

Guacho 3D

V1.1

Generated by Doxygen 1.8.9.1

Thu Apr 2 2015 12:50:14

Contents

1	GUACHO-3D Documentation	1
1.1	Introduction	1
1.2	release.notes	1
1.3	requirements	1
2	Modules Index	3
2.1	Modules List	3
3	File Index	5
3.1	File List	5
4	Module Documentation	7
4.1	boundaries Module Reference	7
4.1.1	Detailed Description	7
4.1.2	Function/Subroutine Documentation	7
4.1.2.1	boundaryi	7
4.1.2.2	boundaryii	8
4.2	coldens_utilities Module Reference	9
4.2.1	Detailed Description	9
4.2.2	Function/Subroutine Documentation	9
4.2.2.1	fill_map	9
4.2.2.2	getxyz	10
4.2.2.3	init_coldens	11
4.2.2.4	read_data	11
4.2.2.5	rotation_x	11
4.2.2.6	rotation_y	11
4.2.2.7	rotation_z	12
4.2.2.8	write_map	12
4.3	constants Module Reference	12
4.4	cooling_chi Module Reference	13
4.4.1	Detailed Description	13
4.4.2	Function/Subroutine Documentation	13

4.4.2.1	coolchi	13
4.4.2.2	coolingchi	14
4.4.2.3	read_table	14
4.5	cooling_dmc Module Reference	14
4.5.1	Detailed Description	15
4.5.2	Function/Subroutine Documentation	15
4.5.2.1	cooldmc	15
4.5.2.2	coolingdmc	15
4.5.2.3	read_table	15
4.6	cooling_h Module Reference	15
4.6.1	Detailed Description	16
4.6.2	Function/Subroutine Documentation	16
4.6.2.1	aloss	16
4.6.2.2	alpha	16
4.6.2.3	alpha1	17
4.6.2.4	atomic	17
4.6.2.5	betah	17
4.6.2.6	colf	18
4.6.2.7	coolingh	18
4.7	difrad Module Reference	18
4.7.1	Detailed Description	19
4.7.2	Function/Subroutine Documentation	19
4.7.2.1	diffuse_rad	19
4.7.2.2	emdiff	20
4.7.2.3	init_rand	20
4.7.2.4	photons	20
4.7.2.5	progress	21
4.7.2.6	radbounds	21
4.7.2.7	random_versor	21
4.7.2.8	starsource	21
4.8	exoplanet Module Reference	22
4.8.1	Detailed Description	23
4.8.2	Function/Subroutine Documentation	23
4.8.2.1	impose_exo	23
4.8.2.2	init_exo	23
4.9	globals Module Reference	23
4.9.1	Detailed Description	24
4.10	h_alpha_utilities Module Reference	24
4.10.1	Detailed Description	25
4.10.2	Function/Subroutine Documentation	25

4.10.2.1	fill_map	25
4.10.2.2	getxyz	26
4.10.2.3	init_ha	27
4.10.2.4	read_data	27
4.10.2.5	rotation_x	27
4.10.2.6	rotation_y	27
4.10.2.7	rotation_z	28
4.10.2.8	write_ha	28
4.10.2.9	write_rg	28
4.11	hll Module Reference	29
4.11.1	Detailed Description	29
4.11.2	Function/Subroutine Documentation	29
4.11.2.1	hllfluxes	29
4.11.2.2	prim2fhll	29
4.12	hllc Module Reference	30
4.12.1	Detailed Description	30
4.12.2	Function/Subroutine Documentation	30
4.12.2.1	hllcfluxes	30
4.12.2.2	prim2fhllc	31
4.13	hlld Module Reference	31
4.13.1	Detailed Description	32
4.13.2	Function/Subroutine Documentation	32
4.13.2.1	hlldfluxes	32
4.13.2.2	prim2fhlld	32
4.14	hlle Module Reference	33
4.14.1	Detailed Description	33
4.14.2	Function/Subroutine Documentation	33
4.14.2.1	hllefluxes	33
4.14.2.2	prim2fhlle	34
4.15	hydro_core Module Reference	34
4.15.1	Detailed Description	35
4.15.2	Function/Subroutine Documentation	35
4.15.2.1	calcprim	35
4.15.2.2	cfast	36
4.15.2.3	cfastx	36
4.15.2.4	csound	36
4.15.2.5	get_timestep	36
4.15.2.6	limiter	37
4.15.2.7	prim2f	37
4.15.2.8	prim2u	37

4.15.2.9	swapy	38
4.15.2.10	swapz	39
4.15.2.11	u2prim	39
4.16	hydro_solver Module Reference	39
4.16.1	Detailed Description	39
4.16.2	Function/Subroutine Documentation	39
4.16.2.1	step	39
4.16.2.2	tstep	41
4.16.2.3	viscosity	42
4.17	init Module Reference	42
4.17.1	Detailed Description	43
4.17.2	Function/Subroutine Documentation	43
4.17.2.1	initflow	43
4.17.2.2	initmain	43
4.18	lyman_alpha_utilities Module Reference	44
4.18.1	Detailed Description	44
4.18.2	Function/Subroutine Documentation	44
4.18.2.1	fill_map	44
4.18.2.2	getxyz	45
4.18.2.3	init_la	45
4.18.2.4	phigauss	46
4.18.2.5	read_data	46
4.18.2.6	rotation_x	46
4.18.2.7	rotation_y	46
4.18.2.8	rotation_z	46
4.18.2.9	write_la	47
4.19	out_silo_module Module Reference	47
4.19.1	Detailed Description	47
4.19.2	Function/Subroutine Documentation	47
4.19.2.1	outputsilo	47
4.19.2.2	writeblocks	48
4.19.2.3	writemaster	48
4.20	output Module Reference	48
4.20.1	Detailed Description	48
4.20.2	Function/Subroutine Documentation	49
4.20.2.1	write_output	49
4.21	parameters Module Reference	50
4.21.1	Detailed Description	52
4.21.2	Variable Documentation	52
4.21.2.1	mpi_real_kind	52

4.22	sources Module Reference	52
4.22.1	Detailed Description	52
4.22.2	Function/Subroutine Documentation	53
4.22.2.1	divbcorr_source	53
4.22.2.2	divergence_b	54
4.22.2.3	getpos	54
4.22.2.4	grav_source	54
4.22.2.5	radpress_source	55
4.22.2.6	source	55
4.23	thermal_cond Module Reference	56
4.23.1	Detailed Description	57
4.23.2	Function/Subroutine Documentation	57
4.23.2.1	dt_cond	57
4.23.2.2	heatfluxes	57
4.23.2.3	init_thermal_cond	58
4.23.2.4	ksp	58
4.23.2.5	progress	58
4.23.2.6	st_steps	58
4.23.2.7	substep	58
4.23.2.8	superstep	59
4.23.2.9	thermal_bounds	59
4.23.2.10	thermal_conduction	59
4.24	user_mod Module Reference	59
4.24.1	Detailed Description	60
4.24.2	Function/Subroutine Documentation	60
4.24.2.1	impose_user_bc	60
4.24.2.2	initial_conditions	60
5	File Documentation	61
5.1	doc/mainpage.h File Reference	61
5.2	src/boundaries.f90 File Reference	61
5.2.1	Detailed Description	61
5.3	src/coldens.f90 File Reference	61
5.3.1	Detailed Description	62
5.3.2	Function/Subroutine Documentation	62
5.3.2.1	coldens	62
5.4	src/constants.f90 File Reference	63
5.4.1	Detailed Description	64
5.5	src/cooling_chi.f90 File Reference	64
5.5.1	Detailed Description	65

5.6	src/cooling_dmc.f90 File Reference	65
5.6.1	Detailed Description	65
5.7	src/cooling_h.f90 File Reference	65
5.7.1	Detailed Description	66
5.8	src/difrad.f90 File Reference	66
5.8.1	Detailed Description	67
5.9	src/exoplanet.f90 File Reference	67
5.9.1	Detailed Description	69
5.10	src/globals.f90 File Reference	69
5.10.1	Detailed Description	70
5.11	src/h_alpha_proj.f90 File Reference	70
5.11.1	Detailed Description	71
5.11.2	Function/Subroutine Documentation	71
5.11.2.1	h_alpha_proj	71
5.12	src/hll.f90 File Reference	71
5.12.1	Detailed Description	72
5.13	src/hllc.f90 File Reference	72
5.13.1	Detailed Description	72
5.14	src/hlld.f90 File Reference	72
5.14.1	Detailed Description	73
5.15	src/hlle.f90 File Reference	73
5.15.1	Detailed Description	73
5.16	src/hydro_core.f90 File Reference	73
5.16.1	Detailed Description	74
5.17	src/hydro_solver.f90 File Reference	74
5.17.1	Detailed Description	75
5.18	src/init.f90 File Reference	75
5.18.1	Detailed Description	75
5.19	src/lyman_alpha_tau.f90 File Reference	75
5.19.1	Detailed Description	76
5.19.2	Function/Subroutine Documentation	77
5.19.2.1	lyman_alpha_tau	77
5.20	src/main.f90 File Reference	77
5.20.1	Detailed Description	78
5.21	src/Out_Silo_Module.f90 File Reference	78
5.21.1	Detailed Description	78
5.22	src/output.f90 File Reference	78
5.22.1	Detailed Description	79
5.23	src/parameters.f90 File Reference	79
5.23.1	Detailed Description	81

5.24	src/sources.f90 File Reference	81
5.24.1	Detailed Description	82
5.25	src/thermal_cond.f90 File Reference	82
5.25.1	Detailed Description	83
5.26	src/user_mod.f90 File Reference	83
5.26.1	Detailed Description	83
Index		85

Chapter 1

GUACHO-3D Documentation

Authors

Alejandro Esquivel et al.

1.1 Introduction

Documentation of the Guacho code

1.2 release.notes

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/gpl.html>

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:

boundaries	Boundary conditions	7
coldens_utilities	Column densirt projection	9
constants	Module containing physical and asronomical constants	12
cooling_chi	Cooling module with CHIANTI generated cooling curves	13
cooling_dmc	Cooling module with Dalgarno McCray coronal cooling curve	14
cooling_h	Cooling with parametrized cooling and H rate equation	15
difrad	Ray tracing Radiative Trasnport	18
exoplanet	Exoplanet module	22
globals	Module containing global variables	23
h_alpha_utilities	H alpha projection	24
hll	HLL approximate Riemann solver module	29
hllc	HLLC approximate Riemann solver module	30
hlld	HLLD approximate Riemann solver module	31
hlle	HLL E approximate Riemann solver module	33
hydro_core	Basic hydro (and MHD) subroutines utilities	34
hydro_solver	Advances the simulation one timestep	39
init	Guacho-3D initialization	42
lyman_alpha_utilities	Lyman_alpha_utilities	44
out_silo_module	Output in Silo (+HDF5) Format	47

output		
	Writes output	48
parameters		
	Parameters module	50
sources		
	Adds source terms	52
thermal_cond		
	Adds (isotropuic) thermal conduction	56
user_mod		
	User imput module	59

Chapter 3

File Index

3.1 File List

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

Makefile	??
doc/mainpage.h	
Webpage frontend	61
src/boundaries.f90	
Boundary conditions	61
src/coldens.f90	
Column density projection	61
src/constants.f90	
Constants module	63
src/cooling_chi.f90	
Cooling module with CHIANTI generated cooling curves	64
src/cooling_dmc.f90	
Cooling module with Dlgarno Mac Cray coronal cooling curve	65
src/cooling_h.f90	
Cooling with hydrogen rate parametrized cooling	65
src/difrad.f90	
Diffuse radiation module	66
src/exoplanet.f90	
Exoplanet problem module	67
src/globals.f90	
Global variables	69
src/h_alpha_proj.f90	
H alpha projection	70
src/hll.f90	
HLL approximate Riemann solver module	71
src/hllc.f90	
HLLC approximate Riemann solver module	72
src/hlld.f90	
HLLD approximate Riemann solver module	72
src/hlle.f90	
HLL E approximate Riemann solver module	73
src/hydro_core.f90	
Hydrodynamical and Magnetohydrodynamical basic module	73
src/hydro_solver.f90	
Hydrodynamical and Magnetohydrodynamical solver module	74
src/init.f90	
Guacho-3D initialization module	75

src/lyman_alpha_tau.f90	
Lyman_alpha_utilities	75
src/main.f90	
Guacho-3D main program	77
src/Out_Silo_Module.f90	
Output in Silo Format	78
src/output.f90	
Writes Output	78
src/parameters.f90	
Parameters module	79
src/sources.f90	
Adds source terms	81
src/thermal_cond.f90	
Thermal conduction module	82
src/user_mod.f90	
User input module	83

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

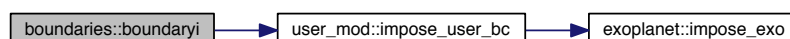
The conditions only are imposed at the innermost ghost cell, on the u (unstepped) variables

Parameters

<i>real</i>	[in] optional, time : integration time
<i>real</i>	[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

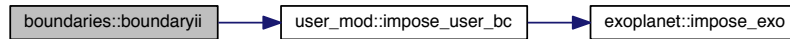
The conditions only are imposed in two ghost cells on the up (stepped) variables

Parameters

<i>real</i>	[in] optional, time : integration time
<i>real</i>	[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*)

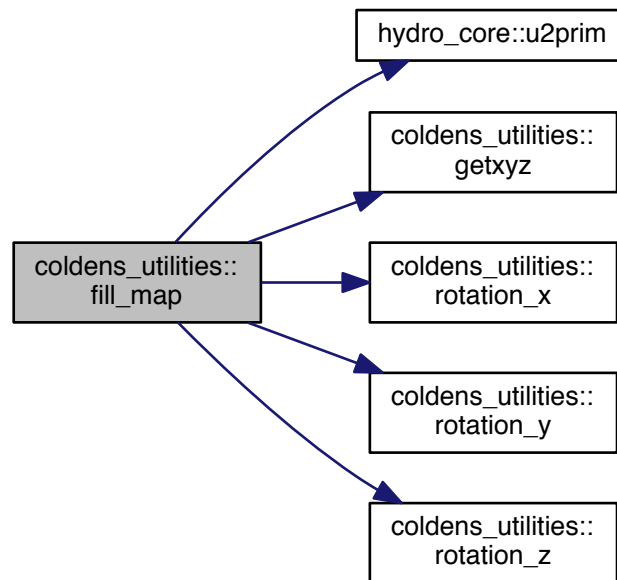
Fills the target map of one MPI block

Parameters

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

Definition at line 286 of file coldens.f90.

Here is the call graph for this function:



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

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

<i>real</i>	[in] y : y position in the grid
<i>real</i>	[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

<i>real</i>	[out] u(neq,nxmin:nxmax,nymin:nymax,nzmin:nzmax) : conserved variables
<i>integer</i>	[in] itprint : number of output
<i>string</i>	[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

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

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

<i>real</i>	[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

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

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

<i>real</i>	[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

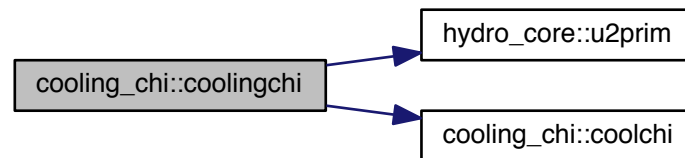
cooling is applied in the entire domain and updates both the conserved and primitive variables

Parameters

<i>real</i>	[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/coolingCHIANTI.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

<i>real</i>	[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

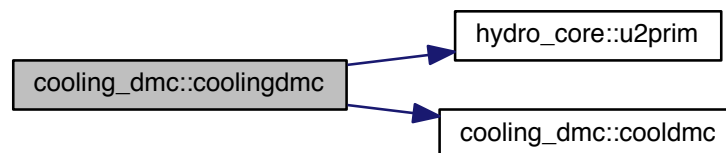
cooling is applied in the entire domain and updates both the conserved and primitive variables

Parameters

<i>real</i>	[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 curve 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 applies 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

<i>real8</i>	[in] x1 : initial H ionization fraction
<i>real8</i>	[in] x2 : final H ionization fraction
<i>real</i>	[in] dt : timestep
<i>real8</i>	[in] den : total density of hydrogen
<i>real8</i>	[in] dh0 : density of neutral hydrogen
<i>real8</i>	[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

<i>real8</i>	[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

<i>real8</i>	[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)`

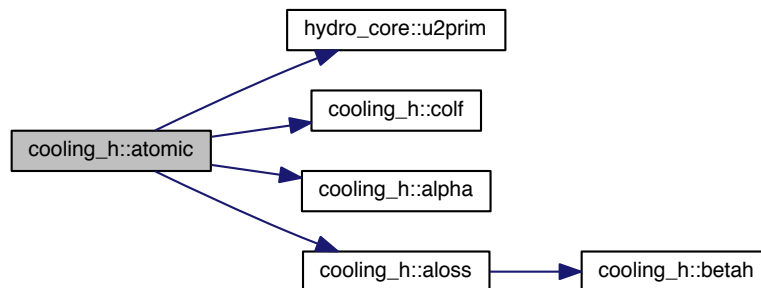
Calculates the new ionization state and energy density using a time dependent ionization calculation and an approximate time dependent cooling calculation

Parameters

<i>real</i>	[in] dt : timestep (seconds)
<i>real</i>	[in] uu(neq) : conserved variables in one cell
<i>real</i>	[in] tau : optical depth (not in use)
<i>real</i>	[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

<i>real</i>	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

<i>real8[in]</i>	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

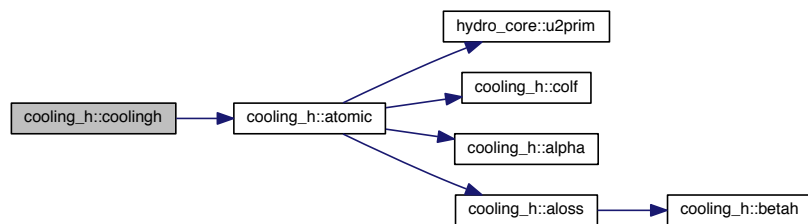
parametrized cooling curve, uses the ionization state of hydrogen and ties the O I and II to it

Parameters

<i>real</i>	[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
Photoionization 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 Transport

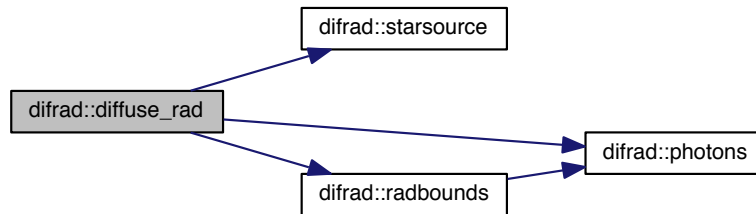
4.7.2 Function/Subroutine Documentation

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

<i>real</i>	[out] <i>emax</i> : 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) *x/0*, *real*, intent(in) *y/0*, *real*, intent(in) *z/0*, *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

<i>real</i>	[in] xI0 : Initial X position
<i>real</i>	[in] yI0 : Initial Y position
<i>real</i>	[in] zI0 : Initial Z position
<i>real</i>	[in] xd : Direction in X
<i>real</i>	[in] yd : Direction in Y
<i>real</i>	[in] zd : Direction in Z
<i>real</i>	[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

<i>integer</i>	[in] j : current iteration
<i>integer</i>	[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

<i>real</i>	[out] xd : x component
<i>real</i>	[out] yd : y component
<i>real</i>	[out] zd : z component

Definition at line 149 of file difrad.f90.

4.7.2.8 subroutine difrad::starsource (real, intent(in) srاد, 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

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

<i>real</i>	[out] <code>u(neq,nxmin:nxmax,nymin:nymax,nzmin:nzmax)</code> : conserver variables
<i>real</i>	[time] <code>time</code> : 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 variables
- real, `dimension(:, :, :)`, allocatable [up](#)
conserved variables after 1/2 timestep
- real, `dimension(:, :, :)`, allocatable [primit](#)
primitive variables
- 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 communicator.

4.9.1 Detailed Description

This module 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*)

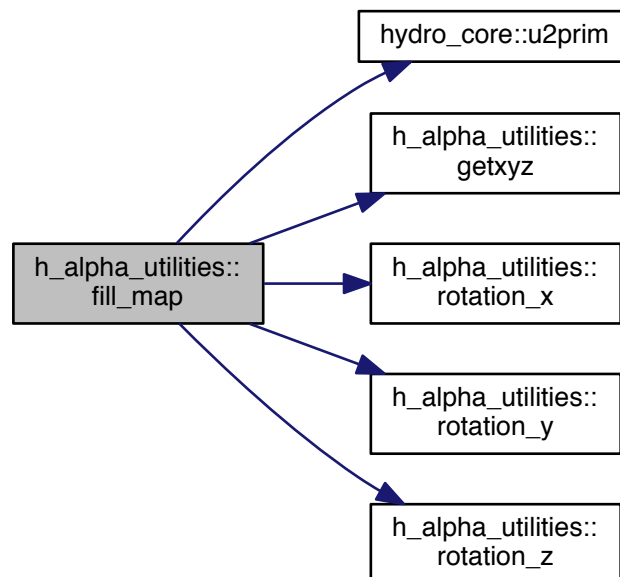
Fills the target map of one MPI block

Parameters

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

Definition at line 285 of file `h_alpha_proj.f90`.

Here is the call graph for this function:



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

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

<i>real</i>	[out] u(neq,nxmin:nxmax,nymin:nymax,nzmin:nzmax) : conserved variables
<i>integer</i>	[in] itprint : number of output
<i>string</i>	[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

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

Parameters

<i>real</i>	[in], theta : Angle of rotation (in radians)
-------------	--

<i>real</i>	[in], x : original x position in the grid
<i>real</i>	[in], y : original y position in the grid
<i>real</i>	[in], x : original z position in the grid
<i>real</i>	[out], x : final x position in the grid
<i>real</i>	[out], y : final y position in the grid
<i>real</i>	[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

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

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

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

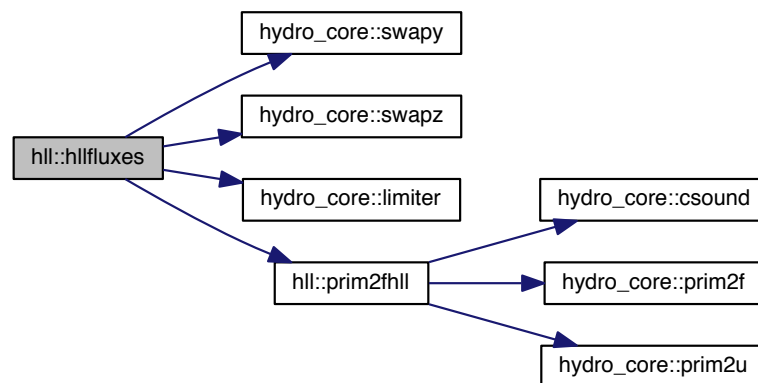
Calculates HLL fluxes from the primitive variables on all the domain

Parameters

<i>integer</i>	[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 between PL and PR using the HLL solver

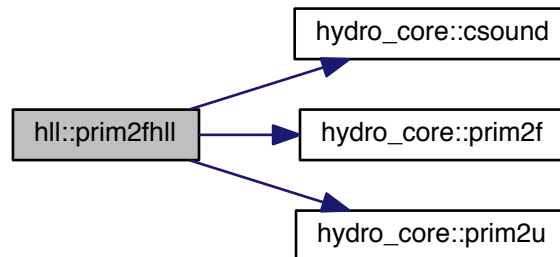
The fluxes are computed in the X direction, to obtain the y and z directions a swap is performed

Parameters

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

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)

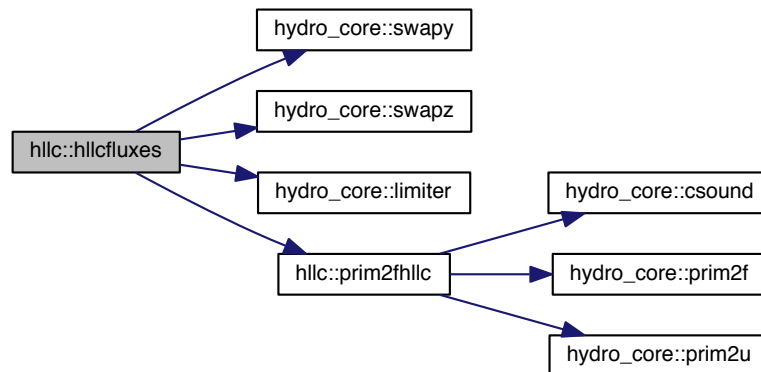
Calculates HLLC fluxes from the primitive variables on all the domain

Parameters

<i>integer</i>	[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*)

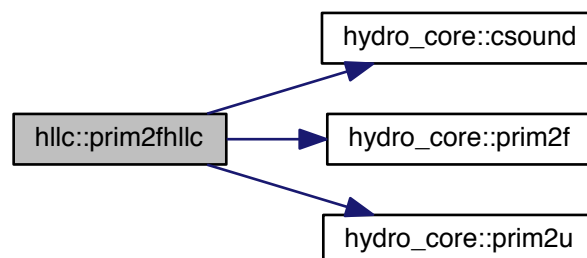
Solves the Riemann problem at the interface between PL and PR using the HLLC solver
The fluxes are computed in the X direction, to obtain the y and z directions a swap is performed

Parameters

<i>real</i>	[in] primL : primitives at the Left state
<i>real</i>	[in] primR : primitives at the Right state
<i>real</i>	[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 `prim2fhld` (`priml`, `primr`, `ff`)
Solves the Riemann problem at the interface PL,PR using the HLLD solver.
- subroutine `hldfluxes` (`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 `hld::hldfluxes` (`integer`, `intent(in)` *choice*)

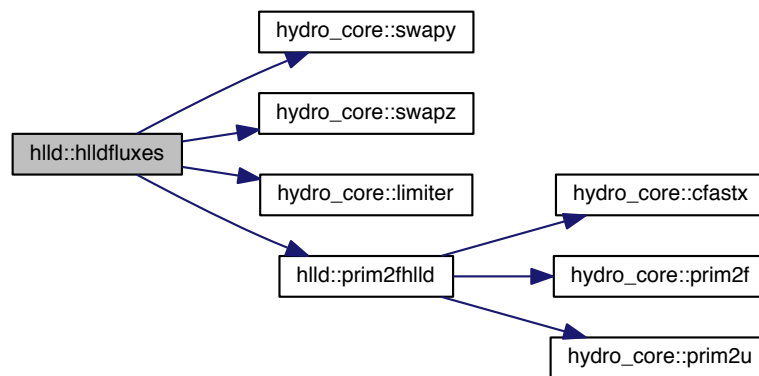
Calculates HLLD fluxes from the primitive variables on all the domain

Parameters

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

Definition at line 319 of file `hld.f90`.

Here is the call graph for this function:



4.13.2.2 subroutine `hld::prim2fhld` (`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 between PL and PR using the HLLD solver

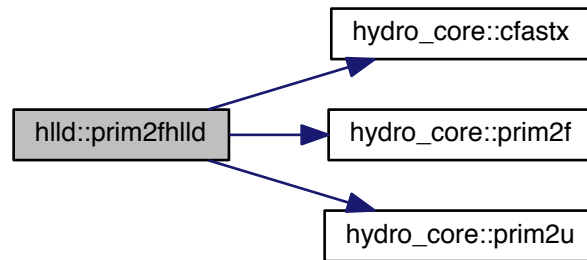
The fluxes are computed in the X direction, to obtain the y and z directions a swap is performed

Parameters

<i>real</i>	[in] primL : primitives at the Left state
<i>real</i>	[in] primR : primitives at the Right state
<i>real</i>	[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)

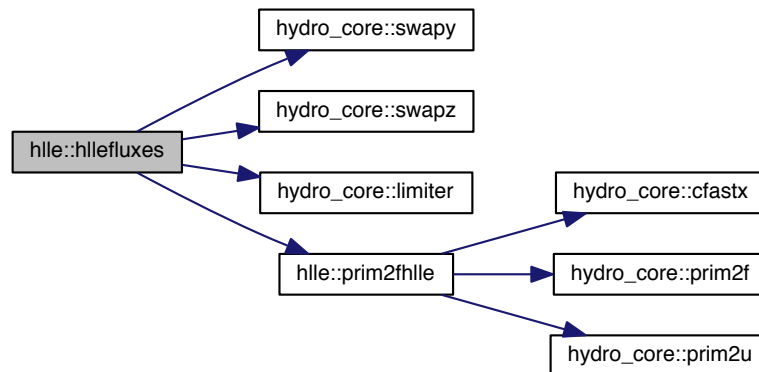
Calculates HLLE fluxes from the primitive variables on all the domain

Parameters

<i>integer</i>	[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::prim2fhle` (*real*, dimension(neq), intent(in) *priml*, *real*, dimension(neq), intent(in) *primr*, *real*, dimension(neq), intent(inout) *ff*)

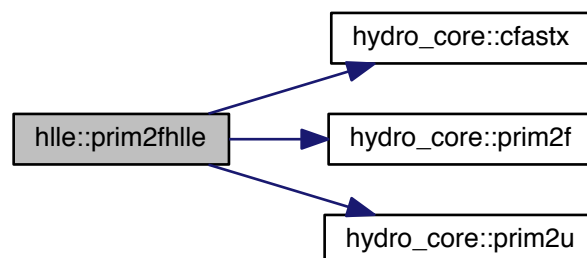
Solves the Riemann problem at the interface between PL and PR using the HLL solver
The fluxes are computed in the X direction, to obtain the y and z directions a swap is performed

Parameters

<i>real</i>	[in] <i>primL</i> : primitives at the Left state
<i>real</i>	[in] <i>primR</i> : primitives at the Right state
<i>real</i>	[out] <i>ff</i> : 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 `calcpri` (u, primit)
Updated the primitives, using the conserved variables in the entire domain.
- subroutine `prim2u` (prim, uu)
Computes the 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)
Obtains 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::calcpri` (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

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

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

<i>real</i>	[in] <i>prim</i> (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

<i>real</i>	[in] <i>p</i> : value of pressure
<i>real</i>	[in] <i>d</i> : value of density
<i>real</i>	[out] <i>cs</i> : sound speed

Definition at line 299 of file hydro_core.f90.

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

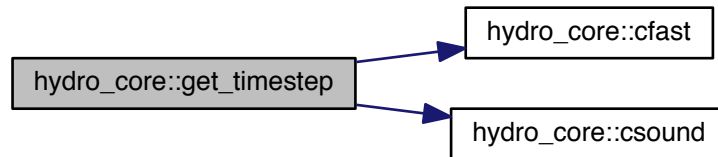
Obtains the timestep allowed by the CFL condition in the entire domain using the global primitives

Parameters

<i>real</i>	[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 between 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

<i>real</i>	[in] : primitives at the left of the left state
<i>real</i>	[inout] : primitives at the left state
<i>real</i>	[inout] : primitives at the right state
<i>real</i>	[in] : primitives at the right of the right state
<i>real</i>	[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 primitives

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

<i>real</i>	[in] <i>prim(neq)</i> : primitives in one cell
<i>real</i>	[out] <i>ff(neq)</i> : 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 variables from the primitives in a single cell

Parameters

<i>real</i>	[in] <i>prim(neq)</i> : primitives in one cell
<i>real</i>	[out] <i>uu(neq)</i> : conserved variables 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

<i>real</i>	[inout] var(neq) : variable to be swapped
<i>real</i>	[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

<i>real</i>	[inout] var(neq) : variable to be swapped
<i>real</i>	[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

<i>real</i>	[in] uu(neq) : conserved variables in one cell
<i>real</i>	[out] prim(neq) : primitives in one cell
<i>real</i>	[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 advance 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

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

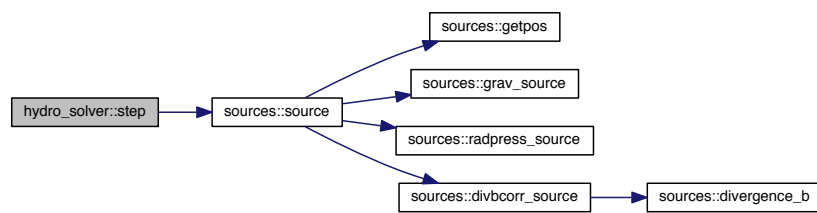
(in 3D), it takes U^{n+1} =up from the global variables and U^n =u

Parameters

<i>real</i>	[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)

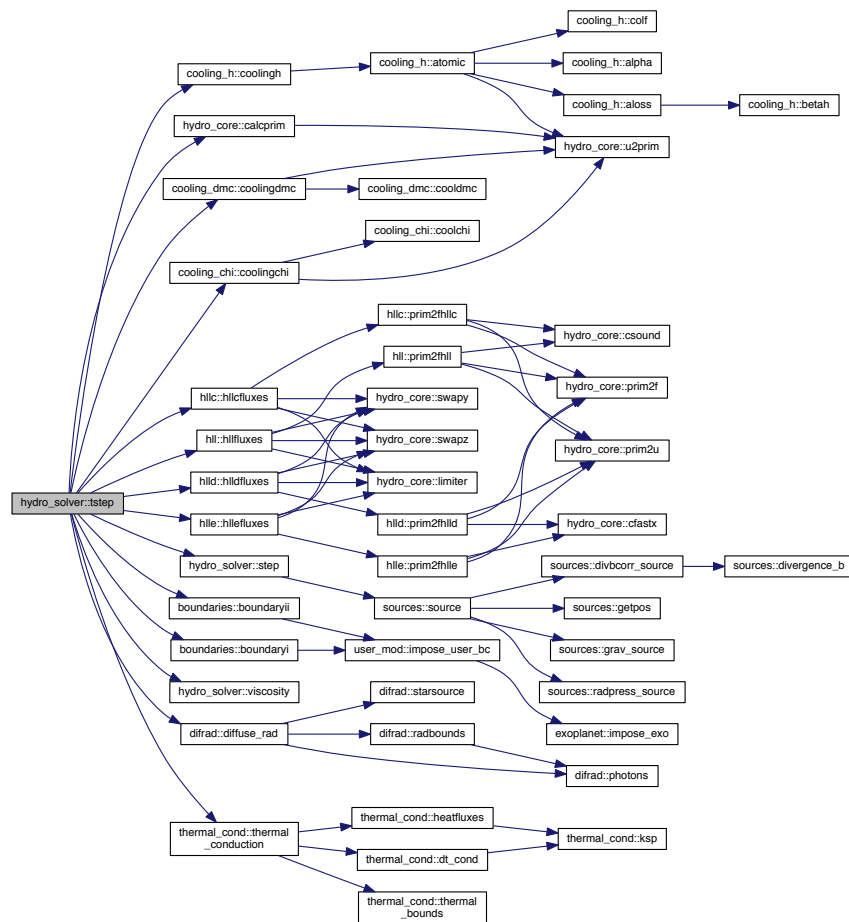
High level wrapper to advance the simulation
The variables are taken from the globals module.

Parameters

<i>real</i>	[in] time : integration time
<i>real</i>	[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 initialize the code, it also initializes all the modules set by the user.

4.17.2 Function/Subroutine Documentation

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

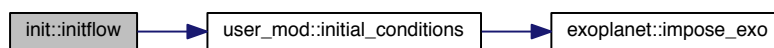
Initializes the conserved variables, in the globals module

Parameters

<i>real</i>	[inout] <i>itprint</i> : 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*)

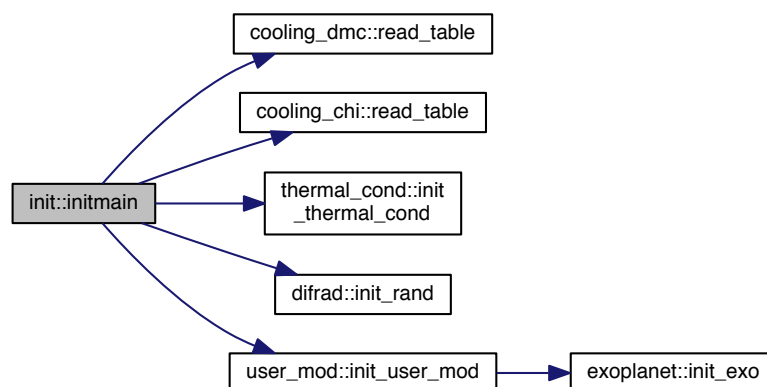
This subroutine initializes all the variables in the globals module, MPI, cooling and [user_mod](#) routines; and outputs to screen the main parameters used in the run

Parameters

<i>real</i>	[out] <i>time</i> : time (code units)
<i>real</i>	[out] <i>tprint</i> : time of next output
<i>integer</i>	[out] <i>itprint</i> : 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

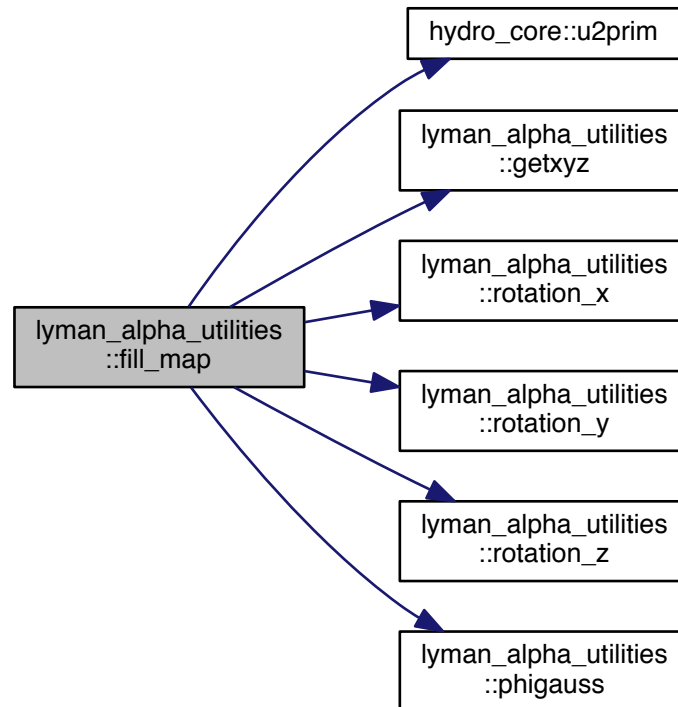
Parameters

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

<i>real</i>	[in] thetaz : Rotation around Z
-------------	---------------------------------

Definition at line 284 of file lyman_alpha_tau.f90.

Here is the call graph for this function:



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

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

<i>real</i>	[out] <i>u</i> (neq,nxmin:nxmax,nymin:nymax,nzmin:nzmax) : conserved variables
<i>integer</i>	[in] <i>itprint</i> : number of output
<i>string</i>	[in] <i>filepath</i> : 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

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

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

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

<i>integer</i>	[in] <i>itprint</i> : number of output
<i>string</i>	[in] <i>filepath</i> : path where to write
<i>integer</i>	[in] <i>nxmap</i> : Number of X cells in target
<i>integer</i>	[in] <i>nymap</i> : Number of Y cells in target
<i>integer</i>	[in] <i>nvmap</i> : Number of velocity channels
<i>real</i>	[in] <i>map(nxmap,nymap)</i> : 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 output in SILO (HDF5) format

4.19.2 Function/Subroutine Documentation

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

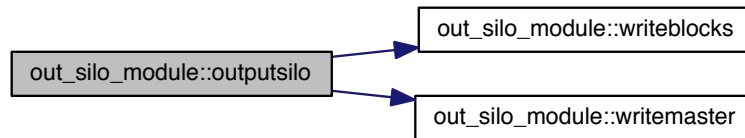
Upper level wrapper for the SILO output

Parameters

<i>integer</i>	[in] itprint : number of output
----------------	---------------------------------

Definition at line 347 of file Out_Silo_Module.f90.

Here is the call graph for this function:



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

Writes Data in silo format one file per processor

Parameters

<i>integer</i>	[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

<i>integer</i>	[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 output 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

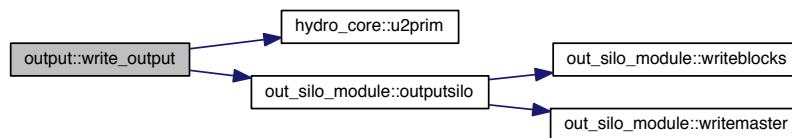
Supported formats are *.dat (ASCII), *.bin and VTK (both BINARY), Silo (+hdf5)

Parameters

<i>integer</i>	[in] itprint : number of output
----------------	---------------------------------

Definition at line 41 of file output.f90.

Here is the call graph for this function:



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 $\text{div}(B)$
- subroutine `divbcorr_source` (i, j, k, pp, s)
8 Wave source terms for $\text{div}(B)$ 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 $\text{div}(B)$ 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 $\text{div } \mathbf{B}$ in Faraday's Law, momentum equation and energy equation as proposed in Powell et al. 1999

Parameters

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

<i>integer</i>	[in] <i>i</i> : cell index in the X direction
<i>integer</i>	[in] <i>j</i> : cell index in the Y direction
<i>integer</i>	[in] <i>k</i> : cell index in the Z direction
<i>real</i>	[out] <i>d</i> :: 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

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

<i>real</i>	[in] xc : X position of the cell
<i>real</i>	[in] yc : Y position of the cell
<i>real</i>	[in] zc : Z position of the cell
<i>real</i>	[in] pp(neq) : vector of primitive variables
<i>real</i>	[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 radiation pressure force due to photo-ionization

Parameters

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

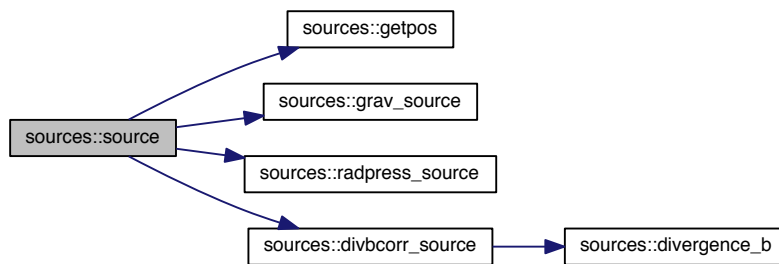
Main driver, this is called from the upwind stepping

Parameters

<i>integer</i>	[in] <i>i</i> : cell index in the X direction
<i>integer</i>	[in] <i>j</i> : cell index in the Y direction
<i>integer</i>	[in] <i>k</i> : cell index in the Z direction
<i>real</i>	[in] prim(neq) : vector of primitive variables
<i>real</i>	[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 (isotropic) thermal conduction.

Functions/Subroutines

- subroutine `init_thermal_cond` ()
Initializes 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

<i>real</i>	[out] <i>dt</i> :: 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 ()

Initializes 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

<i>real</i>	[in] <i>T</i> : 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

<i>integer</i>	[in] <i>j</i> : current iteration
<i>integer</i>	[in] <i>tot</i> : 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

<i>real</i>	<i>fs</i> : ratio of dtcond/dthydro
<i>integer</i>	<i>Ns</i> : Number of Supersteps
<i>real</i>	<i>fstep</i> : 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

<i>integer</i>	[in] <i>j</i> : index of current step
<i>integer</i>	[in] <i>N</i> : Total number of substeps
<i>real</i>	[in] <i>nu</i> : 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

<i>integer</i>	[in] <i>N</i> : Nunber of inner substeps
<i>real</i>	[in] <i>snu</i> : 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*)

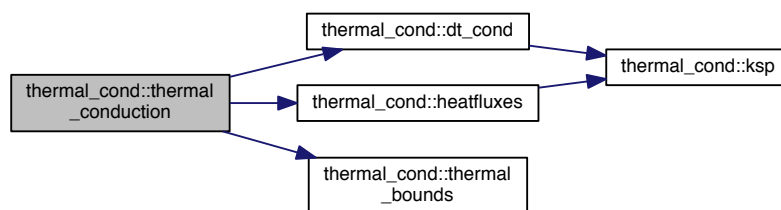
This routine adds the heat conduction, receives the hydro timestep in seconds, and assumes the primitives and Temp(*i,j,k*) arrays are updated

Parameters

<i>real</i>	[in] <i>dt</i> : Hysdrodynamical timestep (in seconds)
-------------	--

Definition at line 408 of file thermal_cond.f90.

Here is the call graph for this function:



4.24 user_mod Module Reference

User input 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 needed 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

<i>real</i>	[out] u(neq,nxmin:nxmax,nymin:nymax,nzmin:nzmax) : conserved variables
<i>real</i>	[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

<i>real</i>	[out] u(neq,nxmin:nxmax,nymin:nymax,nzmin:nzmax) : conserved variables
<i>real</i>	[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

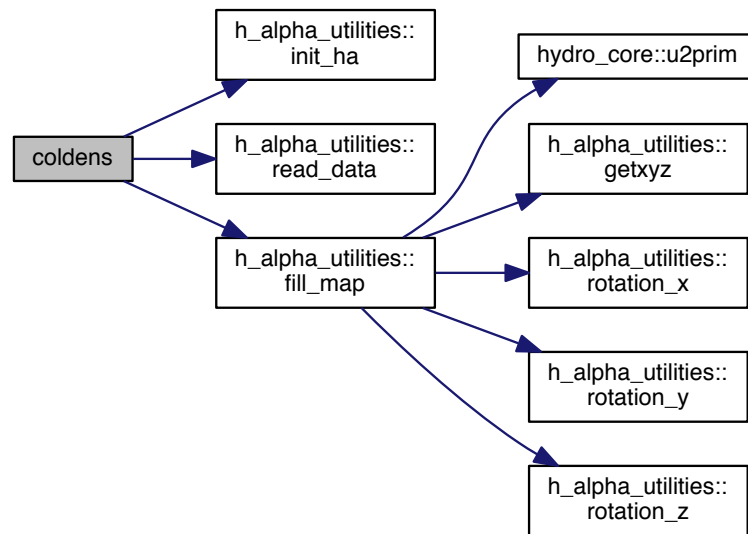
5.3.2.1 program `coldens` ()

Computes the H-alpha absorption

It rotates the data along each of the coordinates axis by an amount θ_x , θ_y , θ_z , and projects 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 applies 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

Photoionization 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. Schneider, 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 variables
- real, dimension(:, :, :), allocatable [globals::up](#)
conserved variables after 1/2 timestep
- real, dimension(:, :, :), allocatable [globals::primit](#)
primitive variables
- 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 communicator.

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

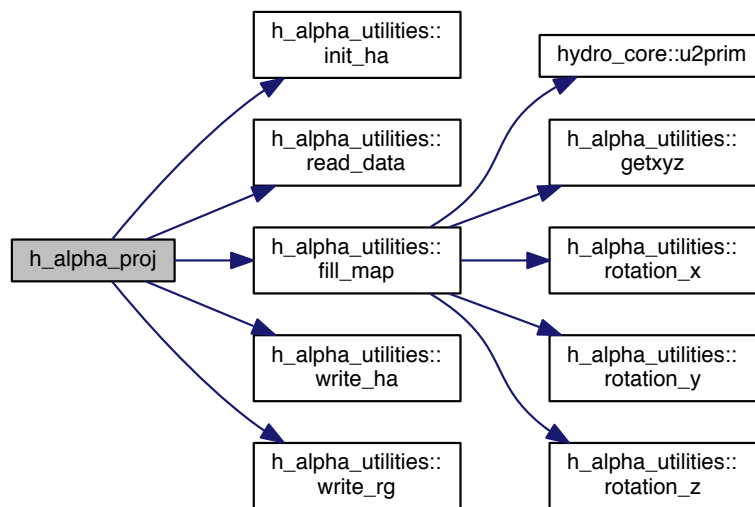
5.11.2.1 program h_alpha_proj ()

Computes the H-alpha absorption

It rotates the data along each of the coordinates axis by an amount $\theta_x, \theta_y, \theta_z$, and projects 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:



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::prim2fhld](#) (priml, primr, ff)
Solves the Riemann problem at the interface PL,PR using the HLLD solver.
- subroutine [hlld::hldfluxes](#) (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

HLLC approximate Riemann solver module.

Modules

- module [hlle](#)
HLLC approximate Riemann solver module.

Functions/Subroutines

- subroutine [hlle::prim2fhle](#) (priml, primr, ff)
Solves the Riemann problem at the interface PL,PR using the HLLC solver.
- subroutine [hlle::hlefluxes](#) (choice)
Calculates HLLC 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 Magnetohydrodynamical basic 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::calcpim` (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)
Obtains 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 Magnetohydrodynamocal 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 advance 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 absorption.

5.19.1 Detailed Description

Author

Alejandro Esquivel

Date

2/Nov/2014

5.19.2 Function/Subroutine Documentation

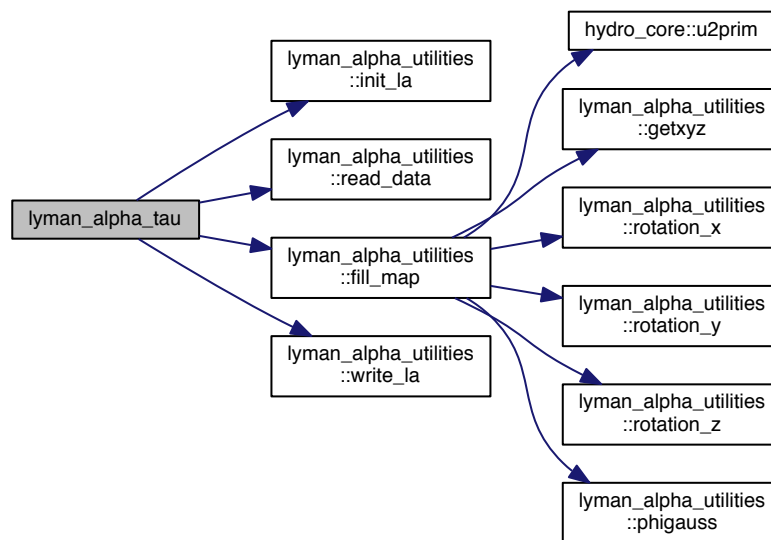
5.19.2.1 program lyman_alpha_tau ()

Computes the Ly-alpha absorption

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:



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 integrates 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 by 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::divbcrr_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 (isotropic) thermal conduction.

Functions/Subroutines

- subroutine [thermal_cond::init_thermal_cond](#) ()
Initializes 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 input 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

- aloss
 - cooling_h, [16](#)
- alpha
 - cooling_h, [16](#)
- alpha1
 - cooling_h, [17](#)
- atomic
 - cooling_h, [17](#)
- betah
 - cooling_h, [17](#)
- boundaries, [7](#)
 - boundaryi, [7](#)
 - boundaryii, [7](#)
- boundaryi
 - boundaries, [7](#)
- boundaryii
 - boundaries, [7](#)
- calcprim
 - hydro_core, [35](#)
- cfast
 - hydro_core, [35](#)
- cfastx
 - hydro_core, [36](#)
- coldens
 - coldens.f90, [62](#)
- coldens.f90
 - coldens, [62](#)
- coldens_utilities, [9](#)
 - fill_map, [9](#)
 - getxyz, [10](#)
 - init_coldens, [11](#)
 - read_data, [11](#)
 - rotation_x, [11](#)
 - rotation_y, [11](#)
 - rotation_z, [12](#)
 - write_map, [12](#)
- colf
 - cooling_h, [18](#)
- constants, [12](#)
- coolchi
 - cooling_chi, [13](#)
- cooldmc
 - cooling_dmc, [15](#)
- cooling_chi, [13](#)
 - coolchi, [13](#)
 - coolingchi, [14](#)
 - read_table, [14](#)
- cooling_dmc, [14](#)
 - cooldmc, [15](#)
 - coolingdmc, [15](#)
 - read_table, [15](#)
- cooling_h, [15](#)
 - aloss, [16](#)
 - alpha, [16](#)
 - alpha1, [17](#)
 - atomic, [17](#)
 - betah, [17](#)
 - colf, [18](#)
 - coolingh, [18](#)
- coolingchi
 - cooling_chi, [14](#)
- coolingdmc
 - cooling_dmc, [15](#)
- coolingh
 - cooling_h, [18](#)
- csound
 - hydro_core, [36](#)
- diffuse_rad
 - difrad, [19](#)
- difrad, [18](#)
 - diffuse_rad, [19](#)
 - emdiff, [20](#)
 - init_rand, [20](#)
 - photons, [20](#)
 - progress, [21](#)
 - radbounds, [21](#)
 - random_versor, [21](#)
 - starsource, [21](#)
- divbcorr_source
 - sources, [53](#)
- divergence_b
 - sources, [54](#)
- doc/mainpage.h, [61](#)
- dt_cond
 - thermal_cond, [57](#)
- emdiff
 - difrad, [20](#)
- exoplanet, [22](#)
 - impose_exo, [23](#)
 - init_exo, [23](#)
- fill_map
 - coldens_utilities, [9](#)
 - h_alpha_utilities, [25](#)
 - lyman_alpha_utilities, [44](#)
- get_timestep

- hydro_core, 36
- getpos
 - sources, 54
- getxyz
 - coldens_utilities, 10
 - h_alpha_utilities, 25
 - lyman_alpha_utilities, 45
- globals, 23
- grav_source
 - sources, 54
- h_alpha_proj
 - h_alpha_proj.f90, 71
- h_alpha_proj.f90
 - h_alpha_proj, 71
- h_alpha_utilities, 24
 - fill_map, 25
 - getxyz, 25
 - init_ha, 27
 - read_data, 27
 - rotation_x, 27
 - rotation_y, 27
 - rotation_z, 28
 - write_ha, 28
 - write_rg, 28
- heatfluxes
 - thermal_cond, 57
- hll, 29
 - hllfluxes, 29
 - prim2fhll, 29
- hllc, 30
 - hllcfluxes, 30
 - prim2fhllc, 31
- hllcfluxes
 - hllc, 30
- hlld, 31
 - hlldfluxes, 32
 - prim2fhlld, 32
- hlldfluxes
 - hlld, 32
- hlle, 33
 - hllfluxes, 33
 - prim2fhlle, 34
- hllfluxes
 - hlle, 33
- hllfluxes
 - hll, 29
- hydro_core, 34
 - calcprim, 35
 - cfast, 35
 - cfastx, 36
 - csound, 36
 - get_timestep, 36
 - limiter, 37
 - prim2f, 37
 - prim2u, 37
 - swapy, 37
 - swapz, 39
 - u2prim, 39
- hydro_solver, 39
 - step, 39
 - tstep, 41
 - viscosity, 42
- impose_exo
 - exoplanet, 23
- impose_user_bc
 - user_mod, 60
- init, 42
 - initflow, 43
 - initmain, 43
- init_coldens
 - coldens_utilities, 11
- init_exo
 - exoplanet, 23
- init_ha
 - h_alpha_utilities, 27
- init_la
 - lyman_alpha_utilities, 45
- init_rand
 - difrad, 20
- init_thermal_cond
 - thermal_cond, 57
- initflow
 - init, 43
- initial_conditions
 - user_mod, 60
- initmain
 - init, 43
- ksp
 - thermal_cond, 58
- limiter
 - hydro_core, 37
- lyman_alpha_tau
 - lyman_alpha_tau.f90, 77
- lyman_alpha_tau.f90
 - lyman_alpha_tau, 77
- lyman_alpha_utilities, 44
 - fill_map, 44
 - getxyz, 45
 - init_la, 45
 - phigauss, 45
 - read_data, 46
 - rotation_x, 46
 - rotation_y, 46
 - rotation_z, 46
 - write_la, 47
- mpi_real_kind
 - parameters, 52
- out_silo_module, 47
 - outputsilo, 47
 - writeblocks, 48
 - writemaster, 48
- output, 48

- write_output, [49](#)
- outputsilo
 - out_silo_module, [47](#)
- parameters, [50](#)
 - mpi_real_kind, [52](#)
- phigauss
 - lyman_alpha_utilities, [45](#)
- photons
 - difrad, [20](#)
- prim2f
 - hydro_core, [37](#)
- prim2fhll
 - hll, [29](#)
- prim2fhllc
 - hllc, [31](#)
- prim2fhlld
 - hlld, [32](#)
- prim2fhlle
 - hlle, [34](#)
- prim2u
 - hydro_core, [37](#)
- progress
 - difrad, [21](#)
 - thermal_cond, [58](#)
- radbounds
 - difrad, [21](#)
- radpress_source
 - sources, [55](#)
- random_versor
 - difrad, [21](#)
- read_data
 - coldens_utilities, [11](#)
 - h_alpha_utilities, [27](#)
 - lyman_alpha_utilities, [46](#)
- read_table
 - cooling_chi, [14](#)
 - cooling_dmc, [15](#)
- rotation_x
 - coldens_utilities, [11](#)
 - h_alpha_utilities, [27](#)
 - lyman_alpha_utilities, [46](#)
- rotation_y
 - coldens_utilities, [11](#)
 - h_alpha_utilities, [27](#)
 - lyman_alpha_utilities, [46](#)
- rotation_z
 - coldens_utilities, [12](#)
 - h_alpha_utilities, [28](#)
 - lyman_alpha_utilities, [46](#)
- source
 - sources, [55](#)
- sources, [52](#)
 - divbcorr_source, [53](#)
 - divergence_b, [54](#)
 - getpos, [54](#)
 - grav_source, [54](#)
 - radpress_source, [55](#)
 - source, [55](#)
- src/Out_Silo_Module.f90, [78](#)
- src/boundaries.f90, [61](#)
- src/coldens.f90, [61](#)
- src/constants.f90, [63](#)
- src/cooling_chi.f90, [64](#)
- src/cooling_dmc.f90, [65](#)
- src/cooling_h.f90, [65](#)
- src/difrad.f90, [66](#)
- src/exoplanet.f90, [67](#)
- src/globals.f90, [69](#)
- src/h_alpha_proj.f90, [70](#)
- src/hll.f90, [71](#)
- src/hllc.f90, [72](#)
- src/hlld.f90, [72](#)
- src/hlle.f90, [73](#)
- src/hydro_core.f90, [73](#)
- src/hydro_solver.f90, [74](#)
- src/init.f90, [75](#)
- src/lyman_alpha_tau.f90, [75](#)
- src/main.f90, [77](#)
- src/output.f90, [78](#)
- src/parameters.f90, [79](#)
- src/sources.f90, [81](#)
- src/thermal_cond.f90, [82](#)
- src/user_mod.f90, [83](#)
- st_steps
 - thermal_cond, [58](#)
- starsource
 - difrad, [21](#)
- step
 - hydro_solver, [39](#)
- substep
 - thermal_cond, [58](#)
- superstep
 - thermal_cond, [59](#)
- swapy
 - hydro_core, [37](#)
- swapz
 - hydro_core, [39](#)
- thermal_bounds
 - thermal_cond, [59](#)
- thermal_cond, [56](#)
 - dt_cond, [57](#)
 - heatfluxes, [57](#)
 - init_thermal_cond, [57](#)
 - ksp, [58](#)
 - progress, [58](#)
 - st_steps, [58](#)
 - substep, [58](#)
 - superstep, [59](#)
 - thermal_bounds, [59](#)
 - thermal_conduction, [59](#)
- thermal_conduction
 - thermal_cond, [59](#)
- tstep
 - hydro_solver, [41](#)

- u2prim
 - hydro_core, [39](#)
- user_mod, [59](#)
 - impose_user_bc, [60](#)
 - initial_conditions, [60](#)
- viscosity
 - hydro_solver, [42](#)
- write_ha
 - h_alpha_utilities, [28](#)
- write_la
 - lyman_alpha_utilities, [47](#)
- write_map
 - coldens_utilities, [12](#)
- write_output
 - output, [49](#)
- write_rg
 - h_alpha_utilities, [28](#)
- writeblocks
 - out_silo_module, [48](#)
- writemaster
 - out_silo_module, [48](#)