Common Multi-dimensional Remapping Software CoR (Common Remap) V1.0
User Reference Manual

Li Liu, Guangwen Yang, Bin Wang
liuli-cess@tsinghua.edu.cn
Ministry of Education Key Laboratory for Earth System Modeling,
Center for Earth System Science (CESS),
Tsinghua University, Beijing, China

1st, March, 2013
Contents

Chapter 1  Introduction ................................................................................................. 1
  1.1  Background ................................................................................................................ 1
  1.2  Target functions of CoR ............................................................................................ 1
  1.3  How to use CoR ......................................................................................................... 2
  1.4  Main functions in CoR V1.0 ..................................................................................... 3
  1.5  Organization of this manual ....................................................................................... 3

Chapter 2  Grid definition ............................................................................................ 3
  2.1  Definition of 1D grid ................................................................................................. 5
  2.2  Definition of multi-dimensional grid ......................................................................... 5
  2.3  Definition of partial grid .......................................................................................... 6
  2.4  Setting of grid center values .................................................................................... 7
  2.4.1  Function read_field .............................................................................................. 7
  2.4.2  Function fspan ....................................................................................................... 7
  2.5  Setting of grid vertex values .................................................................................... 8
  2.6  Setting of grid mask values ...................................................................................... 9
  2.6.1  Function read_field .............................................................................................. 9
  2.6.2  Function extract_mask .......................................................................................... 9
  2.6.3  Function calc_ocn_mask ........................................................................................ 10
  2.7  Vertical coordinate transformation ........................................................................ 11
  2.8  Adding a grid area ................................................................................................... 12
  2.8.1  Registering area name .......................................................................................... 12
  2.8.2  Setting boundaries of grid area ............................................................................ 12
  2.9  Examples .................................................................................................................. 13
    2.9.1  Example 1 ............................................................................................................ 13
    2.9.2  Example 2 ............................................................................................................ 13

Chapter 3  Multi-dimensional remapping ................................................................. 14
  3.1  Definition of remapping operator ............................................................................. 14
    3.1.1  Definition of remapping operator ....................................................................... 14
    3.1.2  Parameter setting of remapping operator ........................................................... 15
  3.2  Definition of remapping scheme ............................................................................. 15
  3.3  Compute remapping weights ................................................................................... 16
  3.4  Remapping calculation ............................................................................................. 16
    3.4.1  Remapping with remapping scheme .................................................................. 16
    3.4.2  Remapping with remapping weights .................................................................. 17

Chapter 4  IO file operation ....................................................................................... 17
  4.1  Open NETCDF file .................................................................................................. 17
  4.2  Reading field data ................................................................................................... 18
  4.3  Allocating field data ............................................................................................... 18
  4.4  Writing field data .................................................................................................... 19
  4.5  Writing remapping weights ..................................................................................... 19
  4.6  Reading remapping weights .................................................................................... 20

Chapter 5  Installation and Usage .............................................................................. 21
5.1 Installation ............................................................ 21
5.2 Running CoR ............................................................ 22
Chapter 1  Introduction

1.1  Background

Remapping, as known as interpolation, which transforms field data between different grids, is a fundamental numerical computation in earth simulation and weather forecasting. It has been widely used in pre-processing, model execution, and post-processing. Generally, the grid of field data consists of at most four dimensions, including longitude, latitude, vertical levels and time frame, where longitude and latitude constitutes horizontal grid. With fast development of earth system models (ESM) and weather forecasting models, there are various requirements for remapping.

First, there are various kinds of horizontal grids, such as latitude-longitude, tripolar grid, cube-sphere grid, icosahedral grid, etc. Moreover, there are unstructured grids corresponding to observation stations. Regarding to vertical grids, there are Z coordinate, sigma coordinate, hybrid coordinate, etc.

Second, there are various remapping algorithms, such as cubic spline, inverse distance weight, bilinear, bicubic, conservation, etc. With the fast development of science and technology, there will be more remapping algorithms used in future.

Third, there are various field data with different numbers of dimensions. For example, the field data for atmosphere-ocean coupling is 2D data on horizontal grid; in pre-processing and post-processing, there are 2D, 3D even 4D data needing to be remapped, where 3D grid can be horizontal grid plus vertical levels or horizontal grid plus time frames; in model nesting, each lateral boundary condition is a 2D data.

1.2  Target functions of CoR

Considering the above requirements for remapping, we design and implement CoR with the following target functions:

1)  Design and implement a grid representation to uniformly manage all kinds of
grids with different structures and different number of dimensions. The grid representation can reflect the relation between grids. For example, one 2D grid is the horizontal grid or lateral grid of a 3D grid.

2) Design and implement various kinds of remapping algorithms with different dimension number. Each 2D remapping algorithm supports all kinds of horizontal grids. Several remapping algorithms can be cascaded together to generate a multi-dimensional remapping scheme.

1.3 How to use CoR

As shown in Figure 1, CoR is designed to support three usage modes as follows:

1) Scripts. Users can remap data or generate remap weights with scripts. The execution of CoR consists two phases. The first phase check the correctness of scripts and the data used for remapping, which does not produce results but reports errors. The second phase produces results and reports the progress of execution.

2) API for remapping algorithms. CoR provides API to enable users implement their own remapping algorithms.

3) API for FORTRAN code. CoR provides API to enable users use the
remapping functions of CoR in their FORTRAN code.

In current version CoR V1.0, only the usage mode of scripts is provided. The fundamental grammars of CoR scripts include two following aspects:

1) Implicit variable definitions.
2) Constants such as integers and floating points are marked by quotes.

1.4 Main functions in CoR V1.0

As the first version, CoR V1.0 does not implement all target functions of CoR. It achieves the following main functions:

1) The uniform grid representation.
2) Multiple remapping algorithms, including inverse distance weight, bilinear, first-order conservation, linear and spline, which can be cascaded to generate multi-dimensional remapping scheme.
3) Only file format of NETCDF is supported.
4) Two phases of CoR execution is implemented to accelerate checking the correctness of scripts and data.

1.5 Organization of this manual

The following context of this manual focuses on how to write scripts of CoR, including the common definition of grids, multi-dimensional remapping and IO operations.

Chapter 2 Grid definition

As a common remapping software, the grid definition of CoR satisfies the following targets:

1) Uniformly define one-dimensional grid and multi-dimensional grid with the concept of coordinate system;
2) Uniformly define horizontal grids with different structures, such as
latitude-longitude grids, GAUSS reduced grids, tri-polar grids, cube-spherical grids, icosahedral grids, unstructured grids, and so on.  

3) Uniformly define vertical grids, such as Z grid and sigma grid.  
4) Uniformly define global grid and regional grid, such as the lateral boundary grid.  
5) Record the relation between grids.  
6) Check the correctness in grid definition and usage.  

For earth system models, CoR support four kinds of coordinates, including longitude (lon), latitude (lat), vertical levels (lev) and time frames (time), and a grid is an arrangement of coordinates. To define a grid, users should not only specify the arrangement of coordinates, but also set grid values when the grid will be used for remapping. Grid values include center values, vertex values and mask. Center values record the coordinates of center point of each cell. Vertex values record the coordinates of vertexes of each cell. Masks mark the existence of each cell. For example, masks of value 0 on the ocean grid are always used to specify land cell. Center values are always read from data file. The center values of 1D grid can be generated by \textit{span} function. Additionally, the center values of vertical grids can be generated through grid transformation. For example, user can transform center values from sigma grid to z grid. Vertex values are always read from data file. Vertex values can also be automatically generated according to the center values. However, the automatically generated vertex values cannot be used for conservation remapping algorithms. Mask values are always read from data file. They can also be extracted from model data through specifying missing value. Moreover, mask values can also be computed from terrain data.  

The definition of regional grid is based on global grid. Global grid is the grid with certain grid size and grid values. A regional grid consists of multiple regions on global grid. Therefore, users should set the region of regional grid and cannot set the grid values of regional grid.  

The following context will introduce each function of CoR for grid definition, where red words specify the parameters of function and blue words specify the
function name.

2.1 Definition of 1D grid

Function call: \texttt{grid\_1D = new\_1D\_grid(coord\_label, unit, cyclic, grid\_size)}

Description of function: define 1D grid.

Description of parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Data type</th>
<th>Necessary or optional</th>
<th>Other info.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid_1D</td>
<td>The name of 1D grid</td>
<td>Grid object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Coord_label</td>
<td>The name of coordinate, must be one of “lon”, “lat”, “lev” and “time”</td>
<td>String</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Unit</td>
<td>The unit of coordinate, for “lon” and “lat”, the unit must be “degrees” or “radians”</td>
<td>String</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Cyclic</td>
<td>A sign to mark whether the 1D grid is cyclic, must be “cyclic” or “acyclic”</td>
<td>String</td>
<td>Optional: necessary when coordinate is “lon”</td>
<td></td>
</tr>
<tr>
<td>grid_size</td>
<td>The size of 1D grid</td>
<td>Integer</td>
<td>Optional: necessary when coordinate is “lev” or “time”</td>
<td>grid_size is a constant given by users or dimlen info. read from file</td>
</tr>
</tbody>
</table>

Constraints of usage:

1) Grid grid\_1D has not been defined in the script;

2) All input parameters are correct.

2.2 Definition of multi-dimensional grid

Function call: \texttt{grid\_MD = combine\_grids(subgrid\_1, subgrid\_2, …, subgrid\_n, grid\_size)}

Description of function: define a multi-dimensional grid with an arrangement of sub grids. The order of sub grids in the input parameters determines the arrangement of sub grids.
Description of parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Data type</th>
<th>Necessary or optional</th>
<th>Other info.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid_MD</td>
<td>The name of multi-dimensional grid</td>
<td>Grid object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>subgrid_i</td>
<td>The name of sub grid, which can be 1D grid or multi-dimensional grid</td>
<td>Grid object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>grid_size</td>
<td>The size (number of cells) of the multi-dimensional grid</td>
<td>Integer</td>
<td>Optional: necessary when the sizes of sub grids are unknown</td>
<td>grid_size is a constant given by users or dimlen info, read from file</td>
</tr>
</tbody>
</table>

Constraints of usage:

1) Grid grid_MD has not been defined in the script;
2) Sub grids Subgrid_1~subgrid_n all have been defined;
3) All input parameters are correct;
4) Sub grids subgrid_1~subgrid_n do not have common coordinate;
5) When the size of one sub grid is known, all sizes of all sub grids are known;
6) When the sizes of all sub grids are known and parameter grid_size is given by users, grid_size must equals to the multiple of the sizes of all sub grids.

### 2.3 Definition of partial grid

Function call: \texttt{grid\_partial = new\_partial\_grid(grid\_global)}

Description of function: define a partial grid based on a global grid.

Description of parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Data type</th>
<th>Necessary or optional</th>
<th>Other info.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid_partial</td>
<td>The name of partial grid</td>
<td>Grid object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>grid_global</td>
<td>The name of global grid</td>
<td>Grid object</td>
<td>Necessary</td>
<td></td>
</tr>
</tbody>
</table>

Constraints of usage:

1) Grid grid\_partial has not been defined;
2) Grid\_global is a global grid which has been defined.
2.4 Setting of grid center values

2.4.1 Function read_field

Function call: \texttt{grid\%center\%coord\_label = read\_field(grid\_file, field\_name)}

Description of function: set grid center values on one coordinate through reading data from IO file.

Description of parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Data type</th>
<th>Necessary or optional</th>
<th>Other info.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid</td>
<td>The name of grid</td>
<td>Grid object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Center</td>
<td>The label of grid center values</td>
<td>Reserved word</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>coord_label</td>
<td>The name of coordinate</td>
<td>String name</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>grid_file</td>
<td>IO file which records grid data</td>
<td>IO object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Field_name</td>
<td>Variable name in IO file</td>
<td>String</td>
<td>Necessary</td>
<td></td>
</tr>
</tbody>
</table>

Constraints of usage:
1) Grid has been defined;
2) Grid is a global grid;
3) The size of grid is known;
4) Coord\_label is a coordinate of grid;
5) The grid values on coord\_label has not been set before;
6) Grid\_file is an IO object whose IO file has been opened for reading;
7) Field\_name is a variable in grid\_file, whose size is the same as the size of grid.

2.4.2 Function fspan

Function call: \texttt{grid\%center\%coord\_label = fspan(start, end, num\_points)}

Description of function: set grid center values of 1D grid through fspan. Fspan generates arithmetic progression of size num\_points between start and end.

Description of parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Data type</th>
<th>Necessary or optional</th>
<th>Other info.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid</td>
<td>The name of grid</td>
<td>Grid object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Center</td>
<td>The label of grid center values</td>
<td>Reserved word</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td>Description</td>
<td>Data type</td>
<td>Necessary or optional</td>
<td>Other info.</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------</td>
<td>----------------</td>
<td>-----------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>coord_label</td>
<td>The name of coordinate</td>
<td>String name</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Start</td>
<td>The first value of arithmetic progression</td>
<td>Floating point</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>End</td>
<td>The last value of arithmetic progression</td>
<td>Floating point</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Num_points</td>
<td>The size of arithmetic progression</td>
<td>Integer</td>
<td>Necessary</td>
<td></td>
</tr>
</tbody>
</table>

Constraints of usage:

1) Grid has been defined;
2) Grid is a 1D grid;
3) Grid is a global grid;
4) The size of grid is known;
5) Coord_label is the unique coordinate of grid;
6) The grid values on coord_label has not been set before;
7) num_points is the same as the size of grid.

### 2.5 Setting of grid vertex values

Function call: `grid%vertex%coord_label = read_field(grid_file, field_name, num_vertex)`

Description of function: set grid vertex values on one coordinate through reading data from IO file.

Description of parameters:

1) Grid has been defined;
2) Grid is a global grid;
3) The size of grid is known;
4) Coord_label is a coordinate of grid;
5) The center values on coord_label of grid has been set;
6) The vertex values on coord_label has not been set before;

7) Grid_file is an IO object whose IO file has been opened for reading;

8) Num_vertex is a positive integer;

9) Field_name is a variable in grid_file, whose size equals grid_size*Num_vertex, where grid_size is the size of grid.

2.6 Setting of grid mask values

2.6.1 Function read_field

Function call: `grid%mask = read_field(grid_file, field_name)`

Description of function: set grid mask values through reading data from IO file.

Description of parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Data type</th>
<th>Necessary or optional</th>
<th>Other info.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid</td>
<td>The name of grid</td>
<td>Grid object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Mask</td>
<td>The label of grid mask values</td>
<td>Reserved word</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>grid_file</td>
<td>IO file which records grid data</td>
<td>IO object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Field_name</td>
<td>Variable name in IO file</td>
<td>String</td>
<td>Necessary</td>
<td></td>
</tr>
</tbody>
</table>

Constraints of usage:

1) Grid has been defined;

2) Grid is a global grid;

3) The size of grid is known;

4) The mask values of grid has not been set before;

5) Grid_file is an IO object whose IO file has been opened for reading;

6) Field_name is a variable in grid_file, whose size is the same as the size of grid.

2.6.2 Function extract_mask

Function call: `grid%mask = extract_mask(data_field, min_value, max_value)`

Description of function: extract grid mask values through specifying missing value. min_value and max_value are boundaries of missing value. For grid cell with value between min_value and max_value, its mask will be set to false.
Description of parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Data type</th>
<th>Necessary or optional</th>
<th>Other info</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid</td>
<td>The name of grid</td>
<td>Grid object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Mask</td>
<td>The label of grid mask values</td>
<td>Reserved word</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>data_field</td>
<td>Field data</td>
<td>Field data object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Min_value</td>
<td>Lower boundary of missing value</td>
<td>Floating point</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>max_value</td>
<td>Upper boundary of missing value</td>
<td>Floating point</td>
<td>Necessary</td>
<td></td>
</tr>
</tbody>
</table>

Constraints of usage:
1) Grid has been defined;
2) Grid is a global grid;
3) The size of grid is known;
4) The mask values of grid has not been set before;
5) data_field has been defined and its values has been set;
6) min_value <= max_value.

### 2.6.3 Function `calc_ocn_mask`

Function call: `ocn_grid%mask = calc_ocn_mask(topo_data, unit_trans_ratio)`

Description of function: calculate 2D or 3D ocean grid mask values from terrain data `topo_data`. `unit_trans_ratio` is used for unit transformation. For example, when the unit of `topo_data` is decimeter and the unit of vertical grid of ocean grid is meter, `unit_trans_ratio` is 10.

Description of parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Data type</th>
<th>Necessary or optional</th>
<th>Other info</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ocn_grid</td>
<td>Name of 2D/3D ocean grid</td>
<td>Grid object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Mask</td>
<td>The label of grid mask values</td>
<td>Reserved word</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Topo_data</td>
<td>Field data: terrain data</td>
<td>Field data object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>unit_trans_ratio</td>
<td>Ratio for unit transformation</td>
<td>Floating point</td>
<td>Necessary</td>
<td></td>
</tr>
</tbody>
</table>

Constraints of usage:
1) Ocn_grid has been defined;
2) Ocn_grid is a global grid;
3) The size of ocn_grid is known;
4) The mask values of ocn_grid has not been set before;

5) When ocn_grid is a 2D grid, it is a horizontal grid; when ocn_grid is a 3D grid, it is a spatial grid and its vertical sub grid is a Z grid;

6) The grid center values and vertex values of ocn_grid have been set.

7) The grid of topo_data is a horizontal grid, whose resolution is much higher than the horizontal resolution of ocn_grid.

2.7 Vertical coordinate transformation

Function call: \texttt{grid\%center\%lev = lev\_coord\_from\_sigma(field\_bot, top\_value, sigma\_grid, sorting\_order)}

Description of function: set center values on vertical coordinate though transforming sigma coordinate to Z coordinate.

Description of parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Data type</th>
<th>Necessary or optional</th>
<th>Other info</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid</td>
<td>Name of grid</td>
<td>Grid object</td>
<td>必要</td>
<td></td>
</tr>
<tr>
<td>Center</td>
<td>Label of grid center values</td>
<td>Reserve word</td>
<td>必要</td>
<td></td>
</tr>
<tr>
<td>lev</td>
<td>Label of vertical coordinate</td>
<td>Reserve word</td>
<td>必要</td>
<td></td>
</tr>
<tr>
<td>field_bot</td>
<td>Coordinate values at the bottom</td>
<td>Field data object</td>
<td>必要</td>
<td></td>
</tr>
<tr>
<td>Top_value</td>
<td>Coordinate value at the top</td>
<td>Floating point</td>
<td>必要</td>
<td></td>
</tr>
<tr>
<td>Sigma_grid</td>
<td>Vertical grid of sigma coordinate</td>
<td>Grid object</td>
<td>必要</td>
<td></td>
</tr>
<tr>
<td>Sorting_order</td>
<td>Sorting order of vertical coordinate values: “ascending” or “descending”</td>
<td>String</td>
<td>必要</td>
<td></td>
</tr>
</tbody>
</table>

Constraints of usage:

(1) Grid is a defined 3D spatial grid;

(2) Grid is a global grid;

(3) The grid values on vertical coordinate of grid have not been set before;

(4) The grid of field\_bot is a horizontal grid and sub grid of grid, and the data in field\_bot are known;

(5) Sigma\_grid is a 1D vertical grid, whose size is the same as the number of vertical levels in grid; center values of sigma\_grid have been set, which are between 0 and 1, and sorted in ascending or descending order.
2.8 Adding a grid area

2.8.1 Registering area name

Function call: \texttt{add\_grid\_area(grid\_partial, area\_name)}

Description of function: register an area name \texttt{area\_name} for \texttt{grid\_partial}

Description of parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Data type</th>
<th>Necessary or optional</th>
<th>Other info</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid_partial</td>
<td>Name of partial grid</td>
<td>Grid object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Area_name</td>
<td>Name of grid area</td>
<td>String</td>
<td>Necessary</td>
<td></td>
</tr>
</tbody>
</table>

Constraints of usage:

1) \texttt{Grid\_partial} is a defined partial grid;
2) \texttt{Grid\_partial} does not have the area named \texttt{area\_name}.

2.8.2 Setting boundaries of grid area

Function call: \texttt{add\_area\_bound(grid\_partial, area\_name, subgrid, bound\_type, min\_value, max\_value)}

Description of function: set boundaries of grid area. Two types of boundary are supported.

Description of parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Data type</th>
<th>Necessary or optional</th>
<th>Other info</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid_partial</td>
<td>Name of partial grid</td>
<td>Grid object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Area_name</td>
<td>Name of grid area</td>
<td>String</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>subgrid</td>
<td>The sub grid the boundaries corresponding to</td>
<td>Grid object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Bound_type</td>
<td>Types of boundary: “value” (coordinate values) or “index” (index of grid cells)</td>
<td>String</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Min_value</td>
<td>Lower boundary</td>
<td>Integer or floating point</td>
<td>Necessary</td>
<td>If bound_type is “index”, the data type of boundary values are integer, otherwise, boundary values are Integer or floating point</td>
</tr>
<tr>
<td>Max_value</td>
<td>Upper boundary</td>
<td>Integer or floating point</td>
<td>Necessary</td>
<td></td>
</tr>
</tbody>
</table>
Constraints of usage:

1) Grid_partial is a defined partial grid;
2) Area_name is a defined grid area of Grid_partial;
3) Subgrid is a sub grid of grid_partial;
4) Bound_type is “value” or “index”;
5) When bound_type is “index”, the size of subgrid is known; when bound_type is “value”, subgrid is a 1D grid;
6) Min_value and max_value are consistent to bound_type, and min_value<=max_value.

2.9 Examples

2.9.1 Example 1

(1) grid_lat = new_1D_grid("lat", "degrees", 60)
(2) grid_lon = new_1D_grid("lon", "degrees", "cyclic")
(3) grid_lonlat = combine_grids(grid_lon, grid_lat)

Explanation: the third statement is wrong because the size of grid_lon is known but the size of grid_lat is unknown.

2.9.2 Example 2

(1) grid_lat = new_1D_grid("lat", "degrees", "60")
(2) grid_lon = new_1D_grid("lon", "degrees", "cyclic", "128")
(3) grid_lonlat = combine_grids(grid_lon, grid_lat)
(4) grid_lev = new_1D_grid("lev", "levels", "26")
(5) grid_lonlatlev = combine_grids(grid_lon, grid_lat, grid_lev)
(6) grid_lat%center%lat = fspan("-90", "90", "61")
(7) grid_lon%center%lon = fspan("2.8125", "360", "128")
(8) grid_lonlatlev%center%lev = read_field(file1, "lev_value")
(9) grid_lonlat%center%lon = read_field(file1, "lon_value")
(10) grid_lev%center%lev = fspan("0", "5000", "26")

Explanation: the 6th statement is wrong because the size of arithmetic progression is 61, which is different from the size of grid_lat. The 9th statement is wrong, because
the center values on “lon” of grid grid_lonlat have been set in the 7th statement. Note that, grid_lon and grid_lonlat share the same coordinate “lon”. The 10th statement is wrong, because the center values on “lev” of grid grid_lev have been set in the 8th statement. Note that grid_lonlatlev and grid_lev share the same coordinate “lev”.

Chapter 3  Multi-dimensional remapping

3.1  Definition of remapping operator

3.1.1  Definition of remapping operator

Function call: remap_optr = new_operator(optr_name, grid_src, grid_dst)

Description of function: define remapping operator according to remapping algorithm name, source grid and destination grid.

Description of parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Data type</th>
<th>Necessary or optional</th>
<th>Other info.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remap_optr</td>
<td>Name of remapping operator</td>
<td>Remapping operator object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Optr_name</td>
<td>Name of remapping algorithm,</td>
<td>String</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td></td>
<td>currently including “bilinear”, “conserv_2D”, “distwgt”, “linear” and “spline_1D”</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>grid_src</td>
<td>Name of source grid</td>
<td>Grid object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>grid_dst</td>
<td>Name of destination grid</td>
<td>Grid object</td>
<td>Necessary</td>
<td></td>
</tr>
</tbody>
</table>

Constraints of usage:

1) Remap_optr is not defined before;

2) Optr_name is one of “bilinear”, “conserv_2D”, “distwgt”, “linear”, and “spline_1D”

3) grid_src and grid_dst has been defined, and their grid size are known, and their coordinate systems are consistent with each other and the remapping algorithm.
3.1.2 Parameter setting of remapping operator

Function call: `set_parameter(remap_optr, para_name, para_value)`

Description of function: set one parameter of remap operator

Description of parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Data type</th>
<th>Necessary or optional</th>
<th>Other info.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remap_optr</td>
<td>Name of remapping operator</td>
<td>Remapping operator object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Para_name</td>
<td>Name of parameter</td>
<td>String</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Para_value</td>
<td>Value of parameter</td>
<td>String</td>
<td>Necessary</td>
<td></td>
</tr>
</tbody>
</table>

Constraints of usage:

1) Remap_optr is a defined remap operator;
2) Para_name is a parameter of the remapping algorithm corresponding to Remap_optr;
3) Para_value matches the parameter of para_name.

3.2 Definition of remapping scheme

Function call: `remap_scheme = combine_operators(optr_1, optr_2, …, optr_n)`

Description of function: define remapping scheme through combining multiple remapping operators.

Description of parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Data type</th>
<th>Necessary or optional</th>
<th>Other info.</th>
</tr>
</thead>
<tbody>
<tr>
<td>remap_scheme</td>
<td>Name of remapping scheme</td>
<td>Remapping scheme object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Optr_1, …,</td>
<td>Name of one of remapping</td>
<td>Remapping operator object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>optr_n</td>
<td>operator</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Constraints of usage:

1) Remap_scheme has not been defined;
2) optr_1~optr_n are remapping operators which have been defined before;
3) Before remapping field data with one remapping operator, the source grid of the remapping operator is a sub grid of the field data grid.
### 3.3 Compute remapping weights

Function call: \( \text{remap}_\text{wgts} = \text{calc}_\text{remap}_\text{wgts}(\text{remap}_\text{scheme}, \text{grid}_\text{src}, \text{grid}_\text{dst}) \)

Description of function: compute the remapping weights of \( \text{remap}_\text{scheme} \) when remapping field data from grid \( \text{grid}_\text{src} \) to \( \text{grid}_\text{dst} \).

Description of parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Data type</th>
<th>Necessary or optional</th>
<th>Other info.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{remap}_\text{wgts} )</td>
<td>Name of remap weights</td>
<td>Remapping weight object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>( \text{Remap}_\text{scheme} )</td>
<td>Name of remapping scheme</td>
<td>Remapping scheme object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>( \text{Grid}_\text{src} )</td>
<td>Source grid of ( \text{remap}_\text{wgts} )</td>
<td>Grid object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>( \text{Grid}_\text{dst} )</td>
<td>Destination grid of ( \text{remap}_\text{wgts} )</td>
<td>Grid object</td>
<td>Necessary</td>
<td></td>
</tr>
</tbody>
</table>

Constraints of usage:

1) \( \text{remap}_\text{wgts} \) has not been defined;
2) \( \text{remap}_\text{scheme} \), \( \text{grid}_\text{src} \) and \( \text{grid}_\text{dst} \) have been defined before;
3) \( \text{grid}_\text{src} \) and \( \text{grid}_\text{dst} \) are consistent with \( \text{remap}_\text{scheme} \).

### 3.4 Remapping calculation

Description of function: remap field data from source grid to destination grid with remapping scheme or remapping weights.

#### 3.4.1 Remapping with remapping scheme

Function call: \( \text{remap}(\text{remap}_\text{scheme}, \text{field}_\text{src}, \text{field}_\text{dst}) \)

Description of parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Data type</th>
<th>Necessary or optional</th>
<th>Other info.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{Remap}_\text{scheme} )</td>
<td>Name of remapping scheme</td>
<td>Remapping scheme object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>( \text{field}_\text{src} )</td>
<td>Name of field data on source grid</td>
<td>Field data object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>( \text{field}_\text{dst} )</td>
<td>Name of field data on destination grid</td>
<td>Field data object</td>
<td>Necessary</td>
<td></td>
</tr>
</tbody>
</table>

Constraints of usage:
1) remap_scheme has been defined;
2) field_src and field_dst have been defined, and the data field_src have been set or read from IO file;
3) The grids of field_src and field_dst are consistent with remapping scheme.

### 3.4.2 Remapping with remapping weights

Function call: `remap(remap_wgts, data_src, data_dst)`

Description of parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Data type</th>
<th>Necessary or optional</th>
<th>Other info</th>
</tr>
</thead>
<tbody>
<tr>
<td>remap_wgts</td>
<td>Name of remap weights</td>
<td>Remapping weight object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>field_src</td>
<td>Name of field data on source grid</td>
<td>Field data object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>field_dst</td>
<td>Name of field data on destination grid</td>
<td>Field data object</td>
<td>Necessary</td>
<td></td>
</tr>
</tbody>
</table>

Constraints of usage:
1) remap_wgts has been defined and calculated;
2) field_src and field_dst have been defined, and the data field_src have been set or read from IO file;
3) The grids of field_src and field_dst are the source and destination grids of remap_wgts respectively.

### Chapter 4 IO file operation

#### 4.1 Open NETCDF file

Function call: `nc_file = add_nc_file(file_name, open_pattern)`

Description of function: Open NETCDF file file_name with pattern Open_pattern.

Description of parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Data type</th>
<th>Necessary or optional</th>
<th>Other info</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nc_file</td>
<td>Name of the file object</td>
<td>File object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>File_name</td>
<td>NETCDF file name</td>
<td>String</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------</td>
<td>--------</td>
<td>-----------</td>
<td></td>
</tr>
<tr>
<td>Open_pattern</td>
<td>Pattern of opening file, only “r” and “w” currently.</td>
<td>String</td>
<td>Necessary</td>
<td></td>
</tr>
</tbody>
</table>

Constraints of usage:

1) `Nc_file` has not been defined.
2) `File_name` is an existing NETCDF file.
3) `Open_pattern` is “r” or “w”.

### 4.2 Reading field data

Function call: `field_data = read_field(field_grid, io_file, field_name)`

Description of function: read the data of `field_data` from `io_file` with the variable name `field_name`. `field_grid` is the grid `field_data` located on.

Description of parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Data type</th>
<th>Necessary or optional</th>
<th>Other info.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field_data</td>
<td>Name of field data in script</td>
<td>Field data object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Field_grid</td>
<td>Grid of field_data</td>
<td>Grid object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Io_file</td>
<td>IO file object</td>
<td>IO file object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Field_name</td>
<td>Name of field in IO file</td>
<td>String</td>
<td>Necessary</td>
<td></td>
</tr>
</tbody>
</table>

Constraints of usage:

1) `Field_data` has not been defined;
2) `Field_grid` is a grid which has been defined, and the grid size is known;
3) `Io_file` is an IO file object, which opens a NETCDF file with pattern “r”;
4) `Field_name` is a variable in `io_file` and its size is the same as the size of `field_grid`.

### 4.3 Allocating field data

Function call: `field_data = alloc_field(field_grid, field_io_name)`

Description of function: allocate the memory of `field_data` according to `field_grid`.

Description of parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Data type</th>
<th>Necessary or optional</th>
<th>Other info.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field_data</td>
<td>Name of field data in script</td>
<td>Grid object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Field_grid</td>
<td>Grid of field_data</td>
<td>IO file object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------</td>
<td>----------------</td>
<td>-----------</td>
<td></td>
</tr>
<tr>
<td>Field_io_name</td>
<td>Name of field data in IO file</td>
<td>String</td>
<td>Optional: when Field_io_name is not set, the name of field data in IO file will be inherited from the original physical quantity.</td>
<td></td>
</tr>
</tbody>
</table>

1) Field_data has not been defined;
2) Field_grid has been defined and its grid size is known.

### 4.4 Writing field data

Function call: `write_field(io_file, field_data, write_grid_name)`

Description of function: write field_data into io_file. When write_grid_name is set, the name of grid values in io_file will be extended with the grid name.

Description of parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Data type</th>
<th>Necessary or optional</th>
<th>Other info.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Io_file</td>
<td>IO file object</td>
<td>IO file object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Field_data</td>
<td>Field data to be written</td>
<td>Field data object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>Write_grid_name</td>
<td>Mark whether extends the name of grid values in io_file with the grid name. If mark, its value must be &quot;write_grid_name&quot;</td>
<td>String</td>
<td>Optional</td>
<td></td>
</tr>
</tbody>
</table>

1) Io_file is an IO file object, which opens a NETCDF file with pattern “w”;
2) field_data have been defined, and its values have been set or computed through remapping.

### 4.5 Writing remapping weights

Function call: `write_remap_wgts(io_file, remap_wgts, format_str)`

Description of function: write remap_wgts into io_file. Format_str specifies the format of write remap_wgts, including “SCRIP” and “C-Coupler”. “SCRIP” is the format that SCRIP software writes the weights of horizontal remapping into NETCDF file. The format can be used by NCAR couplers, OASIS couplers, NCL, CDO and
C-Coupler in China. “C-Coupler” is a specific format of multi-dimensional remapping weights in unformatted binary file for C-Coupler.

Description of parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Data type</th>
<th>Necessary or optional</th>
<th>Other info.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Io_file</td>
<td>IO file object</td>
<td>IO file object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>remap_wgts</td>
<td>Name of remap weights</td>
<td>Remapping weight object</td>
<td>Necessary</td>
<td></td>
</tr>
<tr>
<td>format_str</td>
<td>The format of writing remap_wgts, must be one of “SCRIP” and “C-Coupler”</td>
<td>String</td>
<td>Necessary</td>
<td></td>
</tr>
</tbody>
</table>

Constraints of usage:

1) Io_file is an IO file object, which opens a IO file with pattern “w”;
2) Remap_wgts is the remap weights which have been defined and calculated;
3) When format_str is “SCRIP”, io_file corresponds to a NETCDF file, there are only one horizontal remapping algorithm in the remapping scheme corresponding to remap_wgts, and the grid vertex values have been set by users;
4) When format_str is “C-Coupler”, io_file corresponds to a binary file.

### 4.6 Reading remapping weights

Function call: `remap_wgts = read_remap_wgts(remap_scheme, field_grid_src, field_grid_dst, io_file, format_str)`

Description of function: read remapping weights from io_file, and check the consistence between remapping weights, remapping scheme and grids. format_str specifies the format of remapping weights in IO file, including “SCRIP” format and “C-Coupler” format. “SCRIP” format is the format of SCRIP software writing 2D remapping weights in NETCDF file, which can be used by NCAR couplers, OASIS couplers, NCL, CDO and our C-Coupler. “C-Coupler” is a format of multi-dimensional remapping weights in binary file for our C-coupler.

Description of parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Data type</th>
<th>Necessary or optional</th>
<th>Other info.</th>
</tr>
</thead>
</table>


### Constraints of usage:

1) `Io_file` has been opened and can be written;
2) Remapping scheme `remap_scheme` has been defined;
3) Grid `Field_grid_src` and `field_grid_dst` have been defined;
4) When `format_str` is “SCRIP”, `io_file` corresponds to NETCDF file and when `format_str` is “C-Coupler”, `io_file` corresponds to a binary file;
5) The remapping weights in `Io_file` are consistent with `Remap_scheme`, `Field_grid_src` and `Field_grid_dst`.

### Chapter 5  Installation and Usage

#### 5.1 Installation

The package of CoR V1.0 includes three directories: `src`, `run`, and `example`. `Src` directory stores the source code and compiling script `make.sh` (shell script) of CoR V1.0. The compiling of CoR V1.0 requires Intel C++ compiler and NETCDF library. The compiling command is “./make.sh” under the `src` directory. After compiling, there is an executable named “CoR” under the `run` directory.

Example directory includes the examples for how to use CoR. Sub directory `example/scripts` gives examples of script of CoR and sub directory `example/data` stores the input and output data of these CoR scripts. Users can write their own CoR
scripts based on these examples.

5.2 Running CoR

We suggest users run CoR under the run directory, with the command "./CoR script_name", for example, "./CoR ../example/scripts/template.scripts.MRI-CGCM3_to_360x181". We suggest users to set stack size to be unlimited, with command "ulimit -s unlimited".