

  Path used to write the output.
- character(len=128), parameter workdir ='./'

  working directory
- integer, parameter neqdyn =8

  num. of eqs (+scal)
- integer, parameter ndim =3

  num. of dimensions
- integer, parameter npas =2

  num. of passive scalars
- integer, parameter nghost =2

  num. of ghost cells
- integer, parameter nxtot =600

  Total grid size in X.
- integer, parameter nytot =150

  Total grid size in Y.
- integer, parameter nztot =600

  Total grid size in Z.
- integer, parameter mpicol =4

  number of MPI blocks in X
- integer, parameter mpirow =2

  number of MPI blocks in Y
- integer, parameter mpirowz =4

  number of MPI blocks in Z
- integer, parameter np =mpicol∗mpirow∗mpirowz

  total number of MPI processes
- real, parameter xmax =1.

  grid extent in X (code units)

- real, parameter ymax =0.25

    grid extent in Y (code units)
- real, parameter zmax =1.

    grid extent in Z (code units)
- real, parameter xphys =0.30∗AU

    grid extent in X (pohysical units, cgs)
- real, parameter cv =1.5

    Specific heat at constant volume (/R)
- real, parameter gamma =(cv+1.)/cv

    Cp/Cv.
- real, parameter t0 =1.e4

    reference temperature (to set cs)
- real, parameter rsc =xphys/xmax

    distance scaling
- real, parameter rhosc =amh∗mu

    mass density scaling
- real, parameter tempsc =T0∗gamma

    Temperature scaling.
- real, parameter vsc2 = gamma∗Rg∗T0/mu

    Velocity scaling.
- real, parameter psc = rhosc∗vsc2

    Pressure scaling.
- real, parameter tsc =rsc/sqrt(vsc2)

    time scaling
- real, parameter bsc = sqrt(4.0∗pi∗Psc)

    magnetic fiewld scaling
- real, parameter tmax = 3.8∗day/tsc

    Maximum integration time.
- real, parameter dtprint = 0.025∗day/tsc

    interval between consecutive outputs
- real, parameter cfl =0.4

    Courant-Friedrichs-Lewy number.
- real, parameter eta =0.01

    artificial viscosity
- logical, parameter iwarm =.false.

    Warm start flag, if true restarts the code from previous output.
- integer itprint0 =135

    number of output to do warm start
- integer, parameter neq =neqdyn + npas

    number of equations
- integer, parameter nx =nxtot/mpicol

    number of physical cells in x in each MPI block
- integer, parameter ny =nytot/mpirow

    number of physical cells in y in each MPI block
- integer, parameter nz =nztot/mpirowz

    number of physical cells in z in each MPI block
- integer, parameter nxmin =1-nghost

    lower bound of hydro arrays in x
- integer, parameter nxmax =nx+nghost

    upper bound of hydro arrays in x
- integer, parameter nymin =1-nghost

lower bound of hydro arrays in y

- integer, parameter nymax =ny+nghost

  upper bound of hydro arrays in y
- integer, parameter nzmin =1-nghost

  lower bound of hydro arrays in z
- integer, parameter nzmax =nz+nghost

  upper bound of hydro arrays in z
- integer, parameter master =0

  rank of master of MPI processes
- integer, parameter mpi_real_kind =mpi_real8

  MPI double precision.

4.21.1 Detailed Description

This module contains parameters of the run, some of this can be moved later to a runtime input file

4.21.2 Variable Documentation

4.21.2.1 integer parameter parameters::mpi_real_kind =mpi_real8

MPI single precision.

Definition at line 136 of file parameters.f90.

4.22 sources Module Reference

Adds source terms.

Functions/Subroutines

- subroutine getpos (i, j, k, x, y, z, r)

  Gets position in the grid.
- subroutine grav_source (xc, yc, zc, pp, s)

  Gravity due to point sources.
- subroutine radpress_source (i, j, k, xc, yc, zc, rc, pp, s)

  Radiation pressure force.
- subroutine divergence_b (i, j, k, d)

  Computes div(B)
- subroutine divbcorr_source (i, j, k, pp, s)

  8 Wave source terms for div(B) correction
- subroutine source (i, j, k, prim, s)

  Upper level wrapper for sources.

4.22.1 Detailed Description

This module adds the source terms from gravity, radiation pressure (not fully tested), and div(B) cleaning if the 8 wave scheme is used

4.22.2 Function/Subroutine Documentation

4.22.2.1 subroutine sources::divbcorr_source (integer *i*, integer *j*, integer *k*, real, dimension(neq), intent(in) *pp*, real, dimension(neq), intent(inout) *s*)

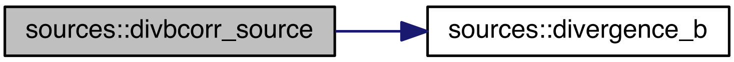
Adds terms proportional to div B in Faraday's Law, momentum equationand energy equation as propoes in Powell et al. 1999

Parameters

integer	[in] i : cell index in the X direction
integer	[in] j : cell index in the Y direction
integer	[in] k : cell index in the Z direction
real	[in] pp(neq) : vector of primitive variables
real	[out] s(neq) : vector with source terms

Definition at line 199 of file sources.f90.

Here is the call graph for this function:



4.22.2.2 subroutine sources::divergence_b (integer, intent(in) *i*, integer, intent(in) *j*, integer, intent(in) *k*, real, intent(out) *d*)

Computes div(B)

Parameters

integer	[in] i : cell index in the X direction
integer	[in] j : cell index in the Y direction
integer	[in] k : cell index in the Z direction
real	[out] d :: div(B)

Definition at line 176 of file sources.f90.

4.22.2.3 subroutine sources::getpos (integer, intent(in) *i*, integer, intent(in) *j*, integer, intent(in) *k*, real, intent(out) *x*, real, intent(out) *y*, real, intent(out) *z*, real, intent(out) *r*)

Gets the position and spherical radius calculated with respect to the center of the grid

Parameters

integer	[in] i : index in the X direction
integer	[in] j : index in the Y direction
integer	[in] k : index in the Z direction
real	[out] x : X position form the center of the grid (code units)
real	[out] y : Y position form the center of the grid (code units)
real	[out] z : Z position form the center of the grid (code units)
real	[out] r : Spherical radius form the center of the grid (code units)

Definition at line 54 of file sources.f90.

4.22.2.4 subroutine sources::grav_source (real, intent(in) *xc*, real, intent(in) *yc*, real, intent(in) *zc*, real, dimension(neq), intent(in) *pp*, real, dimension(neq), intent(inout) *s*)

Adds the gravitational force due to point particles, at this moment is fixed to two point sources (exoplanet)

Parameters

real	[in] xc : X position of the cell
real	[in] yc : Y position of the cell
real	[in] zc : Z position of the cell
real	[in] pp(neq) : vector of primitive variables
real	[out] s(neq) : vector with source terms

Definition at line 81 of file sources.f90.

4.22.2.5 subroutine sources::radpress_source (integer, intent(in) *i*, integer, intent(in) *j*, integer, intent(in) *k*, real, intent(in) *xc*, real, intent(in) *yc*, real, intent(in) *zc*, real, intent(in) *rc*, real, dimension(neq), intent(in) *pp*, real, dimension(neq), intent(inout) *s*)

Adds the radiaiton pressure force due to photo-ionization

Parameters

integer	[in] i : cell index in the X direction
integer	[in] j : cell index in the Y direction
integer	[in] k : cell index in the Z direction
real	[in] xc : X position of the cell
real	[in] yc : Y position of the cell
real	[in] zc : Z position of the cell
reak	[in] rc : $\sqrt{x^2+y^2+z^2}$
real	[in] pp(neq) : vector of primitive variables
real	[out] s(neq) : vector with source terms

Definition at line 138 of file sources.f90.

4.22.2.6 subroutine sources::source (integer, intent(in) *i*, integer, intent(in) *j*, integer, intent(in) *k*, real, dimension(neq), intent(in) *prim*, real, dimension(neq), intent(out) *s*)

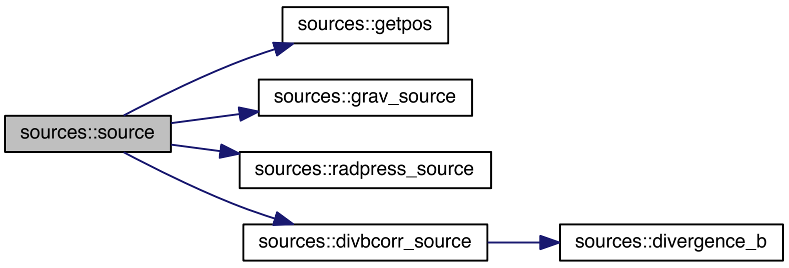
Upper level wrapper for sources
Main driver, this is called from the upwind stepping

Parameters

integer	[in] i : cell index in the X direction
integer	[in] j : cell index in the Y direction
integer	[in] k : cell index in the Z direction
real	[in] prim(neq) : vector of primitive variables
real	[out] s(neq) : vector with source terms'

Definition at line 238 of file sources.f90.

Here is the call graph for this function:



4.23 thermal_cond Module Reference

Adds (isotropuic) thermal conducion.

Functions/Subroutines

- subroutine init_thermal_cond ()

  Intializes Temperature array.
- subroutine dt_cond (dt)

  computes conduction timescale
- subroutine progress (j, tot)

  Progress bar.
- real function ksp (T)

  Spitzer conductivity.
- subroutine heatfluxes ()

  Returns Heat Fluxes.
- subroutine thermal_bounds ()

  Exchanges ghost cells for energy only.
- real function superstep (N, snu)

  Length of superstep.
- real function substep (j, N, nu)

  Size of substep j.
- subroutine st_steps (fs, Ns, fstep)

  Returns the number of Supersteps.
- subroutine thermal_conduction (dt)

  Upper level wrapper for thermal conduction.

Variables

- real, parameter ph =0.4

  Parameter for the sturated regime in McKee.
- real, parameter nu =0.005

  Super-stepping damping factor.
- real, parameter snu =sqrt(nu)

Sqrt of damping factor.

- real, dimension(:,:,:), allocatable temp

  Temperature array [K].

- real dtcond

  conduction timestep

4.23.1 Detailed Description

Adds thermal conduction term, affects both the primitive and conserved variables

4.23.2 Function/Subroutine Documentation

4.23.2.1 subroutine thermal_cond::dt_cond (real, intent(out) *dt*)

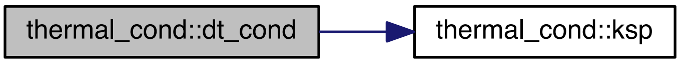
computes conduction timescale

Parameters

real	[out] dt :: conduction timescale

Definition at line 62 of file thermal_cond.f90.

Here is the call graph for this function:

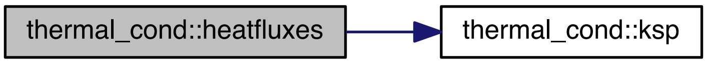


4.23.2.2 subroutine thermal_cond::heatfluxes ( )

Heat flux, takes minimum of spitzer and saturated value
The result is stored in the 5th component of global the F,G,H fluxes (in cgs, conversion is done in dt product)

Definition at line 143 of file thermal_cond.f90.

Here is the call graph for this function:



4.23.2.3 subroutine thermal_cond::init_thermal_cond ( )

Intializes Temperature array

Definition at line 49 of file thermal_cond.f90.

4.23.2.4 real function thermal_cond::ksp (real, intent(in) *T*)

Computes the Spitzer conductivity

Parameters

real	[in] T : temperature [K]

Definition at line 127 of file thermal_cond.f90.

4.23.2.5 subroutine thermal_cond::progress (integer(kind=4) *j*, integer(kind=4), intent(in) *tot*)

Progress bar (only tested with Fortran conmpiler) takes a number between 1 and tot

Parameters

integer	[in] j : current iteration
integer	[in] tot : total number of iterartions

Definition at line 105 of file thermal_cond.f90.

4.23.2.6 subroutine thermal_cond::st_steps (real, intent(in) *fs*, integer, intent(out) *Ns*, real, intent(out) *fstep*)

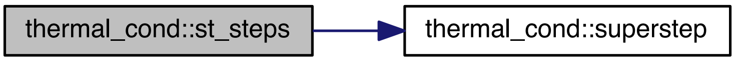
Returns the number of Supersteps

Parameters

real	fs : ratio of dtcond/dthydro
integer	Ns : Number of Supersteps
real	fstep : Number of supersteps (float)

Definition at line 382 of file thermal_cond.f90.

Here is the call graph for this function:



4.23.2.7 real function thermal_cond::substep (integer, intent(in) *j*, integer, intent(in) *N*, real, intent(in) *nu*)

Returns the size of substep j of N

Parameters

integer	[in] j : index of current step
integer	[in] N : Total number of substeps
real	[in] nu : damping factor

Definition at line 364 of file thermal_cond.f90.

4.23.2.8 real function thermal_cond::superstep (integer *N*, real, intent(in) *snu*)

Returns the length of the superstep with N inner substeps

Parameters

integer	[in] N : Nunber of inner substeps
real	[in] snu : sqrt of damping factor

Definition at line 343 of file thermal_cond.f90.

4.23.2.9 subroutine thermal_cond::thermal_bounds ( )

Exchanges one layer of boundaries, only the equation that corresponds to the energy

Definition at line 216 of file thermal_cond.f90.

4.23.2.10 subroutine thermal_cond::thermal_conduction (real, intent(in) *dt*)

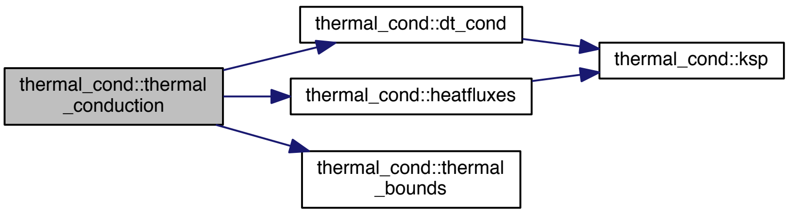
This routine adds the heat conduction, receives the hydro timestep in seconds, and assumes the primitives and Temp(i,j,k) arrays are updated

Parameters

real	[in] dt : Hysdrodynamical timestep (in seconds)

Definition at line 408 of file thermal_cond.f90.

Here is the call graph for this function:


thermal_cond::thermal _conduction
thermal_cond::dt_cond
thermal_cond::heatfluxes
thermal_cond::thermal _bounds
thermal_cond::ksp

4.24 user_mod Module Reference

User imput module.

Functions/Subroutines

- subroutine init_user_mod ()

Initializes variables in the module, as well as other modules loaded by user.
It has to be present, even if empty.

- subroutine initial_conditions (u, time)

  Here the domain is initialized at t=0.

- subroutine impose_user_bc (u, time)

  User Defined Boundary conditions.

4.24.1 Detailed Description

This is an attempt to have all input neede from user in a single file This module should load additional modules (i.e. star, jet, sn), to impose initial and boundary conditions (such as sources)

4.24.2 Function/Subroutine Documentation

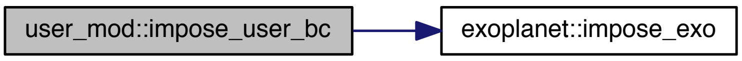
4.24.2.1 subroutine user_mod::impose_user_bc (real, dimension(neq,nxmin:nxmax,nymin:nymax,nzmin:nzmax), intent(out) *u*, real, intent(in) *time*)

Parameters

real	[out] u(neq,nxmin:nxmax,nymin:nymax,nzmin:nzmax) : conserved variables
real	[in] time : time in the simulation (code units)

Definition at line 114 of file user_mod.f90.

Here is the call graph for this function:



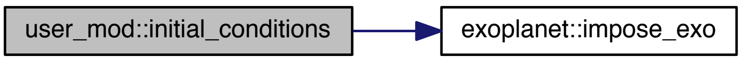
4.24.2.2 subroutine user_mod::initial_conditions (real, dimension(neq,nxmin:nxmax,nymin:nymax,nzmin:nzmax), intent(out) *u*, real, intent(in) *time*)

Parameters

real	[out] u(neq,nxmin:nxmax,nymin:nymax,nzmin:nzmax) : conserved variables
real	[in] time : time in the simulation (code units)

Definition at line 58 of file user_mod.f90.

Here is the call graph for this function:



Chapter 5

File Documentation

5.1 doc/mainpage.h File Reference

Webpage frontend.

5.2 src/boundaries.f90 File Reference

Boundary conditions.

Modules

- module boundaries

  Boundary conditions.

Functions/Subroutines

- subroutine boundaries::boundaryi (time, dt)

  Boundary conditions for 1st order half timestep.
- subroutine boundaries::boundaryii (time, dt)

  Boundary conditions for 2nd order half timestep.

5.2.1 Detailed Description

Author

Alejandro Esquivel

Date

2/Nov/2014

5.3 src/coldens.f90 File Reference

Column density projection.

Modules

- module coldens_utilities

  Column densirt projection.

Functions/Subroutines

- subroutine coldens_utilities::init_coldens ()

  Initializes data.
- subroutine coldens_utilities::read_data (u, itprint, filepath)

  reads data from file
- subroutine coldens_utilities::getxyz (i, j, k, x, y, z)

  gets position of a cell
- subroutine coldens_utilities::rotation_x (theta, x, y, z, xn, yn, zn)

  Rotation around the X axis.
- subroutine coldens_utilities::rotation_y (theta, x, y, z, xn, yn, zn)

  Rotation around the Y axis.
- subroutine coldens_utilities::rotation_z (theta, x, y, z, xn, yn, zn)

  Rotation around the Z axis.
- subroutine coldens_utilities::fill_map (nxmap, nymap, u, map, dxT, dyT, theta_x, theta_y, theta_z)

  Fill target map.
- subroutine coldens_utilities::write_map (fileout, nxmap, nymap, map)

  Writes projection to file.
- program coldens

  Computes the H-alpha emission.

5.3.1 Detailed Description

Author

Alejandro Esquivel

Date

2/Nov/2014

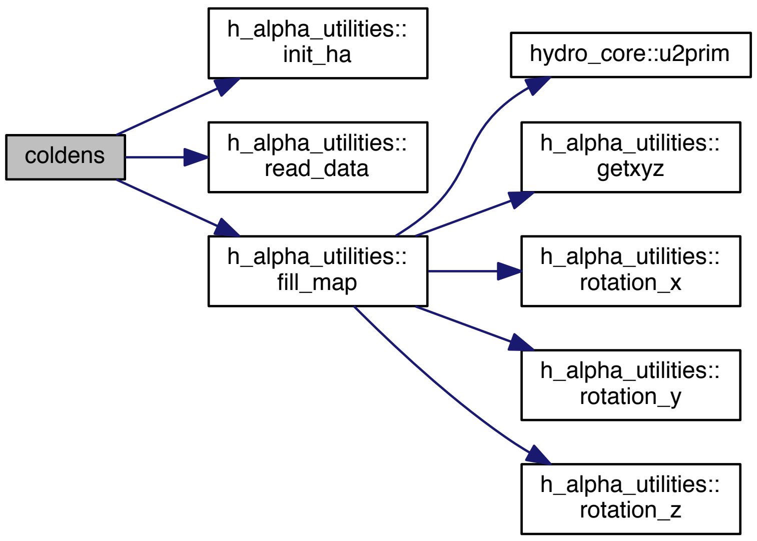
5.3.2 Function/Subroutine Documentation

5.3.2.1 program coldens ( )

Computes the H-alpha apbsorption
It rotates the data along each of the coordinates axis by an amount $\theta_x, \theta_y, \theta_z$, and projectcs the map along the the LOS, which is taken to be the Z axis

Definition at line 370 of file coldens.f90.

Here is the call graph for this function:



5.4 src/constants.f90 File Reference

Constants module.

Modules

- module constants

  Module containing physical and asronomical constants.

Variables

- real, parameter constants::pi =acos(-1.)

  π
- real, parameter constants::amh =1.66e-24

  hydrogen mass
- real, parameter constants::mu =1.3

  mean atomic mass
- real, parameter constants::kb =1.38e-16

  Boltzmann constant (cgs)
- real, parameter constants::rg =8.3145e7

  Gas constant (cgs)
- real, parameter constants::ggrav =6.67259e-8

  Gravitational constant (cgs)
- real, parameter constants::clight =2.99E10

  speed of light in vacuum (cgs)
- real, parameter constants::msun =1.99E33

solar radius (cgs)
- real, parameter constants::rsun =6.955e10

  solar mass (cgs)
- real, parameter constants::mjup =1.898E30

  Jupiter mass (cgs)
- real, parameter constants::rjup =7.1492E9

  Jupiter radius (cgs)
- real, parameter constants::au =1.496e13

  1AU in cm
- real, parameter constants::pc =3.0857E18

  1pc in cm
- real, parameter constants::kpc =3.0857E21

  1Kpc in cm
- real, parameter constants::hr =3600.

  1hr in seconds
- real, parameter constants::day =86400.

  1day in seconds
- real, parameter constants::yr =3.1536E7

  1yr in seconds
- real, parameter constants::myr =3.1536E13

  1Myr in seconds

5.4.1 Detailed Description

Author

Alejandro Esquivel

Date

2/Nov/2014

5.5 src/cooling_chi.f90 File Reference

Cooling module with CHIANTI generated cooling curves.

Modules

- module cooling_chi

  Cooling module with CHIANTI generated cooling curves.

Functions/Subroutines

- subroutine cooling_chi::read_table ()

  Reads the cooling curve table.
- real(kind=8) function cooling_chi::coolchi (T)

  Returns the cooling coefficient interpolating the table.
- subroutine cooling_chi::coolingchi (dt)

  High level wrapper to apply cooling with CHIANTI tables.

Variables

- real(kind=8), dimension(2, 41) **cooling_chi::cooltab**

5.5.1 Detailed Description

Author

Alejandro Esquivel

Date

2/Nov/2014

5.6 src/cooling_dmc.f90 File Reference

Cooling module with Dlgarno Mac Cray coronal cooling curve.

Modules

- module cooling_dmc

  Cooling module with Dalgarno McCray coronal cooling curve.

Functions/Subroutines

- subroutine cooling_dmc::read_table ()

  Reads the cooling curve table.
- real(kind=8) function cooling_dmc::cooldmc (T)

  Returns the cooling coefficient interpolating the table.
- subroutine cooling_dmc::coolingdmc (dt)

  High level wrapper to apply cooling with DMC table.

Variables

- real(kind=8), dimension(2, 41) **cooling_dmc::cooltab**

5.6.1 Detailed Description

Author

Alejandro Esquivel

Date

2/Nov/2014

5.7 src/cooling_h.f90 File Reference

Cooling with hydrogen rate parametrized cooling.

Modules

- module cooling_h

  Cooling with parametrized cooling and H rate equation.

Functions/Subroutines

- subroutine cooling_h::coolingh (dt)

  High level wrapper to apply cooling.
- real(kind=8) function cooling_h::alpha (T)

  calculates the recombination rate (case B)
- real(kind=8) function cooling_h::alpha1 (T)

  calculates the recombination rate to level 1
- real(kind=8) function cooling_h::colf (T)

  calculates the collisional ionization rate
- real(kind=8) function cooling_h::betah (T)

  betaH(T)
- real(kind=8) function cooling_h::aloss (X1, X2, DT, DEN, DH0, TE0)

  Non equilibrium cooling.
- subroutine cooling_h::atomic (dt, uu, tau, radphi)

  Updates the ionization fraction and applpies cooling.

5.7.1 Detailed Description

Author

Alejandro Esquivel

Date

2/Nov/2014

5.8 src/difrad.f90 File Reference

Diffuse radiation module.

Modules

- module difrad

  Ray tracing Radiative Trasnport.

Functions/Subroutines

- subroutine difrad::init_rand ()

  initializes random number generation
- subroutine difrad::emdiff (emax)

  calculates the diffuse fotoionization emissivity
- subroutine difrad::random_versor (xd, yd, zd)

  returns the 3 components of a random versor
- subroutine difrad::starsource (srad, x0, y0, z0, x, y, z, xd, yd, zd)

Place photon packets at a "star" surface.
- subroutine difrad::photons (xl0, yl0, zl0, xd, yd, zd, f)

  Photon trajectories.
- subroutine difrad::radbounds ()

  follows the rays across MPI boundaries
- subroutine difrad::progress (j, tot)

  Progress bar.
- subroutine difrad::diffuse_rad ()

  Diffuse radiation driver.

Variables

- real, parameter difrad::a0 =6.3e-18

  Fotoionization cross section.
- integer, parameter difrad::nrays =1000000

  Number of rays.
- real, dimension(:,:,:), allocatable difrad::ph

  Photoionizing rate.
- real, dimension(:,:,:), allocatable difrad::em

  Photoionizing emissivity.
- real, dimension(:,:,:), allocatable difrad::photl

  Auxiliary buffer for MPI.
- real, dimension(:,:,:), allocatable difrad::photr

  Auxiliary buffer for MPI.
- real, dimension(:,:,:), allocatable difrad::photb

  Auxiliary buffer for MPI.
- real, dimension(:,:,:), allocatable difrad::phott

  Auxiliary buffer for MPI.
- real, dimension(:,:,:), allocatable difrad::photo

  Auxiliary buffer for MPI.
- real, dimension(:,:,:), allocatable difrad::photi

  Auxiliary buffer for MPI.
- integer, dimension(6) difrad::buffersize

  Auxiliary buffer for MPI.

5.8.1 Detailed Description

Author

Alejandro Esquivel

Date

2/Nov/2014

5.9 src/exoplanet.f90 File Reference

Exoplanet problem module.

Modules

- module exoplanet

  Exoplanet module.

Functions/Subroutines

- subroutine exoplanet::init_exo ()

  Module initialization.
- subroutine exoplanet::impose_exo (u, time)

  Inject sources of wind.

Variables

- real exoplanet::rsw

  Stellar radius.
- real exoplanet::tsw

  Stellar wind temperature.
- real exoplanet::vsw

  Stellar wind velocity.
- real exoplanet::dsw

  Stellar Wind Density.
- real exoplanet::b0

  Magnetic Field.
- real exoplanet::rpw

  Planetary radius.
- real exoplanet::tpw

  Planetary wind temperature.
- real exoplanet::vpw

  Planetary wind velocity.
- real **exoplanet::dpw**
- real exoplanet::torb

  planet: orbital period
- real exoplanet::rorb

  orbital radius
- real exoplanet::masss

  Mass of the Star.
- real exoplanet::massp

  Mass of the Planet.
- real exoplanet::xp

  X position of the planet.
- real exoplanet::yp

  Y position of the planet.
- real exoplanet::zp

  Z position of the planet.

5.9.1 Detailed Description

Author

M. Schneiter, C. Villarreal D'Angelo, A. Esquivel

Date

2/Nov/2014

5.10 src/globals.f90 File Reference

Global variables.

Modules

- module globals

  Module containing global variables.

Variables

- real, dimension(:,:,:,:), allocatable globals::u

  conserved varibles
- real, dimension(:,:,:,:), allocatable globals::up

  conserved varibles after 1/2 timestep
- real, dimension(:,:,:,:), allocatable globals::primit

  primitive varibles
- real, dimension(:,:,:,:), allocatable globals::f

  X fluxes.
- real, dimension(:,:,:,:), allocatable globals::g

  Y fluxes.
- real, dimension(:,:,:,:), allocatable globals::h

  Z fluxes.
- real globals::dx

  grid spacing in X
- real globals::dy

  grid spacing in Y
- real globals::dz

  grid spacing in Z
- integer, dimension(0:2) globals::coords

  position of neighboring MPI blocks
- integer globals::left

  MPI neighbor in the -x direction.
- integer globals::right

  MPI neighbor in the +x direction.
- integer globals::top

  MPI neighbor in the -y direction.
- integer globals::bottom

  MPI neighbor in the +y direction.
- integer globals::out

  MPI neighbor in the -z direction.

- integer globals::in

  MPI neighbor in the +z direction.
- integer globals::rank

  MPI rank.
- integer globals::comm3d

  Cartesian MPI comunicator.

5.10.1 Detailed Description

Author

Alejandro Esquivel

Date

2/Nov/2014

5.11 src/h_alpha_proj.f90 File Reference

H alpha projection.

Modules

- module h_alpha_utilities

  H alpha projection.

Functions/Subroutines

- subroutine h_alpha_utilities::init_ha ()

  Initializes data.
- subroutine h_alpha_utilities::read_data (u, itprint, filepath)

  reads data from file
- subroutine h_alpha_utilities::getxyz (i, j, k, x, y, z)

  gets position of a cell
- subroutine h_alpha_utilities::rotation_x (theta, x, y, z, xn, yn, zn)

  Rotation around the X axis.
- subroutine h_alpha_utilities::rotation_y (theta, x, y, z, xn, yn, zn)

  Rotation around the Y axis.
- subroutine h_alpha_utilities::rotation_z (theta, x, y, z, xn, yn, zn)

  Rotation around the Z axis.
- subroutine h_alpha_utilities::fill_map (nxmap, nymap, u, map, dxT, dyT, theta_x, theta_y, theta_z)

  Fill target map.
- subroutine h_alpha_utilities::write_ha (fileout, nxmap, nymap, map)

  Writes projection to file.
- subroutine h_alpha_utilities::write_rg (fileout, nxmap, nymap, map)

  Writes projection to file in rg format.
- program h_alpha_proj

  Computes the H-alpha emission.

5.11.1 Detailed Description

Author

Alejandro Esquivel

Date

2/Nov/2014

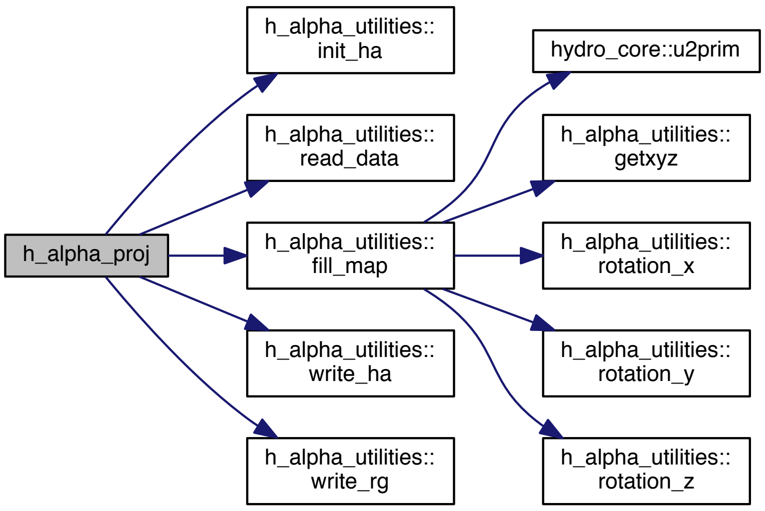
5.11.2 Function/Subroutine Documentation

5.11.2.1 program h_alpha_proj ( )

Computes the H-alpha apbsorption
It rotates the data along each of the coordinates axis by an amount $\theta_x, \theta_y, \theta_z$, and projectcs the map along the the LOS, which is taken to be the Z axis

Definition at line 428 of file h_alpha_proj.f90.

Here is the call graph for this function:


h_alpha_proj
h_alpha_utilities:: init_ha
h_alpha_utilities:: read_data
h_alpha_utilities:: fill_map
h_alpha_utilities:: write_ha
h_alpha_utilities:: write_rg
hydro_core::u2prim
h_alpha_utilities:: getxyz
h_alpha_utilities:: rotation_x
h_alpha_utilities:: rotation_y
h_alpha_utilities:: rotation_z

5.12 src/hll.f90 File Reference

HLL approximate Riemann solver module.

Modules

- module hll

  HLL approximate Riemann solver module.

Functions/Subroutines

- subroutine hll::prim2fhll (priml, primr, ff)

  Solves the Riemann problem at the interface PL,PR using the HLL solver.
- subroutine hll::hllfluxes (choice)

  Calculates HLL fluxes from the primitive variables on all the domain.

5.12.1 Detailed Description

Author

Alejandro Esquivel

Date

2/Nov/2014

5.13 src/hllc.f90 File Reference

HLLC approximate Riemann solver module.

Modules

- module hllc

  HLLC approximate Riemann solver module.

Functions/Subroutines

- subroutine hllc::prim2fhllc (priml, primr, ff)

  Solves the Riemann problem at the interface PL,PR using the HLLC solver.
- subroutine hllc::hllcfluxes (choice)

  Calculates HLLC fluxes from the primitive variables on all the domain.

5.13.1 Detailed Description

Author

Alejandro Esquivel

Date

2/Nov/2014

5.14 src/hlld.f90 File Reference

HLLD approximate Riemann solver module.

Modules

- module hlld

  HLLD approximate Riemann solver module.

Functions/Subroutines

- subroutine hlld::prim2fhlld (priml, primr, ff)

    Solves the Riemann problem at the interface PL,PR using the HLLD solver.
- subroutine hlld::hlldfluxes (choice)

    Calculates HLLD fluxes from the primitive variables on all the domain.

5.14.1 Detailed Description

Author

C. Villarreal D'Angelo, A. Esquivel, M. Schneiter

Date

2/Nov/2014

5.15 src/hlle.f90 File Reference

HLLE approximate Riemann solver module.

Modules

- module hlle

    HLLE approximate Riemann solver module.

Functions/Subroutines

- subroutine hlle::prim2fhlle (priml, primr, ff)

    Solves the Riemann problem at the interface PL,PR using the HLLE solver.
- subroutine hlle::hllefluxes (choice)

    Calculates HLLE fluxes from the primitive variables on all the domain.

5.15.1 Detailed Description

Author

C. Villarreal D'Angelo, A. Esquivel, M. Schneiter

Date

2/Nov/2014

5.16 src/hydro_core.f90 File Reference

Hydrodynamical and Magnetohidrodynamocal bacic module.

Modules

- module hydro_core

    Basic hydro (and MHD) subroutines utilities.

Functions/Subroutines

- subroutine hydro_core::u2prim (uu, prim, T)

  Computes the primitive variables and temperature from conserved variables on a single cell.
- subroutine hydro_core::calcprim (u, primit)

  Updated the primitives, using the conserved variables in the entire domain.
- subroutine hydro_core::prim2u (prim, uu)

  Computes the conserved conserved variables from the primitives in a single cell.
- subroutine hydro_core::prim2f (prim, ff)

  Computes the Euler Fluxes in one cell.
- subroutine hydro_core::swapy (var, neq)

  Swaps the x and y components in a cell.
- subroutine hydro_core::swapz (var, neq)

  Swaps the x and z components in a cell.
- subroutine hydro_core::csound (p, d, cs)

  Computes the sound speed.
- subroutine hydro_core::cfast (p, d, bx, by, bz, cfx, cfy, cfz)

  Computes the fast magnetosonic speeds in the 3 coordinates.
- subroutine hydro_core::cfastx (prim, cfX)

  Computes the fast magnetosonic speed in the x direction.
- subroutine hydro_core::get_timestep (dt)

  Otains the timestep allowed by the CFL condition in the entire.
- subroutine hydro_core::limiter (PLL, PL, PR, PRR, neq)

  Performs a linear reconstruction of the primitive variables.
- real function **average** (a, b)

5.16.1 Detailed Description

Author

Alejandro Esquivel

Date

2/Nov/2014

5.17 src/hydro_solver.f90 File Reference

Hydrodynamical and Magnetohidrodynamocal solver module.

Modules

- module hydro_solver

  Advances the simulation one timestep.

Functions/Subroutines

- subroutine hydro_solver::viscosity ()

  Adds artificial viscosity to the conserved variables.
- subroutine hydro_solver::step (dt)

  Upwind timestep.
- subroutine hydro_solver::tstep (time, dt)

  High level wrapper to advancce the simulation.

5.17.1 Detailed Description

Author

Alejandro Esquivel

Date

2/Nov/2014

5.18 src/init.f90 File Reference

Guacho-3D initialization module.

Modules

- module init

  Guacho-3D initialization.

Functions/Subroutines

- subroutine init::initmain (time, tprint, itprint)

  Main initialization routine.
- subroutine init::initflow (itprint)

  Initializes the conserved variables, in the globals module.

5.18.1 Detailed Description

Author

Alejandro Esquivel

Date

2/Nov/2014

5.19 src/lyman_alpha_tau.f90 File Reference

Lyman_alpha_utilities.

Modules

- module lyman_alpha_utilities

  Lyman_alpha_utilities.

Functions/Subroutines

- subroutine lyman_alpha_utilities::init_la ()

  Initializes data.
- subroutine lyman_alpha_utilities::read_data (u, itprint, filepath)

reads data from file

- subroutine lyman_alpha_utilities::getxyz (i, j, k, x, y, z)

  gets position of a cell

- subroutine lyman_alpha_utilities::rotation_x (theta, x, y, z, xn, yn, zn)

  Rotation around the X axis.

- subroutine lyman_alpha_utilities::rotation_y (theta, x, y, z, xn, yn, zn)

  Rotation around the Y axis.

- subroutine lyman_alpha_utilities::rotation_z (theta, x, y, z, xn, yn, zn)

  Rotation around the Z axis.

- subroutine lyman_alpha_utilities::fill_map (nxmap, nymap, nvmap, vmin, vmax, u, map, dxT, dyT, theta_x, theta_y, theta_z)

  Fill target map.

- subroutine lyman_alpha_utilities::write_la (itprint, filepath, nxmap, nymap, nvmap, map)

  Writes projection to file.

- subroutine lyman_alpha_utilities::phigauss (T, vzn, vmin, vmax, nvmap, profile)

  This routine computes a gaussian line profile.

- program lyman_alpha_tau

  Computes the Ly-alpha apbsorption.

5.19.1 Detailed Description

Author

Alejandro Esquivel

Date

2/Nov/2014

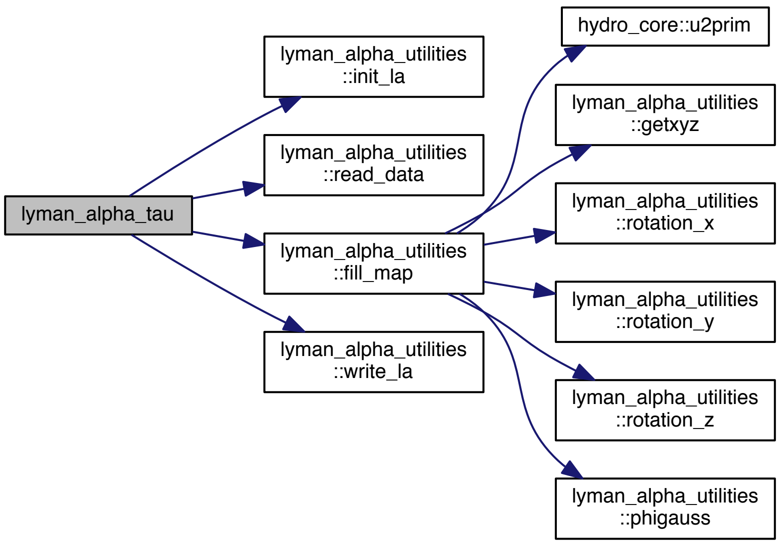
5.19.2 Function/Subroutine Documentation

5.19.2.1 program lyman_alpha_tau ( )

Computes the Ly-alpha apbsorption
It rotates the data along each of the coordinates axis by an amount $\theta_x, \theta_y, \theta_z$, and the LOS is along the Z axis

Definition at line 418 of file lyman_alpha_tau.f90.

Here is the call graph for this function:


lyman_alpha_tau
lyman_alpha_utilities ::init_la
lyman_alpha_utilities ::read_data
lyman_alpha_utilities ::fill_map
lyman_alpha_utilities ::write_la
hydro_core::u2prim
lyman_alpha_utilities ::getxyz
lyman_alpha_utilities ::rotation_x
lyman_alpha_utilities ::rotation_y
lyman_alpha_utilities ::rotation_z
lyman_alpha_utilities ::phigauss

5.20 src/main.f90 File Reference

Guacho-3D main program.

Functions/Subroutines

- program guacho

  Guacho-3D Main Program This is the main program unit of the Guacho-3D code.
  The code itegrates Euler equations in three dimensions, the choice of the integration method is set in the makefile.
  The flow (conserved) variables are taken to be:
  ieq=
  1 : rho (total)
  2 : rho u
  3 : rho v
  4 : rho w
  5 : Internal energy (thermal+kinetic)
  6 : bx (optional, if MHD or PMHD)
  7 : by (optional, if MHD or PMHD)

8 : bz (optional, if MHD or PMHD)
additional variables advected into the flow, e.g.:
9 (6): n_HI
10 (7): n_HII
11 (8): n_HeI
12 (9): n_HeII
13 (10): n_HeIII
14 (11): rho∗zbar
15 (12): ne
This can be changed bu the user according to cooling function for instance.

5.20.1 Detailed Description

Author

Alejandro Esquivel

Date

2/Nov/2014

5.21 src/Out_Silo_Module.f90 File Reference

Output in Silo Format.

Modules

- module out_silo_module

  Output in Silo (+HDF5) Format.

Functions/Subroutines

- subroutine out_silo_module::writeblocks (itprint)

  Writes Data, one file per processor.
- subroutine out_silo_module::writemaster (itprint)

  Writes the Master File.
- subroutine out_silo_module::outputsilo (itprint)

  Upper level wrapper.

5.21.1 Detailed Description

Author

Alejandro Esquivel

Date

2/Nov/2014

5.22 src/output.f90 File Reference

Writes Output.

Modules

- module output

  Writes output.

Functions/Subroutines

- subroutine output::write_output (itprint)

  Writes output.

5.22.1 Detailed Description

Author

Alejandro Esquivel

Date

2/Nov/2014

5.23 src/parameters.f90 File Reference

parameters module

Modules

- module parameters

  Parameters module.

Variables

- character(len=128), parameter parameters::outputpath ='/datos/esquivel/EXO-GUACHO/P5/'

  Path used to write the output.
- character(len=128), parameter parameters::workdir ='./'

  working directory
- integer, parameter parameters::neqdyn =8

  num. of eqs (+scal)
- integer, parameter parameters::ndim =3

  num. of dimensions
- integer, parameter parameters::npas =2

  num. of passive scalars
- integer, parameter parameters::nghost =2

  num. of ghost cells
- integer, parameter parameters::nxtot =600

  Total grid size in X.
- integer, parameter parameters::nytot =150

  Total grid size in Y.
- integer, parameter parameters::nztot =600

  Total grid size in Z.
- integer, parameter parameters::mpicol =4

number of MPI blocks in X

- integer, parameter parameters::mpirow =2

  number of MPI blocks in Y
- integer, parameter parameters::mpirowz =4

  number of MPI blocks in Z
- integer, parameter parameters::np =mpicol∗mpirow∗mpirowz

  total number of MPI processes
- real, parameter parameters::xmax =1.

  grid extent in X (code units)
- real, parameter parameters::ymax =0.25

  grid extent in Y (code units)
- real, parameter parameters::zmax =1.

  grid extent in Z (code units)
- real, parameter parameters::xphys =0.30∗AU

  grid extent in X (pohysical units, cgs)
- real, parameter parameters::cv =1.5

  Specific heat at constant volume (/R)
- real, parameter parameters::gamma =(cv+1.)/cv

  Cp/Cv.
- real, parameter parameters::t0 =1.e4

  reference temperature (to set cs)
- real, parameter parameters::rsc =xphys/xmax

  distance scaling
- real, parameter parameters::rhosc =amh∗mu

  mass density scaling
- real, parameter parameters::tempsc =T0∗gamma

  Temperature scaling.
- real, parameter parameters::vsc2 = gamma∗Rg∗T0/mu

  Velocity scaling.
- real, parameter parameters::psc = rhosc∗vsc2

  Pressure scaling.
- real, parameter parameters::tsc =rsc/sqrt(vsc2)

  time scaling
- real, parameter parameters::bsc = sqrt(4.0∗pi∗Psc)

  magnetic fiewld scaling
- real, parameter parameters::tmax = 3.8∗day/tsc

  Maximum integration time.
- real, parameter parameters::dtprint = 0.025∗day/tsc

  interval between consecutive outputs
- real, parameter parameters::cfl =0.4

  Courant-Friedrichs-Lewy number.
- real, parameter parameters::eta =0.01

  artificial viscosity
- logical, parameter parameters::iwarm =.false.

  Warm start flag, if true restarts the code from previous output.
- integer parameters::itprint0 =135

  number of output to do warm start
- integer, parameter parameters::neq =neqdyn + npas

  number of equations
- integer, parameter parameters::nx =nxtot/mpicol

  number of physical cells in x in each MPI block

- integer, parameter parameters::ny =nytot/mpirow

  number of physical cells in y in each MPI block
- integer, parameter parameters::nz =nztot/mpirowz

  number of physical cells in z in each MPI block
- integer, parameter parameters::nxmin =1-nghost

  lower bound of hydro arrays in x
- integer, parameter parameters::nxmax =nx+nghost

  upper bound of hydro arrays in x
- integer, parameter parameters::nymin =1-nghost

  lower bound of hydro arrays in y
- integer, parameter parameters::nymax =ny+nghost

  upper bound of hydro arrays in y
- integer, parameter parameters::nzmin =1-nghost

  lower bound of hydro arrays in z
- integer, parameter parameters::nzmax =nz+nghost

  upper bound of hydro arrays in z
- integer, parameter parameters::master =0

  rank of master of MPI processes
- integer, parameter parameters::mpi_real_kind =mpi_real8

  MPI double precision.

5.23.1 Detailed Description

Author

Alejandro Esquivel

Date

2/Nov/2014

5.24 src/sources.f90 File Reference

Adds source terms.

Modules

- module sources

  Adds source terms.

Functions/Subroutines

- subroutine sources::getpos (i, j, k, x, y, z, r)

  Gets position in the grid.
- subroutine sources::grav_source (xc, yc, zc, pp, s)

  Gravity due to point sources.
- subroutine sources::radpress_source (i, j, k, xc, yc, zc, rc, pp, s)

  Radiation pressure force.
- subroutine sources::divergence_b (i, j, k, d)

  Computes div(B)

- subroutine sources::divbcorr_source (i, j, k, pp, s)

  8 Wave source terms for div(B) correction
- subroutine sources::source (i, j, k, prim, s)

  Upper level wrapper for sources.

5.24.1 Detailed Description

Author

Alejandro Esquivel

Date

2/Nov/2014

5.25 src/thermal_cond.f90 File Reference

Thermal conduction module.

Modules

- module thermal_cond

  Adds (isotropuic) thermal conducion.

Functions/Subroutines

- subroutine thermal_cond::init_thermal_cond ()

  Intializes Temperature array.
- subroutine thermal_cond::dt_cond (dt)

  computes conduction timescale
- subroutine thermal_cond::progress (j, tot)

  Progress bar.
- real function thermal_cond::ksp (T)

  Spitzer conductivity.
- subroutine thermal_cond::heatfluxes ()

  Returns Heat Fluxes.
- subroutine thermal_cond::thermal_bounds ()

  Exchanges ghost cells for energy only.
- real function thermal_cond::superstep (N, snu)

  Length of superstep.
- real function thermal_cond::substep (j, N, nu)

  Size of substep j.
- subroutine thermal_cond::st_steps (fs, Ns, fstep)

  Returns the number of Supersteps.
- subroutine thermal_cond::thermal_conduction (dt)

  Upper level wrapper for thermal conduction.

Variables

- real, parameter thermal_cond::ph =0.4

  Parameter for the sturated regime in McKee.
- real, parameter thermal_cond::nu =0.005

  Super-stepping damping factor.
- real, parameter thermal_cond::snu =sqrt(nu)

  Sqrt of damping factor.
- real, dimension(:,:,:), allocatable thermal_cond::temp

  Temperature array [K].
- real thermal_cond::dtcond

  conduction timestep

5.25.1 Detailed Description

Author

Alejandro Esquivel

Date

2/Nov/2014

5.26 src/user_mod.f90 File Reference

User input module.

Modules

- module user_mod

  User imput module.

Functions/Subroutines

- subroutine user_mod::init_user_mod ()

  Initializes variables in the module, as well as other modules loaded by user.
  It has to be present, even if empty.
- subroutine user_mod::initial_conditions (u, time)

  Here the domain is initialized at t=0.
- subroutine user_mod::impose_user_bc (u, time)

  User Defined Boundary conditions.

5.26.1 Detailed Description

Author

Alejandro Esquivel

Date

2/Nov/2014

Index