AGRIF

Generated by Doxygen 1.8.9.1

Tue Sep 1 2015 11:38:13

Contents

1 Agrif

1.1 Introduction

Blabla bla

1.2 Install :

1.2.1 Requirements

- a fortran compiler and a proper mpi implementation
- cmake $>$ 2.8

The install consists in 3 steps. First configuration of the package, makefile and other files generation, then build of the underlying fortran libraries and finally copy of the required files in the appropriate place.

We denote :

- SOURCEDIR as the directory which contains the sources (and this INSTALL file)
- BUILDDIR the directory where the package will be configured and build
- INSTALLDIR the directory where the package will be installed. Those 3 locations must be different.

1.2.2 Configuration :

You need to generate a makefile that fits with your platform, your compilers, the libraries versions and so on. That will take place in BUILDDIR.

1. get SOURCEDIR (i.e. download the package any way you want)
2. create BUILDDIR. Any place, preferably local to your machine for efficiency reasons.
3. enforce a fortran compiler using FC environment variable
4. Change to BUILDDIR and run cmake

```
mkdir $BUILDIR
cd $BUILDDIR
export FC=mpif90
cmake $SOURCEDIR
```

At the end of this step BUILDDIR contains all makefiles, setup.py and other required files for compilation.

Some useful options for cmake :

- -DUSE_DOXYGEN=ON : to use doxygen to generate source code documentation
- -DCMAKE_INSTALL_PREFIX=somewhere to choose where you want to install the library

example :

```
mkdir /home/mylogin/build_agrif
cd /home/mylogin/build_agrif
export FC=mpif90
module load cmake-2.8
cmake ~/Softs/agrif
```

2 Modules Index

2.1 Modules List

Here is a list of all modules with brief descriptions:

3 Data Type Index

3.1 Data Types List

Here are the data types with brief descriptions:

4 Module Documentation

4.1 agrif_bcfunction Module Reference

Module Agrif_BcFunction.

Data Types

- interface agrif_save_forrestore
- interface agrif_set_parent

Functions/Subroutines

- subroutine agrif_set_parent_int (integer_variable, value)

  This subroutine is used to set the type of the variable of the parent grid of the current grid as integer variable.
- subroutine agrif_set_parent_real4 (real_variable, value)

  This subroutine is used to set a real variable of the parent grid of the current grid as Single-precision real floating-point value.

- subroutine agrif_set_parent_real8 (real_variable, value)

  This subroutine is used to set a real variable of the parent grid of the current grid as Double-precision real floating-point value.
- subroutine agrif_set_restore (tabvarsindic)

  This subroutine is used to set the index of the current grid variable we want to restore.
- subroutine agrif_save_forrestore0d (tabvarsindic0, tabvarsindic)
- subroutine agrif_save_forrestore2d (q, tabvarsindic)

  This function is used to restore a current grid variable (with the index tabvarsindic) to the input 2D-variable.
- subroutine agrif_save_forrestore3d (q, tabvarsindic)

  This function is used to restore a current grid variable (with the index tabvarsindic) to the input 3D-variable.
- subroutine agrif_save_forrestore4d (q, tabvarsindic)

  This function is used to restore a current grid variable (with the index tabvarsindic) to the input 4D-variable.

4.1.1 Detailed Description

Module Agrif_BcFunction.

4.1.2 Function/Subroutine Documentation

4.1.2.1 subroutine agrif_bcfunction::agrif_set_parent_int (integer, intent(in) *integer_variable,* integer, intent(in) *value*)

This subroutine is used to set the type of the variable of the parent grid of the current grid as integer variable.

Parameters

in	*integer_variable*	indice of the variable in tabvars
in	*value*	input value

4.1.2.2 subroutine agrif_bcfunction::agrif_set_parent_real4 (real(kind=4), intent(in) *real_variable,* real(kind=4), intent(in) *value*)

This subroutine is used to set a real variable of the parent grid of the current grid as Single-precision real floating-point value.

Parameters

in	*real_variable*	input variable
in	*value*	input value for the parent grid

4.1.2.3 subroutine agrif_bcfunction::agrif_set_parent_real8 (real(kind=8), intent(in) *real_variable,* real(kind=8), intent(in) *value*)

This subroutine is used to set a real variable of the parent grid of the current grid as Double-precision real floating-point value.

Parameters

in	*real_variable*	input variable
in	*value*	input value for the parent grid

4.1.2.4 subroutine agrif_bcfunction::agrif_set_restore (integer, intent(in) *tabvarsindic*)

This subroutine is used to set the index of the current grid variable we want to restore.

Parameters

in	*tabvarsindic*	indice of the variable in tabvars

4.1.2.5 subroutine agrif_bcfunction::agrif_save_forrestore0d (integer, intent(in) *tabvarsindic0,* integer, intent(in) *tabvarsindic*)

Parameters

in	*tabvarsindic0*	index of the current grid variable
in	*tabvarsindic*	index of the varible which should be restored

4.1.2.6 subroutine agrif_bcfunction::agrif_save_forrestore2d (real, dimension(:,:), intent(in) *q,* integer, intent(in) *tabvarsindic*)

This function is used to restore a current grid variable (with the index tabvarsindic) to the input 2D-variable.

Parameters

in	*q*	input 2D-variable which should be saved
in	*tabvarsindic*	index of the current grid variable we want to restore

4.1.2.7 subroutine agrif_bcfunction::agrif_save_forrestore3d (real, dimension(:,:,:), intent(in) *q,* integer, intent(in) *tabvarsindic*)

This function is used to restore a current grid variable (with the index tabvarsindic) to the input 3D-variable.

Parameters

in	*q*	input 3D-variable which should be saved
in	*tabvarsindic*	index of the current grid variable we want to restore

4.1.2.8 subroutine agrif_bcfunction::agrif_save_forrestore4d (real, dimension(:,:,:,:), intent(in) *q,* integer, intent(in) *tabvarsindic*)

This function is used to restore a current grid variable (with the index tabvarsindic) to the input 4D-variable.

Parameters

in	*q*	input 4D-variable which should be saved
in	*tabvarsindic*	index of the current grid variable we want to restore

4.2 agrif_user_functions Module Reference

Module Agrif_user_Functions.

Data Types

- interface agrif_parent

Functions/Subroutines

- real function agrif_rel_dt ()

  Returns the time step of the current grid, relatively to the root grid (for which dt=1.).
- integer function agrif_rel_idt ()

  This function returns an integer which is the time refinement factor of the current grid, relatively to the root grid (for which time step = 1).
- integer function agrif_irhot ()

  This function returns an integer number which is the time refinement factor of the current grid.
- real function agrif_rhot ()

This function returns a real number which is the time refinement factor of the current grid.

- integer function agrif_parent_irhot ()

  This function returns an integer number which represents the time refinement factor of the parent of the current grid.
- real function agrif_parent_rhot ()

  Returns a real number which represents the time refinement factor of the parent of the current grid.
- integer function agrif_nbstepint ()

  This function returns an integer which is the time step of the current grid inside one time step of its parent grid.
- integer function agrif_parent_nbstepint ()

  It returns an integer which represents time step of the parent grid inside one time step of its parent grid.
- subroutine agrif_interpnearborderx ()

  Allows to interpolate (in the x direction) on a near (West) border of the current grid if this one has a common border with the root coarse grid.
- subroutine agrif_interpdistantborderx ()

  Allows to interpolate (in the x direction) on a distant (East) border of the current grid if this one has a common border with the root coarse grid.
- subroutine agrif_interpnearbordery ()

  Allows to interpolate (in the y direction) on a near (South) border of the current grid if this one has a common border with the root coarse grid.
- subroutine agrif_interpdistantbordery ()

  Allows to interpolate (in the y direction) on a distant (North) border of the current grid if this one has a common border with the root coarse grid.
- subroutine agrif_interpnearborderz ()

  Allows to interpolate (in the z direction) on a near (Down) border of the current grid if this one has a common border with the root coarse grid.
- subroutine agrif_interpdistantborderz ()

  Allows to interpolate (in the z direction) on a distant (Up) border of the current grid if this one has a common border with the root coarse grid.
- integer function agrif_parent_nb_step ()

  This function returns an integer which is the number of time step of the parent of the current grid.
- logical function agrif_root ()

  This function returns a logical which indicates if the current grid is the root grid or not.
- logical function agrif_grandmother ()

  This function returns a logical which indicates if the current grid is the grandmother grid or not.
- logical function agrif_parent_root ()

  This function indicates if the parent of the current grid is the root grid or not.
- integer function agrif_fixed ()

  This function returns the number of the current grid.
- integer function agrif_parent_fixed ()

  This function returns the number of the parent of the current grid.
- logical function agrif_is_fixed ()

  This function returns .TRUE. if the current grid is fixed even if we are on the root grid.
- logical function agrif_parent_is_fixed ()

  This function returns .TRUE. if the parent grid of the current grid is fixed even if we are on the root grid.
- character(3) function agrif_cfixed ()

  This function returns the number of the current grid as a string.
- character(3) function agrif_parent_cfixed ()

  This function returns the number of the parent of the current grid as a string.
- subroutine agrif_childgrid_to_parentgrid ()

  This subroutine makes #Agrif_Curgrid point on the parent grid of the current grid.
- subroutine agrif_parentgrid_to_childgrid ()

  This subroutine makes the pointer #Agrif_Curgrid point on the child grid after having called the subroutine ↩ :agrifdoc:moduserfunctions.F90::agrif_childgrid_to_parentgrid<Agrif_ChildGrid_↩ to_ParentGrid()>.

This function returns a logical which indicates whether the current grid and root grid have a common border on the western side of the first dimension of space (the x direction).

- logical function agrif_nearcommonbordery ()

  This function returns a logical which indicates whether the current grid and root grid have a common border on the southern side of the second dimension of space (the y direction).
- logical function agrif_nearcommonborderz ()

  This function returns a logical which indicates whether the current grid and root grid have a common border on the down side of the third dimension of space (the z direction).
- logical function agrif_distantcommonborderx ()

  This function returns a logical which indicates whether the current grid and root grid have a common border on the eastern side of the first dimension of space (the x direction).
- logical function agrif_distantcommonbordery ()

  This function returns a logical which indicates whether the current grid and root grid have a common border on the northern side of the second dimension of space (the y direction).
- logical function agrif_distantcommonborderz ()

  This function returns a logical which indicates whether the current grid and root grid have a common border on the up side of the third dimension of space (the z direction).
- integer function agrif_ix ()

  This function returns an integer which indicates the minimal position of the current grid in the x direction.
- integer function agrif_iy ()

  This function returns an integer which indicates the minimal position of the current grid in the y direction.
- integer function agrif_iz ()

  This function returns an integer which indicates the minimal position of the current grid in the z direction.
- integer function agrif_get_grid_id ()

  This function is used to get the grid id of the current grid.
- integer function agrif_get_parent_id ()

  This function is used to get the grid id of the parent grid of the current grid.
- real function agrif_rhox ()

  This function returns a real number which represents the space refinement factor of the current grid for the first dimension x.
- real function agrif_rhoy ()

  This function returns a real number which represents the space refinement factor of the current grid for the second dimension y.
- real function agrif_rhoz ()

  This function returns a real number which represents the space refinement factor of the current grid for the third dimension z.
- integer function agrif_nb_step ()

  This function returns an integer which represents the number of time steps of the current grid.
- integer function agrif_nb_fine_grids ()

  This function returns the number of fixed grids.

4.2.1 Detailed Description

Module Agrif_user_Functions.

This module defines procedures concerning current grids informations.

4.2.2 Function/Subroutine Documentation

4.2.2.1 real function agrif_user_functions::agrif_rel_dt ( )

Returns the time step of the current grid, relatively to the root grid (for which dt=1.).

In fact, it is a real number which could be define as the ratio between the current grid time step and the root grid time step.

Returns

time step

4.2.2.2 integer function agrif_user_functions::agrif_rel_idt ( )

This function returns an integer which is the time refinement factor of the current grid, relatively to the root grid (for which time step = 1).

It is the inverse of Agrif_rel_dt set to integer. Here, it is the ratio between the time step of the root grid and the time step of the current grid.

Returns

time step

4.2.2.3 integer function agrif_user_functions::agrif_irhot ( )

This function returns an integer number which is the time refinement factor of the current grid.

Returns

time refinement factor of the current grid

4.2.2.4 real function agrif_user_functions::agrif_rhot ( )

This function returns a real number which is the time refinement factor of the current grid.

Returns

time refinement factor of the current grid

4.2.2.5 integer function agrif_user_functions::agrif_parent_irhot ( )

This function returns an integer number which represents the time refinement factor of the parent of the current grid.

Returns

time refinement factor of the parent grid

4.2.2.6 real function agrif_user_functions::agrif_parent_rhot ( )

Returns a real number which represents the time refinement factor of the parent of the current grid.

Returns

time refinement factor of the parent grid

4.2.2.7 integer function agrif_user_functions::agrif_nbstepint ( )

This function returns an integer which is the time step of the current grid inside one time step of its parent grid.

It variates from zero to the time refinement factor of the current grid minus 1 (Agrif_Rhot - 1). It is useful for example for interpolation/update.

Returns

time step of the current grid

4.2.2.8 integer function agrif_user_functions::agrif_parent_nbstepint ( )

It returns an integer which represents time step of the parent grid inside one time step of its parent grid.

It variates from zero to the time refinement factor of the parent grid minus 1 (Agrif_Parent_Rhot - 1).

Returns

time step of the parent grid

4.2.2.9 subroutine agrif_user_functions::agrif_interpnearborderx ( )

Allows to interpolate (in the x direction) on a near (West) border of the current grid if this one has a common border with the root coarse grid.

4.2.2.10 subroutine agrif_user_functions::agrif_interpdistantborderx ( )

Allows to interpolate (in the x direction) on a distant (East) border of the current grid if this one has a common border with the root coarse grid.

4.2.2.11 subroutine agrif_user_functions::agrif_interpnearbordery ( )

Allows to interpolate (in the y direction) on a near (South) border of the current grid if this one has a common border with the root coarse grid.

4.2.2.12 subroutine agrif_user_functions::agrif_interpdistantbordery ( )

Allows to interpolate (in the y direction) on a distant (North) border of the current grid if this one has a common border with the root coarse grid.

4.2.2.13 subroutine agrif_user_functions::agrif_interpnearborderz ( )

Allows to interpolate (in the z direction) on a near (Down) border of the current grid if this one has a common border with the root coarse grid.

4.2.2.14 subroutine agrif_user_functions::agrif_interpdistantborderz ( )

Allows to interpolate (in the z direction) on a distant (Up) border of the current grid if this one has a common border with the root coarse grid.

4.2.2.15 integer function agrif_user_functions::agrif_parent_nb_step ( )

This function returns an integer which is the number of time step of the parent of the current grid.

In fact, it indicates the number of iterations done on the parent grid of the current grid since the start of the time integration of the grid.

Returns

time step of the parent of the current grid

4.2.2.16 logical function agrif_user_functions::agrif_root ( )

This function returns a logical which indicates if the current grid is the root grid or not.

4.2.2.17 logical function agrif_user_functions::agrif_grandmother ( )

This function returns a logical which indicates if the current grid is the grandmother grid or not.

4.2.2.18 logical function agrif_user_functions::agrif_parent_root ( )

This function indicates if the parent of the current grid is the root grid or not.

4.2.2.19 integer function agrif_user_functions::agrif_fixed ( )

This function returns the number of the current grid.

Indeed, it returns 0 for the coarse grid, a positive number for a fixed grid and -1 for a non fixed grid.

Returns

Result

4.2.2.20 integer function agrif_user_functions::agrif_parent_fixed ( )

This function returns the number of the parent of the current grid.

Indeed, it returns 0 for the coarse grid and non fixed parent grid and positive number for a fixed grid.

4.2.2.21 logical function agrif_user_functions::agrif_is_fixed ( )

This function returns .TRUE. if the current grid is fixed even if we are on the root grid.

4.2.2.22 logical function agrif_user_functions::agrif_parent_is_fixed ( )

This function returns .TRUE. if the parent grid of the current grid is fixed even if we are on the root grid.

4.2.2.23 character(3) function agrif_user_functions::agrif_cfixed ( )

This function returns the number of the current grid as a string.

4.2.2.24 character(3) function agrif_user_functions::agrif_parent_cfixed ( )

This function returns the number of the parent of the current grid as a string.

4.2.2.25 subroutine agrif_user_functions::agrif_childgrid_to_parentgrid ( )

This subroutine makes #Agrif_Curgrid point on the parent grid of the current grid.

4.2.2.26 subroutine agrif_user_functions::agrif_parentgrid_to_childgrid ( )

This subroutine makes the pointer #Agrif_Curgrid point on the child grid after having called the subroutine :agrifdoc:`moduserfunctions.F90::agrif_childgrid_to_parentgrid<Agrif_ChildGrid_to_ParentGrid()>`.

4.2.2.27 integer function agrif_user_functions::agrif_get_unit ( )

This function returns a unit not connected to any file.

4.2.2.28 subroutine agrif_user_functions::agrif_set_extra_boundary_cells (integer, intent(in) *nb_extra_cells*)

This subroutine is used to set the number of extra boundary cells which is take as input parameter.

Parameters

in	*nb_extra_cells*	number of extra cells

4.2.2.29 subroutine agrif_user_functions::agrif_set_efficiency (real, intent(in) *eff*)

This subroutine is used to set the efficiency which is taken as input parameter.

Parameters

in	*eff*	efficiency

4.2.2.30 subroutine agrif_user_functions::agrif_set_regridding (integer, intent(in) *regfreq*)

This subroutine is used to set the regridding frequency which is taken as input parameter.

Parameters

in	*regfreq*	Regridding frequency

4.2.2.31 subroutine agrif_user_functions::agrif_set_coeffref_x (integer, intent(in) *coeffref*)

This subroutine is used to set the space refinement coefficient in the x direction (first dimension).

Parameters

in	*coeffref*	Coefficient of spatial refinement

4.2.2.32 subroutine agrif_user_functions::agrif_set_coeffref_y (integer, intent(in) *coeffref*)

This subroutine is used to set the space refinement coefficient in the y direction (second dimension).

Parameters

in	*coeffref*	Coefficient of spatial refinement

4.2.2.33 subroutine agrif_user_functions::agrif_set_coeffref_z (integer, intent(in) *coeffref*)

This subroutine is used to set the space refinement coefficient in the z direction (third dimension).

Parameters

in	*coeffref*	Coefficient of spatial refinement

4.2.2.34 subroutine agrif_user_functions::agrif_set_coeffreft_x (integer, intent(in) *coeffref*)

This subroutine is used to set the time refinement factor in x direction (first dimension).

Parameters

in	*coeffref*	Coefficient of time refinement

4.2.2.35 subroutine agrif_user_functions::agrif_set_coeffreft_y (integer, intent(in) *coeffref*)

This subroutine is used to set the time refinement factor in y direction (second dimension).

Parameters

in	*coeffref*	Coefficient of time refinement

4.2.2.36 subroutine agrif_user_functions::agrif_set_coeffreft_z (integer, intent(in) *coeffref*)

This subroutine is used to set the time refinement coefficient in the z direction (third dimension).

4.2.2.37 subroutine agrif_user_functions::agrif_set_minwidth (integer, intent(in) *coefminwidth*)

This subroutine is used to set the minimum width which is taken as input parameter.

Parameters

in	*coefminwidth*	Coefficient of minwidth

4.2.2.38 subroutine agrif_user_functions::agrif_set_rafmax (integer, intent(in) *coefrafmax*)

This subroutine is used to set the maximal level in x,y and z-direction which is taken as input parameter.

Parameters

in	*coefrafmax*	maximal space refinement coefficient

4.2.2.39 subroutine agrif_user_functions::agrif_set_maskmaxsearch (integer, intent(in) *mymaxsearch*)

Parameters

in	*mymaxsearch*	input variables

4.2.2.40 integer function agrif_user_functions::agrif_level ( )

This function returns the level of the current grid in the grids hierarchy.

Returns

level of the current grid

4.2.2.41 integer function agrif_user_functions::agrif_maxlevel ( )

This function returns the maximum level of grid location in the hierarchy.

4.2.2.42 logical function agrif_user_functions::agrif_gridallocation_is_done ( )

This function is used to verify if the the allocation is done for the current grid.

4.2.2.43 real(kind=4) function agrif_user_functions::agrif_parent_real_4 (real(kind=4) *real_variable*)

This function returns the list of real grid variables of the parent of the current grid as single-precision real floating-point.

Parameters

real_variable	input real variable

4.2.2.44 real(kind=8) function agrif_user_functions::agrif_parent_real_8 (real(kind=8) *real_variable*)

This function returns the list of real grid variables of the parent of the current grid as double-precision real floating-point.

Parameters

real_variable	input real variable

4.2.2.45 integer function agrif_user_functions::agrif_parent_integer (integer *integer_variable*)

This function returns the list of integer grid variables of the parent of the current grid as integer.

4.2.2.46 character(len(character_variable)) function agrif_user_functions::agrif_parent_character (character(∗) *character_variable*)

This function returns the list of character grid variables of the parent of the current grid as character.

4.2.2.47 logical function agrif_user_functions::agrif_parent_logical (logical *logical_variable*)

This function returns the list of logical grid variables of the parent of the current grid as logical variables.

4.2.2.48 integer function agrif_user_functions::agrif_irhox ( )

This function is used to get the space refinement factor of the current grid in x direction (first dimension) set as integer.

4.2.2.49 integer function agrif_user_functions::agrif_irhoy ( )

This function is used to get the space refinement factor of the current grid in y direction (second dimension) set as integer.

4.2.2.50 integer function agrif_user_functions::agrif_irhoz ( )

This function is used to get the space refinement factor of the current grid in z direction (third dimension) set as integer.

4.2.2.51 logical function agrif_user_functions::agrif_nearcommonborderx ( )

This function returns a logical which indicates whether the current grid and root grid have a common border on the western side of the first dimension of space (the x direction).

4.2.2.52 logical function agrif_user_functions::agrif_nearcommonbordery ( )

This function returns a logical which indicates whether the current grid and root grid have a common border on the southern side of the second dimension of space (the y direction).

4.2.2.53 logical function agrif_user_functions::agrif_nearcommonborderz ( )

This function returns a logical which indicates whether the current grid and root grid have a common border on the down side of the third dimension of space (the z direction).

4.2.2.54 logical function agrif_user_functions::agrif_distantcommonborderx ( )

This function returns a logical which indicates whether the current grid and root grid have a common border on the eastern side of the first dimension of space (the x direction).

4.2.2.55 logical function agrif_user_functions::agrif_distantcommonbordery ( )

This function returns a logical which indicates whether the current grid and root grid have a common border on the northern side of the second dimension of space (the y direction).

4.2.2.56 logical function agrif_user_functions::agrif_distantcommonborderz ( )

This function returns a logical which indicates whether the current grid and root grid have a common border on the up side of the third dimension of space (the z direction).

4.2.2.57 integer function agrif_user_functions::agrif_ix ( )

This function returns an integer which indicates the minimal position of the current grid in the x direction.

Returns

minimal position

4.2.2.58 integer function agrif_user_functions::agrif_iy ( )

This function returns an integer which indicates the minimal position of the current grid in the y direction.

Returns

minimal position

4.2.2.59 integer function agrif_user_functions::agrif_iz ( )

This function returns an integer which indicates the minimal position of the current grid in the z direction.

Returns

minimal position

4.2.2.60 integer function agrif_user_functions::agrif_get_grid_id ( )

This function is used to get the grid id of the current grid.

Returns

grid id of the current grid

4.2.2.61 integer function agrif_user_functions::agrif_get_parent_id ( )

This function is used to get the grid id of the parent grid of the current grid.

Returns

grid id of the parent grid

4.2.2.62 real function agrif_user_functions::agrif_rhox ( )

This function returns a real number which represents the space refinement factor of the current grid for the first dimension x.

Returns

space refinement factor

4.2.2.63 real function agrif_user_functions::agrif_rhoy ( )

This function returns a real number which represents the space refinement factor of the current grid for the second dimension y.

Returns

space refinement factor

4.2.2.64 real function agrif_user_functions::agrif_rhoz ( )

This function returns a real number which represents the space refinement factor of the current grid for the third dimension z.

Returns

space refinement factor

4.2.2.65 integer function agrif_user_functions::agrif_nb_step ( )

This function returns an integer which represents the number of time steps of the current grid.

Returns

number of time steps

4.2.2.66 integer function agrif_user_functions::agrif_nb_fine_grids ( )

This function returns the number of fixed grids.

4.3 agrif_user_hierarchy Module Reference

Module Agrif_User_Hierarchy.

Functions/Subroutines

- subroutine agrif_step (procname)

  *This subroutine Creates the grid hierarchy and manages the time integration procedure. It should be called in the main program and calls the procname recursively for each grid. It also takes account the time refinement factor and Calls subroutines **Agrif_Regrid** and **Agrif_Integrate**.*
- subroutine agrif_step_child (procname)

  This subroutine calls 'procname' recursively for each grid and do not take account the time refinement factor. It Applies 'procname' to each grid of the hierarchy.
- subroutine agrif_step_childs (procname)

  The subroutine Agrif_Step_Childs calls 'procname' recursively on each child grid of the current coarse grid and take account the time refinement. So, it Applies 'procname' to each child grids of the current grid.
- subroutine agrif_regrid (procname)

  This subroutine creates the grid hierarchy from fixed grids and adaptive mesh refinement.
- recursive subroutine agrif_integrate_childgrids (procname)

  This subroutine manages the time integration of the grid hierarchy. It Calls the subroutine procname on each child grid of the current grid.
- subroutine agrif_init_grids (procname1, procname2)

  This subroutine initializes the root coarse grid pointed by #Agrif_Mygrid. It is called in the main program.
- subroutine agrif_deallocation

  This subroutine deallocates all data arrays.
- subroutine agrif_step_adj (procname)

  *This subroutine creates the grid hierarchy and manages the backward time integration procedure. It is called in the main program and calls subroutines **Agrif_Regrid()** and **Agrif_Integrate_adj()**.*
- subroutine agrif_step_child_adj (procname)

  This subroutine applies 'procname' to each grid of the hierarchy from Child to Parent.

4.3.1 Detailed Description

Module Agrif_User_Hierarchy.

This module contains procedures we used to manage the grid hierarchy. For example the procedures below are called in the main program :

- #Agrif_Init_Grids allows the initialization of the root coarse grid.
- #Agrif_Step allows the creation of the grid hierarchy and the management of the time integration.

4.3.2 Function/Subroutine Documentation

4.3.2.1 subroutine agrif_user_hierarchy::agrif_step (procedure(step_proc) *procname*)

This subroutine Creates the grid hierarchy and manages the time integration procedure. It should be called in the main program and calls the procname recursively for each grid. It also takes account the time refinement factor and Calls subroutines **Agrif_Regrid** and **Agrif_Integrate**.

Parameters

procname	subroutine to call on each grid

4.3.2.2 subroutine agrif_user_hierarchy::agrif_step_child (procedure(step_proc) *procname*)

This subroutine calls 'procname' recursively for each grid and do not take account the time refinement factor. It Applies 'procname' to each grid of the hierarchy.

Parameters

procname	subroutine to call on each grid

4.3.2.3 subroutine agrif_user_hierarchy::agrif_step_childs (procedure(step_proc) *procname*)

The subroutine Agrif_Step_Childs calls 'procname' recursively on each child grid of the current coarse grid and take account the time refinement. So, it Applies 'procname' to each child grids of the current grid.

Parameters

procname	subroutine to call on each grid

4.3.2.4 subroutine agrif_user_hierarchy::agrif_regrid (procedure(init_proc), optional *procname*)

This subroutine creates the grid hierarchy from fixed grids and adaptive mesh refinement.

Parameters

procname	Initialisation subroutine (Default: Agrif_InitValues)

4.3.2.5 recursive subroutine agrif_user_hierarchy::agrif_integrate_childgrids (procedure(step_proc) *procname*)

This subroutine manages the time integration of the grid hierarchy. It Calls the subroutine procname on each child grid of the current grid.

Parameters

procname	Subroutine to call on each grid

4.3.2.6 subroutine agrif_user_hierarchy::agrif_init_grids (procedure(typedef_proc), optional *procname1,* procedure(alloc_proc), optional *procname2*)

This subroutine initializes the root coarse grid pointed by #Agrif_Mygrid. It is called in the main program.

Parameters

procname1	(Default: Agrif_probdim_modtype_def)
procname2	(Default: Agrif_Allocationcalls)

4.3.2.7 subroutine agrif_user_hierarchy::agrif_deallocation ( )

This subroutine deallocates all data arrays.

4.3.2.8 subroutine agrif_user_hierarchy::agrif_step_adj (procedure(step_proc) *procname*)

This subroutine creates the grid hierarchy and manages the backward time integration procedure. It is called in the main program and calls subroutines **Agrif_Regrid()** and **Agrif_Integrate_adj()**.

Parameters

procname	Subroutine to call on each grid

4.3.2.9 subroutine agrif_user_hierarchy::agrif_step_child_adj (procedure(step_proc) *procname*)

This subroutine applies 'procname' to each grid of the hierarchy from Child to Parent.

Parameters

procname	Subroutine to call on each grid

4.4 agrif_user_interpolation Module Reference

Module Agrif_User_Interpolation.

Functions/Subroutines

- subroutine agrif_set_bc (tabvarsindic, bcinfsup, Interpolationshouldbemade)

  This subroutine is used to specify the variable we want to interpolate and the boundary location of the current grid we want to interpolate (boundary interpolation).
- subroutine agrif_set_interp (tabvarsindic, interp, interp1, interp2, interp3, interp4)

  This procedure specify the type of interpolation in case of interpolation within the domain. The index of the variable we want to interpolate is a profile defined with the subroutine Agrif_Declare_Variable.
- subroutine agrif_set_bcinterp (tabvarsindic, interp, interp1, interp2, interp3, interp4, interp11, interp12, interp21, interp22)

  This subroutine is used to indicate the type of interpolation in case of boundary interpolation. The index of the variable we want to interpolate is a profile defined with the subroutine Agrif_Declare_Variable.
- subroutine agrif_init_variable (tabvarsindic, procname)

  This subroutine is used for interpolation over the whole domain (within the domain and boundaries). It specifies the variable we want to interpolate and define through an input procedure (procname) what we do either on parent or child grid. The index of the variable we want to interpolate is a profile defined with the subroutine Agrif_Declare_Variable.
- subroutine agrif_bc_variable (tabvarsindic, procname, calledweight)

  This subroutine is used for boundary interpolation. It specifies the variable we want to interpolate and define through an input procedure (procname) what we do either on parent or child grid. The index of the variable we want to interpolate is a profile defined with the subroutine Agrif_Declare_Variable.
- subroutine agrif_interp_variable (tabvarsindic, procname)

  This subroutine is used for interpolation within the domain. It specifies the variable we want to interpolate and define through an input procedure (procname) what we do either on parent or child grid. The index of the variable we want to interpolate is a profile defined with the subroutine Agrif_Declare_Variable.

4.4.1 Detailed Description

Module Agrif_User_Interpolation.

This module contains procedures used by AGRIF users to do an interpolation either within the domain or at boundary or on the whole domain.

4.4.2 Function/Subroutine Documentation

4.4.2.1 subroutine agrif_user_interpolation::agrif_set_bc (integer, intent(in) *tabvarsindic,* integer, dimension(2), intent(in) *bcinfsup,* logical, intent(in), optional *Interpolationshouldbemade*)

This subroutine is used to specify the variable we want to interpolate and the boundary location of the current grid we want to interpolate (boundary interpolation).

Parameters

in	*tabvarsindic*	index of the variable in tabvars
in	*bcinfsup*	location
in	*interpolation-shouldbemade*	interpolation should be made

4.4.2.2 subroutine agrif_user_interpolation::agrif_set_interp (integer, intent(in) *tabvarsindic,* integer, intent(in), optional *interp,* integer, intent(in), optional *interp1,* integer, intent(in), optional *interp2,* integer, intent(in), optional *interp3,* integer, intent(in), optional *interp4*)

This procedure specify the type of interpolation in case of interpolation within the domain. The index of the variable we want to interpolate is a profile defined with the subroutine Agrif_Declare_Variable.

Parameters

in	*tabvarsindic*	index of the variable in tabvars
in	*interp4*	The type of interpolation

4.4.2.3 subroutine agrif_user_interpolation::agrif_set_bcinterp (integer, intent(in) *tabvarsindic,* integer, intent(in), optional *interp,* integer, intent(in), optional *interp1,* integer, intent(in), optional *interp2,* integer, intent(in), optional *interp3,* integer, intent(in), optional *interp4,* integer, intent(in), optional *interp11,* integer, intent(in), optional *interp12,* integer, intent(in), optional *interp21,* integer, intent(in), optional *interp22*)

This subroutine is used to indicate the type of interpolation in case of boundary interpolation. The index of the variable we want to interpolate is a profile defined with the subroutine Agrif_Declare_Variable.

Interp1 indicates the type of interpolation in the first dimension Interp21 indicates the first variable of the second dimension

Parameters

in	*tabvarsindic*	index of the variable in tabvars
in	*interp4*	type of interpolation
in	*interp22*	dimension we want to interpolate

4.4.2.4 subroutine agrif_user_interpolation::agrif_init_variable (integer, intent(in) *tabvarsindic,* procedure() *procname*)

This subroutine is used for interpolation over the whole domain (within the domain and boundaries). It specifies the variable we want to interpolate and define through an input procedure (procname) what we do either on parent or child grid. The index of the variable we want to interpolate is a profile defined with the subroutine Agrif_Declare_↩ Variable.

Parameters

in	*tabvarsindic*	index of the variable in tabvars
	procname	Data recovery procedure written by users

4.4.2.5 subroutine agrif_user_interpolation::agrif_bc_variable (integer, intent(in) *tabvarsindic,* procedure() *procname,* real, intent(in), optional *calledweight*)

This subroutine is used for boundary interpolation. It specifies the variable we want to interpolate and define through an input procedure (procname) what we do either on parent or child grid. The index of the variable we want to interpolate is a profile defined with the subroutine Agrif_Declare_Variable.

Parameters

in	*tabvarsindic*	index of the variable in tabvars
	procname	Data recovery procedure written by users
in	*calledweight*	weight of interpolation

4.4.2.6 subroutine agrif_user_interpolation::agrif_interp_variable (integer, intent(in) *tabvarsindic,* procedure() *procname*)

This subroutine is used for interpolation within the domain. It specifies the variable we want to interpolate and define through an input procedure (procname) what we do either on parent or child grid. The index of the variable we want to interpolate is a profile defined with the subroutine Agrif_Declare_Variable.

Parameters

in	*tabvarsindic*	index of the variable in tabvars
	procname	Data recovery procedure written by users

4.5 agrif_user_update Module Reference

Module Agrif_User_Update.

Functions/Subroutines

- subroutine agrif_set_updatetype (tabvarsindic, update, update1, update2, update3, update4, update5)

  This subroutine is used to specify the type of update we want to do. The index of the variable we want to interpolate is a profile defined with the subroutine Agrif_Declare_Variable.

- subroutine agrif_update_variable (tabvarsindic, procname, locupdate, locupdate1, locupdate2, locupdate3, locupdate4)

  This subroutine is used to update variables. When the location is not specify, the update is made within the domain of the reference grid (not at boundary). The index of the variable we want to interpolate is a profile defined with the subroutine Agrif_Declare_Variable. locupdate1 : specifies the location for the first dimension.

4.5.1 Detailed Description

Module Agrif_User_Update.

This module contains the main procedures used to update a variable.

4.5.2 Function/Subroutine Documentation

4.5.2.1 subroutine agrif_user_update::agrif_set_updatetype (integer, intent(in) *tabvarsindic,* integer, intent(in), optional *update,* integer, intent(in), optional *update1,* integer, intent(in), optional *update2,* integer, intent(in), optional *update3,* integer, intent(in), optional *update4,* integer, intent(in), optional *update5*)

This subroutine is used to specify the type of update we want to do. The index of the variable we want to interpolate is a profile defined with the subroutine Agrif_Declare_Variable.

Parameters

in	*tabvarsindic*	index of the variable in tabvars
in	*update5*	type of update

4.5.2.2 subroutine agrif_user_update::agrif_update_variable (integer, intent(in) *tabvarsindic,* procedure() *procname,* integer, dimension(2), intent(in), optional *locupdate,* integer, dimension(2), intent(in), optional *locupdate1,* integer, dimension(2), intent(in), optional *locupdate2,* integer, dimension(2), intent(in), optional *locupdate3,* integer, dimension(2), intent(in), optional *locupdate4*)

This subroutine is used to update variables. When the location is not specify, the update is made within the domain of the reference grid (not at boundary). The index of the variable we want to interpolate is a profile defined with the subroutine Agrif_Declare_Variable. locupdate1 : specifies the location for the first dimension.

Parameters

in	*tabvarsindic*	Indice of the variable in tabvars
	procname	Data recovery procedure written by users
in	*locupdate*	location to update

4.6 agrif_user_variables Module Reference

Module Agrif_User_Variables.

Functions/Subroutines

- subroutine agrif_declare_variable (posvar, firstpoint, raf, lb, ub, varid, torestore)

  This subroutine is used to declare a profile (new variable) which will be used for interpolation/update.

4.6.1 Detailed Description

Module Agrif_User_Variables.

This module contains a procedure which will help to define a new variable which will be used for interpolation/update.

4.6.2 Function/Subroutine Documentation

4.6.2.1 subroutine agrif_user_variables::agrif_declare_variable (integer, dimension(:), intent(in) *posvar,* integer, dimension(:), intent(in) *firstpoint,* character(1), dimension(:), intent(in) *raf,* integer, dimension(:), intent(in) *lb,* integer, dimension(:), intent(in) *ub,* integer, intent(out) *varid,* logical, intent(in), optional *torestore*)

This subroutine is used to declare a profile (new variable) which will be used for interpolation/update.

Parameters

in	*posvar*	position of the variable on the cell (1 for the border of the edge, 2 for the center)
in	*firstpoint*	index of the first point in the real domain
in	*raf*	Array indicating the type of dimension (space or not) for each of them
in	*lb*	Lower bounds of the array
in	*ub*	Upper bounds of the array
out	*varid*	Id number of the newly created variable
in	*torestore*	Indicates if the array restore is used

5 Data Type Documentation

5.1 agrif_save_forrestore Interface Reference

Public Member Functions

- subroutine agrif_save_forrestore0d (tabvarsindic0, tabvarsindic)
- subroutine agrif_save_forrestore2d (q, tabvarsindic)

  This function is used to restore a current grid variable (with the index tabvarsindic) to the input 2D-variable.
- subroutine agrif_save_forrestore3d (q, tabvarsindic)

  This function is used to restore a current grid variable (with the index tabvarsindic) to the input 3D-variable.
- subroutine agrif_save_forrestore4d (q, tabvarsindic)

  This function is used to restore a current grid variable (with the index tabvarsindic) to the input 4D-variable.

5.1.1 Detailed Description

5.1.2 Member Function/Subroutine Documentation

5.1.2.1 subroutine agrif_save_forrestore0d (integer, intent(in) *tabvarsindic0,* integer, intent(in) *tabvarsindic*)

Parameters

in	*tabvarsindic0*	index of the current grid variable
in	*tabvarsindic*	index of the varible which should be restored

5.1.2.2 subroutine agrif_save_forrestore2d (real, dimension(:,:), intent(in) *q,* integer, intent(in) *tabvarsindic*)

This function is used to restore a current grid variable (with the index tabvarsindic) to the input 2D-variable.

Parameters

in	*q*	input 2D-variable which should be saved
in	*tabvarsindic*	index of the current grid variable we want to restore

5.1.2.3 subroutine agrif_save_forrestore3d (real, dimension(:,:,:), intent(in) *q,* integer, intent(in) *tabvarsindic*)

This function is used to restore a current grid variable (with the index tabvarsindic) to the input 3D-variable.

Parameters

in	*q*	input 3D-variable which should be saved
in	*tabvarsindic*	index of the current grid variable we want to restore

5.1.2.4 subroutine agrif_save_forrestore4d (real, dimension(:,:,:,:), intent(in) *q,* integer, intent(in) *tabvarsindic*)

This function is used to restore a current grid variable (with the index tabvarsindic) to the input 4D-variable.

Parameters

in	*q*	input 4D-variable which should be saved
in	*tabvarsindic*	index of the current grid variable we want to restore

5.2 agrif_set_parent Interface Reference

Public Member Functions

- subroutine agrif_set_parent_int (integer_variable, value)

  This subroutine is used to set the type of the variable of the parent grid of the current grid as integer variable.
- subroutine agrif_set_parent_real4 (real_variable, value)

  This subroutine is used to set a real variable of the parent grid of the current grid as Single-precision real floating-point value.

- subroutine agrif_set_parent_real8 (real_variable, value)

  This subroutine is used to set a real variable of the parent grid of the current grid as Double-precision real floating-point value.

5.2.1 Detailed Description

5.2.2 Member Function/Subroutine Documentation

5.2.2.1 subroutine agrif_set_parent_int (integer, intent(in) *integer_variable,* integer, intent(in) *value*)

This subroutine is used to set the type of the variable of the parent grid of the current grid as integer variable.

Parameters

in	*integer_variable*	indice of the variable in tabvars
in	*value*	input value

5.2.2.2 subroutine agrif_set_parent_real4 (real(kind=4), intent(in) *real_variable,* real(kind=4), intent(in) *value*)

This subroutine is used to set a real variable of the parent grid of the current grid as Single-precision real floating-point value.

Parameters

in	*real_variable*	input variable
in	*value*	input value for the parent grid

5.2.2.3 subroutine agrif_set_parent_real8 (real(kind=8), intent(in) *real_variable,* real(kind=8), intent(in) *value*)

This subroutine is used to set a real variable of the parent grid of the current grid as Double-precision real floating-point value.

Parameters

in	*real_variable*	input variable
in	*value*	input value for the parent grid

5.3 agrif_parent Interface Reference

Public Member Functions

- real(kind=4) function agrif_parent_real_4 (real_variable)

  This function returns the list of real grid variables of the parent of the current grid as single-precision real floating-point.
- real(kind=8) function agrif_parent_real_8 (real_variable)

  This function returns the list of real grid variables of the parent of the current grid as double-precision real floating-point.
- integer function agrif_parent_integer (integer_variable)

  This function returns the list of integer grid variables of the parent of the current grid as integer.
- character(len(character_variable)) function agrif_parent_character (character_variable)

  This function returns the list of character grid variables of the parent of the current grid as character.
- logical function agrif_parent_logical (logical_variable)

  This function returns the list of logical grid variables of the parent of the current grid as logical variables.

5.3.1 Detailed Description

5.3.2 Member Function/Subroutine Documentation

5.3.2.1 real(kind=4) function agrif_parent_real_4 (real(kind=4) *real_variable*)

This function returns the list of real grid variables of the parent of the current grid as single-precision real floating-point.

Parameters

real_variable	input real variable

5.3.2.2 real(kind=8) function agrif_parent_real_8 (real(kind=8) *real_variable*)

This function returns the list of real grid variables of the parent of the current grid as double-precision real floating-point.

Parameters

real_variable	input real variable

5.3.2.3 integer function agrif_parent_integer (integer *integer_variable*)

This function returns the list of integer grid variables of the parent of the current grid as integer.

5.3.2.4 character(len(character_variable)) function agrif_parent_character (character(∗) *character_variable*)

This function returns the list of character grid variables of the parent of the current grid as character.

5.3.2.5 logical function agrif_parent_logical (logical *logical_variable*)

This function returns the list of logical grid variables of the parent of the current grid as logical variables.