ILSIO User's Guide

Takashi Arakawa

December 18, 2022

Chapter 1

Intruduction

1.1 Overview of ILSIO

As the name implies, an IO component ILSIO is a component that substitutes for a file IO of other components construct ILS. ILSIO is one component of ILS executed in parallel with other components. The outline of functions are as follows;

1. Read data from files and pass it to model components after grid remapping and time interpolation.
2. Receive data from model components and output to file after grid remapping.

Figure 1.1 shows the overview of ILSIO. The external data at the left of the figure is read into the IO and passed to the model component after grid remapping and time interpolation. The data passed from the model component is output to a file after grid remapping.

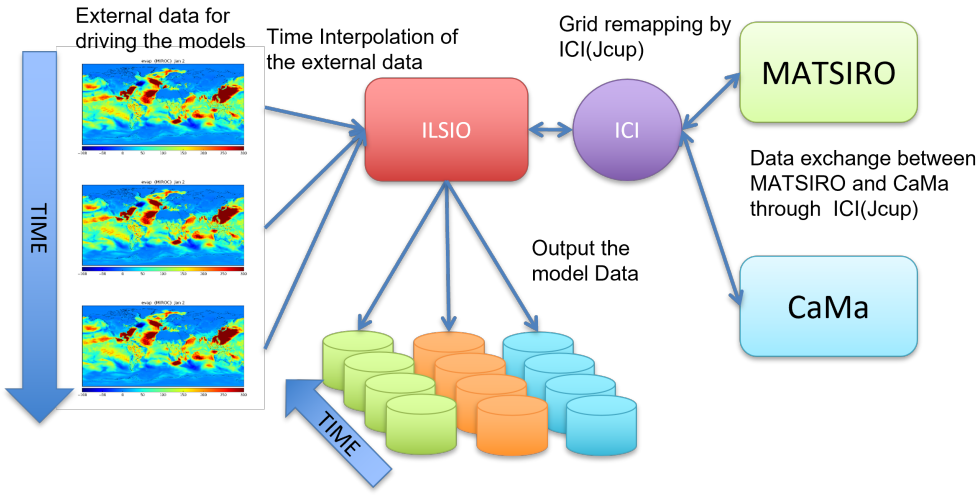

Figure 1.1: Overview of ILSIO

1.2 Concept of ILSIO

ILSIO is executed in parallel with other components and acts as an IO for other components. As a result, the time required for the IO of the model component can be reduced, and an improvement in

overall throughput can be expected. In addition, ILSIO not only simply performs file input/output, but also has a grid conversion function and a time interpolation function (when inputting). With these functions, it is possible to input various external data without modifying the code of each model component and output data of various grid systems and resolutions as needed.

1.3 Supported File Format

ILSIO supports two types of file format Binary file and NetCDF file. The current version of ILSIO uses the NetCDF format as its standard format, and the binary format is secondary option. Therefore, this user's guide describes the input/output of NetCDF files.

Chapter 2

File Format

2.1 NetCDF file format

The format of the netCDF dataset is such that three-dimensional (nx, ny, nz) or two-dimensional (nx, ny) data are arranged in time series. An example of the header of a NetCDF dataset is shown in List.2.1. In this example, two-dimensional data of $nx = 720$ and $ny = 360$ are recorded in chronological order. When output the data, variable attributes and global attributes are set in the Json style configuration file described later.

List 2.1: Example of NetCDF File Header

```
netcdf Rainf {
dimensions:
        longitude = 720 ;
        latitude = 360 ;
        time = UNLIMITED ; // (2 currently)
        bnds = 2 ;
variables:
        float longitude(longitude) ;
                longitude:long_name = "" ;
                longitude:standard_name = "longitude" ;
                longitude:alma_name = "N/A" ;
                longitude:amip_name = "N/A" ;
                longitude:units = "degrees_east" ;
                longitude:positive = "" ;
                longitude:comment = "" ;
        float latitude(latitude) ;
                latitude:long_name = "" ;
                latitude:standard_name = "latitude" ;
                latitude:alma_name = "N/A" ;
                latitude:amip_name = "N/A" ;
                latitude:units = "degrees_north" ;
                latitude:positive = "" ;
                latitude:comment = "" ;
        double time(time) ;
                time:units = "seconds since 2000-1-1 0:0:0" ;
                time:bounds = "time_bnds" ;
        double time_bnds(time, bnds) ;
        float Rainf_check(time, latitude, longitude) ;
                Rainf_check:long_name = "Rainfall Flux" ;
                Rainf_check:standard_name = "rainfall_flux" ;
```

```
31                Rainf_check:alma_name = "Rainf" ;
32                Rainf_check:amip_name = "prra" ;
33                Rainf_check:units = "kg m-2 s-1" ;
34                Rainf_check:type = "real" ;
35                Rainf_check:positive = "down" ;
36                Rainf_check:comment = "Rainfall rate" ;
37                Rainf_check:_FillValue = -999.f ;
38
39  // global attributes:
40                :title = "ILS output for GSWP3" ;
41                :missing_value = -999.f ;
42                :int_missing_value = "-999" ;
43                :contact = "Tomoko Nitta (t-nitta@iis.u-tokyo.ac.jp)" ;
44                :institution = "Institute of Industrial Science, The University of
                      Tokyo" ;
45                :source = "ILS" ;
46                :conventions = "CF-1.7 ALMA3" ;
47  }
```

2.2 Binary file format

The data at each time has a three-dimensional size of (nx, ny, nz). These value (nx, ny, nz) are given in the configuration file described later. Since the data is single precision, the record length of one record is $nx * ny * nz * 4$. Endian is set by compile option when the I/O mode is Gather mode. In MPI IO mode, the endian is checked inside the program and conversion is performed if necessary.

Chapter 3

Time Interpolation and Data Time

3.1 Time Interpolation

ILSIO has the function to interpolate input data in time and send it to the model. The subroutine for time interpolation is subroutine time_interpolation in file ilsio_time_interpolation.f90. This subroutine takes two data values (data1, data2), times (dtime1, dtime2) and time boundss (stime1, etime1, stime2, etime2) as input arguments and calculates the value at a certain time(ctime) as shown in Figure 3.1. If the data dose not have time bounds, stime1=etime1=dtime1 and stime2=etime2=dtime2.

By default, linear interpolation is performed as in the following equation. In this formula, the time boundss (stime1, etime1, stime2, etime2) are not used in the calculation. These values are used if the need arises in the future.

$$cdata = \frac{data1 * (dtime2 - ctime) + data2 * (ctime - dtime1)}{dtime2 - dtime1} \tag{3.1}$$

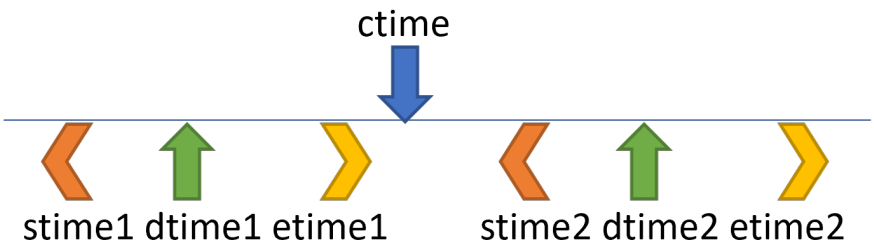

Figure 3.1: Time definision for time interpolation

3.2 Data Time Handling in ILSIO

This section explains how data time is handled internally in ILSIO. ILSIO acquires the necessary time data in two phases, a phase in which the data time before and after a certain model time is searched, and a phase in which the data at that time is read. Figure 3.2 shows these phases for data without time bounds. In the search phase, data times before and after a certain model time are searched, and in the Read phase, the data at that time is read. Since data does not have a time bounds, the data time itself is the target time for search and reading.

On the other hand, when the data has a time bounds, as shown in Figure 3.2, the search targets the time at the beginning of the time bounds (stime), and dtime, stime etime are used to time interpolation..

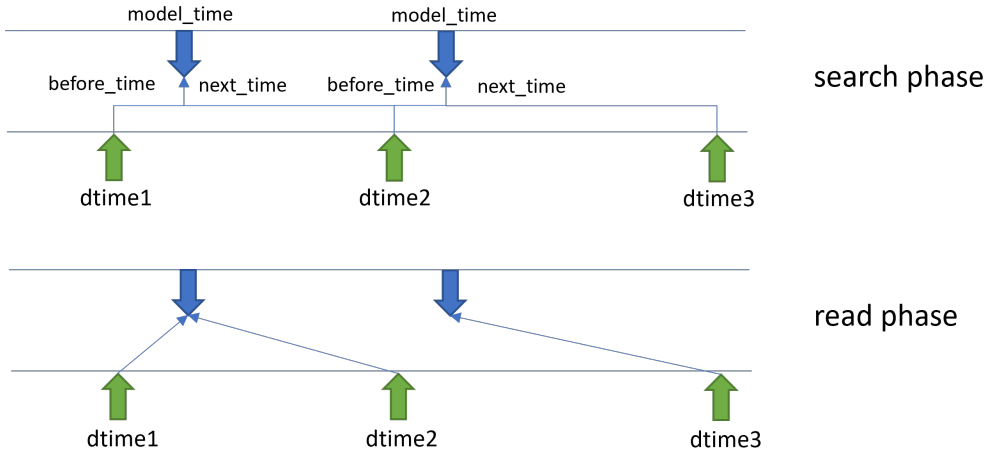

Figure 3.2: Data time handling without time bounds

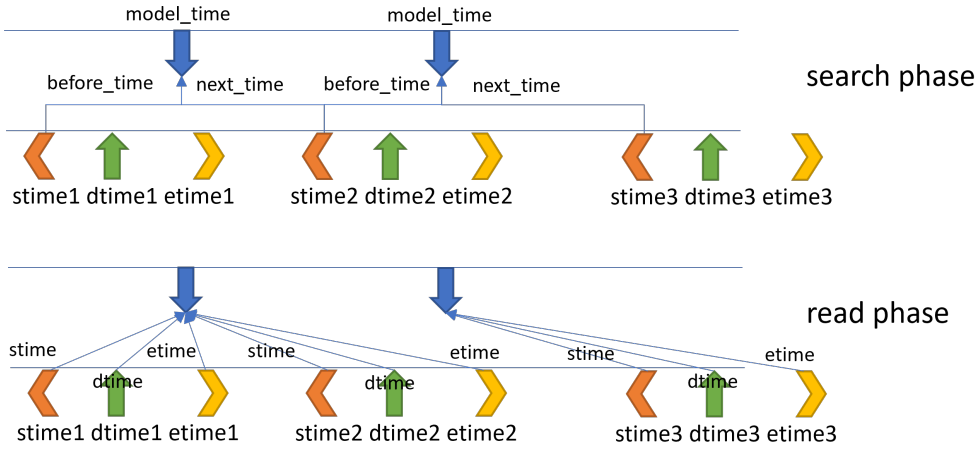

Figure 3.3: Data time handling with time bounds

3.3 Model Time and Data Time

As described in the previous section, the time interpolation subroutine requires two time data as input. For this reason, there are certain limitations between the model integration time and the time of the data. Data can be broadly classified into cyclic and non-cyclic data, with different limitations for each class. For non-cyclic data, the limitations also differ depending on the presence or absence of a time bounds. Therefore, in the following sections, we will explain how data time is handled for each of these categories. For simplicity of explanation, the following conditions are set.

- The data input time interval on the model side is 1 hour

- Integration start time is 2000/01/01-00:00:00
- Integration end time is 2000/01/01-23:00:00
- Data time interval is 1 hour
- Data start time is the same as model start time

3.3.1 Non-cyclic data without time bounds

Non-cyclic data without time bounds is the most basic setting. Model time and data time is as shown in Figure 3.3.1. In this case, final time step (T=23) requires next step data (T=24 or T=0 of next day). Therefore, first step data of the next day must be included in the data file. If the data time is advanced by 30 minutes(Figure 3.3.1), an error will result because there is no data 1 in the first step.

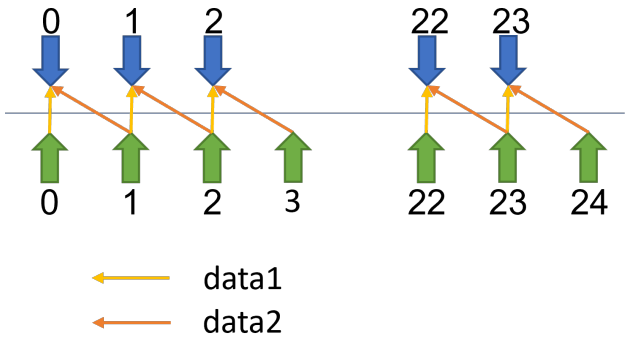

Figure 3.4: Model Time and Data Time of Non-cyclic and without range

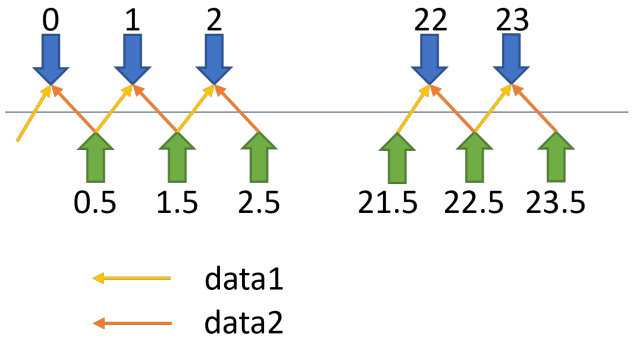

Figure 3.5: Data time advanced case

3.3.2 Non-cyclic data with time bounds

Non-cyclic data with time bounds is a common case in practice. Model time and data time is as shown in Figure 3.3.2. In this case, as described in the previous section, data time seach targets the beginning time of the time bounds. As the same as witiout time bounds case, final time step

(T=23) requires next step data (T=24 or T=0 of next day). Therefore, first step data of the next day must be included in the data file. On the other hand, when the data time is advanced, no error occurs because stime is used for data time search.

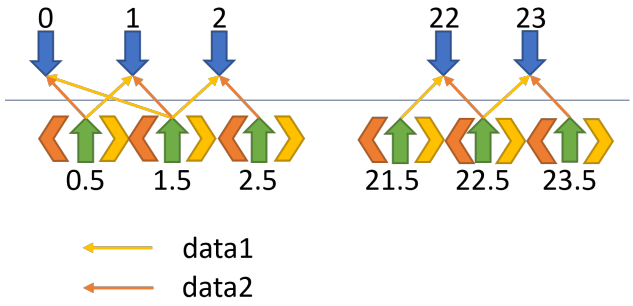

Figure 3.6: Model Time and Data Time of Non-cyclic and with range

3.3.3 Cyclic data

When the data is cyclic (is_clim = 1 is set in ilsio.conf), no error occurs because any model time has data times before and after it.

3.3.4 Summary of limitations on data times

The limitations on data times described above are summarized in the Table 3.1, where model_start is the first step time of the model, model_end is the last step time of the model, data_start is the first time of the data, data_end is the last time of the data, stime_start is the beginning of the time bounds in the first data, and stime_end is the time of the beginning of the time bounds in the last data.

Table 3.1: limitations on data times

condision	start time	end time
Non-Cyclic without time bounds	model_start $\geqq$ data_start	model_end $<$ data_end
Non-Cyclic with time bounds	model_start $\geqq$ stime_start	model_end $<$ stime_end
Cyclic	no limitation	no limitation

Chapter 4

Configuration

4.1 Configuration overview

The name of the configuration file is ilsio.conf by default. This file consists of five defferent sections, io_config, binary_file_config, nam_mio_hgrid, num_mio_vgrid and num_mio. The following sections describe these configuration sections.

4.2 io_config

"io_config" is a section that specifies the overall setting of ILSIO. The setting example of io_config is listed in List 4.1 and the description is summarized in Table 4.1.

"io_name" is a name of ILSIO. Unless there is a specific reason, users do not need to change the default name. "start_time" and "end_time" represent the start and end times of the model calculation in year/month/day/hour/minute/second.

"calender" sets the type of calendar used in the calculation. The possible settings are three types, "CALENDAR_NORMAL", "CALENDAR_NOLEAPYEAR", and "CALENDAR_30360". "CALENDAR_NORMAL" is the normal calendar system, "CALENDAR_NOLEAPYEAR" is the calendar system without leap years, and "CALENDAR_30360" is the calendar system with one month fixed at 30 days.

"default_io_mode" sets the default file I/O mode. Set any of "MPI", "GATHER", and "NETCDF", and perform input/output in MPI IO mode, GATHER mode, and NetCDF mode.

"write_file_extension" sets the extension of write files.

"FILL_VALUE" sets the missing value of the data.

"namelist_ici_file" sets the configuration file name read by data exchange library ICI.

"json_file_name" sets the JSON style file name utilized for NetCDF output. The style of the file will be described in the later section.

"out_history" sets the flag whether or not to output the history file. If 0, no file is output.

List 4.1: io_config section

```
1 &io_config
2   io_name = "ILSIO",
3   start_time = 2000,1,1,0,0,0
4   end_time = 2000,1,3,0,0,0
5   calendar = "CALENDAR_NORMAL"
6   default_io_mode = "NETCDF"
7   write_file_extension = ".nc"
8   FILL_VALUE = -999.d0
```

```
  namelist_ici_file = "coupling.conf"
  json_file_name = "ILS.json"
  out_history = 0
&end
```

Table 4.1: io_config section

name	possible values	description
io_name	string (default "ILSIO")	IO component name
start_time	array of integer	caclulation start time (yyyy/mo/dd/hh/mm/ss)
end_time	array of integer	calculation end time (yyyy/mo/dd/hh/mm/ss)
calendar	"CALENDAR_NORMAL", "CALENDAR_NOLEAPYEAR", "CALENDAR_30360"	calendar type
default_io_mode	"MPI", "GATHER", "NETCDF"	default file io mode)
write_file_extention	string	file extension of write data
fill_value	double presition	value of missing data
namelist_ici_file	string	configuration file name of ICI
json_file_name	string	attribute setting file for netcdf output
out_history	0 or 1	flag to output the history file

4.3 binary_file_config

"binary_file_config" is a section that specifies the setting of binary file input. The setting example of binary_file_config is listed in List 4.2 and the description is summarized in Table 4.2.

"binary_data_start" defines the start time of the file. "binary_data_intvl" defines the data interval. "binary_intvl_type" sets the unit of the data inteval.

List 4.2: binary_file_config section

```
&binary_file_config
  binary_data_start = 2000,1,1,0,0,0
  binary_data_intvl = 10800
  binary_intvl_type = "SEC"
&end
```

Table 4.2: binary_file_config section

name	possible values	description
binary_data_start	array of integer	file data start time
binary_data_intvl	integer	data inteval of binary data
binary_intvl_type	"SEC", "MIN", "HUR", "DAY", "MON"	unit of data inteval

4.4 nam_mio_hgrid

"num_mio_hgrid" defines horizontal grid. The example is listed in List 4.3 and the description is summarized in Table 4.3. Here, grird_name, nx, ny must be set in all cases, and the other elements

are valid only for netcdf output. "xname" is a name of X dimension and "yname" is a name of Y dimension in netcdf file. "xjson" and "yjson" are variable name defined in ILSIO.json file. "xfile" and "yfile" are file name in which the grid point positions are recorded.

List 4.3: num_mio_hgrid section

```
1  &nam_mio_hgrid
2    grid_name = "io_grid1",
3    nx = 720,
4    ny = 360,
5  &end
6
7  &nam_mio_hgrid
8    grid_name = "io_grid2",
9    nx = 720,
10   ny = 360,
11 &end
12
13 &nam_mio_hgrid
14   grid_name = "io_grid3",
15   nx = 720, xname = "longitude", xjson = "lon1", xfile = "../../ILS_data/test/bnd/
         lon1.bin",
16   ny = 360, yname = "latitude", yjson = "lat1", yfile = "../../ILS_data/test/bnd/
         lat1.bin", /
17 &end
```

Table 4.3: num_mio_hgrid section

name	possible values	description
grid_name	string	name of horizontal grid
nx	integer	grid size of X axis
xname	string	name of X axis (valid only netcdf output)
xjson	string	name of X axis in JSON file (valid only netcdf output)
xfile	string	X axis position file name (valid only netcdf output)
ny	integer	grid size of y direction
yname	string	name of Y axis (valid only netcdf output)
yjson	string	name of Y axis in JSON file (valid only netcdf output)
yfile	string	Y axis position file name (valid only netcdf output)

4.5 nam_mio_vgrid

As the same as num_mio_hgrid, "num_mio_vgrid" defines vertical grid. The example is listed in List 4.4 and the description is summarized in Table 4.4. Here, grird_name, nz must be set in all cases, and the other elements are valid only for netcdf output. "zname" is a name of Z dimension in netcdf file. "zjson" is a variable name defined in ILSIO.json file. "zfile" is a file name in which the grid point positions are recorded.

List 4.4: num_mio_vgrid section

```
1    &nam_mio_vgrid
2    grid_name = "v_grid1",
3    nz = 6, zname = "depth", zjson = "depth",
```

```
4  /
5
6  &nam_mio_vgrid
7    grid_name = "v_grid2",
8    nz = 3, zname = "num3", zjson = "num3",
9  /
10
11 &nam_mio_vgrid
12   grid_name = "v_grid3",
13   nz = 6, zname = "num6", zjson = "num6",
14 /
15
16 &nam_mio_vgrid
17   grid_name = "v_grid4",
18   nz =5, zname = "num5", zjson = "num5",
19 /
```

Table 4.4: num_mio_vgrid section

name	possible values	description
grid_name	string	name of vertical grid
nz	integer	grid size of Z axis
zname	string	name of Z axis (valid only netcdf output)
zjson	string	name of Z axis in JSON file (valid only netcdf output)
zfile	string	Z axis position file name (valid only netcdf output)

4.6 nam_mio

"nam_mio" section sets various settings relating to data directory, grid information, initial data, input data, and output data.

The example of "nam_mio" setting is listed in List 4.5 and the description are summarized in Table 4.5. "data_dir" defines data directory applied to the data described below. "hgrid_name" defines horizontal grid name. The name must be defined in nam_mio_hgrid section. "vgrid_name" defines vertical grid name. The name must be defined in nam_mio_vgrid section. When the data is 2 dimensional, "vgrid_name" must be "". "init_file_name" is file name of initial data. "read_file_name" is file name of input data. "write_file_name" is file name of output data. "data_name" is a name of the data. This name is utilized for exchange data name in configuration file "coupling.conf" and input/output data name of NetCDF file.

For data I/O, at least above setting is nesessarry, and below settings are optional. "io_mode" difines individual file IO mode. "BIN" is Binary I/O and "NETCDF" is NetCDF I/O. This setting overrides default_io_mode defined in io_config session. "extension" defines file name extension of output file. This setting overrides write_file_extension in io_config session. "intvl_type" and "intvl" define time interval of the data. This settings are valid only binary input. "data_byte" defines data precision. 4 means single precision (default) and 8 is double precision. "is_clim" is a flag that the data is cyclic or not. This flag is valid only input data.

List 4.5: nam_mio section

```
1    &nam_mio data_dir = "../../ILS_data/test/bnd/", /
2    &nam_mio hgrid_name = "io_grid1",
```

```
3           vgrid_name = "", /
4
5    &nam_mio init_file_name = "lon.bin", data_name = "lon", io_mode="BIN" /
6    &nam_mio init_file_name = "lat.bin", data_name = "lat", io_mode="BIN" /
7
8    &nam_mio data_dir = "../../ILS_data/test/frc/", /
9    &nam_mio hgrid_name = "io_grid2",
10          vgrid_name = "", /
11
12   &nam_mio read_file_name = "GSWP3.BC.SWdown.3hrMap.ILS.2000.nc", data_name = "
         SWdown", /
13   &nam_mio read_file_name = "GSWP3.BC.LWdown.3hrMap.ILS.2000.nc", data_name = "
         LWdown", /
14
15   &nam_mio hgrid_name = "io_grid4",
16         vgrid_name = "", /
17   &nam_mio read_file_name = "Dust_deposition.nc", data_name = "Dust_deposition",
         is_clim=1, /
18   &nam_mio read_file_name = "BC_deposition.nc", data_name = "BC_deposition", is_clim
         =1, /
19
20   &nam_mio hgrid_name = "io_grid1",
21          vgrid_name = "", /
22   &nam_mio read_file_name = "leaf_area_index.nc", data_name = "leaf_area_index",
         is_clim=1, /
23
24   &nam_mio data_dir = "./", /
25   &nam_mio hgrid_name = "io_grid3",
26          vgrid_name = "", /
27
28   &nam_mio write_file_name = "Wind_E", data_name = "Wind_E_check", /
29   &nam_mio write_file_name = "Wind_N", data_name = "Wind_N_check", /
30   &nam_mio write_file_name = "Tair", data_name = "Tair_check", /
```

Table 4.5: nam_mio section

name	possible values	description
data_dir	string	data directory name
hgrid_name	string	horizontal grid name
vgrid_name	string	vertical grid name
init_file_name	string	file name of initial data
read_file_name	string	file name of read (input) data
write_file_name	string	file name of write (output) data
data_name	string	data name
io_mode	”BIN” or ”NETCDF”	file IO format (Binary or NetCDF)
extension	string	file extension (valid only output)
intvl_type	”SEC”, ”MIN”, ”HUR”, ”DAY”, ”MON”	unit of data interval (valid only binary input)
intvl	integer	data timme interval (valid only binary input)
data_byte	4 or 8	data byte (default is 4 : single precision)
is_clim	integer 0 or 1	flag if the data is cyclic or not (valid only input)

Chapter 5

NetCDF file Output

5.1 Overview of this secrion

In this section, how to implement your code and how to set the configuration to output new data to the netcdf file will be described. It is assumed that the model to generate the data is already coupled to the ILS system. When you need to know how to couple the model, please refer the coupling manual.

5.2 Code Implementation

At first, the code for sending the data to ILSIO must be implemented in the model. As mentioned the overview, the model coupling code is assumed to be already implemented. Therefore the code need to send the data to ILSIO is only "call ici_put_data". The arguments for ici_put_data are summarized in Table 5.1. The dimension of the data can be (:) or (:,:) or (:,:,:). When the dimension is 1 dimension, the data is assumed to be horizontal array which size is (nx*ny). When the dimension is 2 dimension, there will be two type of array assumption depends on the setting in coupling.conf file. The one is horizontal array which has (nx, ny) size, it is the case of no layer setting or layer = 1 in the coupling.conf. The other is horizontal+vertical array which has (nx*ny, nz) size, it is the case of layer> 1 in the coupling.conf file. When the dimension is 3 dimension, the data is assumed to be (nx, ny, nz).

Table 5.1: ici_put_data

routine name	argument type	argument	description
ici_put_data	put the send data		
	character(len=*), intent(IN)	data_name	name of data
	real(kind=8), intent(IN)	data(:) or data(:,:) or data(:,:,:)	send data

5.3 Setting for Data Exchange

The second step is to set the configuration for data exchange to coupling.conf file. The example of the setting is listed in List 5.1. There are two blocks on the setting, model/grid setting and data setting. The first block sets the exchange model and grid. In this documant, we assume that the model already coupled to ILSIO. Therefore the description of this block will be ommited. The second block sets the exchange data. Specification of this block is summerlized in the Table 5.2.

In the table, Italics means optional settings. "var_put" defines a data name of the send model. The name must be same as the argument of ici_put_data. "var_get" is a data name of ILSIO. This name must be same "data_name" defined in ILSIO configuration file ilsio.conf. "time_intpl_tag" and "grid_intpl_tag" are omittable or should be omitted for data output. "intvl" is time interval for output data. "lag" must be -1 for data output. "layer" defines the number of vertical layer of the data. "flag" sets whether to average the data over time. "factor" sets multiplier of the data. "io_ok" sets whether the data exchange is valid or not. When is_ok is set to 0, the data exchange and the data ouput is ignored even though the subroutine ici_put_data is called.

List 5.1: nam_ici section

```
1 &nam_ici comp_put = "MATSIRO", comp_get = "ILSIO",
2          grid_put ='matsiro_grid', grid_get ='io_grid3',
3          intpl_map_size = 63723,
4          map_file_name = "../../ILS_data/test/map/mat2io/grid.bin" ,
5 /
6
7 &nam_ici var_put = 'Wind_E_check' , var_get ='Wind_E_check' , grid_intpl_tag = 1,
      intvl=86400 , lag=-1, flag='AVR' /
8 &nam_ici var_put = 'Wind_N_check' , var_get ='Wind_N_check' , grid_intpl_tag = 1,
      intvl=86400 , lag=-1, flag='AVR' /
9 &nam_ici var_put = 'Tair_check' , var_get ='Tair_check' , grid_intpl_tag = 1, intvl
      =86400 , lag=-1, flag='AVR' /
```

Table 5.2: Specification of the section nam_ici(exchange data setting)

element name	description	possible value
var_put	send data name	string of the name
var_get	receive data name	string of the name
time_intpl_tag	tag of time interpolation	integer(¿0)
grid_intpl_tag	tag of spacial interpolation	integer(¿0)
intvl	exchange time interval	integer(¿0)
lag	model coupling pattern	-1
layer	number of vertical layers	integer(default=1)
flag	flag whether to average over time	"SNP" or "AVR"
factor	multiplier of the data	real(default=1)
io_ok	flag whether the data exchange is valid or not	0 or 1(default=1=valid)

5.4 Json file setting

NetCDF allows to add attributes of the variable in the file, and even more, some research projects require to set appropiriate attributes specified by the project. In ILS, attributes are set in the configuration file "ILS.json" as shown ins List 5.2. User must insert your variable attributes in the "variable_entry" section and, if nessesarry, modify the "Header" section.

If there is no attribute definition of the output variable, only "_FILLVALUE" attribute will be output.

List 5.2: example of ILS.json

```
1 {
2     "Header": {
```

```
        "title": "ILS output for GSWP3",
        "missing_value": "1e20",
        "int_missing_value": "-999",
        "contact": "Tomoko Nitta (t-nitta@iis.u-tokyo.ac.jp)",
        "institution": "Institute of Industrial Science, The University of Tokyo",
        "source": "ILS",
        "conventions": "CF-1.7 ALMA3",
        "history": "2019-05-30T18:20:00 Output from ILS"
    },
    "variable_entry": {
        "lat1": {
            "long_name": "",
            "standard_name": "latitude",
            "alma_name": "N/A",
            "amip_name": "N/A",
            "units": "degrees_north",
            "positive": "",
            "comment": ""
        },
        "Qr": {
            "long_name": "Total runoff",
            "standard_name": "runoff_flux",
            "alma_name": "Qr",
            "amip_name": "mrro",
            "units": "kg m-2 s-1",
            "positive": "out",
            "comment": "Total runoff"
        },
        "Qr_ice": {
            "long_name": "N/A",
            "standard_name": "N/A",
            "alma_name": "N/A",
            "amip_name": "N/A",
            "units": "kg m-2 s-1",
            "positive": "out",
            "comment": "Total ice runoff"
        }
    }
}
```

5.5 Setting of ILSIO.conf

Final task you mast do to output the data is to add the variable name and file name definition into IO configuration file ilsio.conf. "write_file_name" is a name of output file without extension. "data_name" is a name of write data. This name must be same as a name of "var_get" in the coupling configuration file "coupling.conf". For more detail of ilsio.conf, please refer the previous section (Section 4).

List 5.3: example of ilsio.conf

```
&nam_mio data_dir = "./", /
&nam_mio hgrid_name = "io_grid3",
         vgrid_name = "", /
```

```
4
5 &nam_mio write_file_name = "Wind_E", data_name = "Wind_E_check", /
6 &nam_mio write_file_name = "Wind_N", data_name = "Wind_N_check", /
7 &nam_mio write_file_name = "Tair", data_name = "Tair_check", /
8 &nam_mio write_file_name = "Qair", data_name = "Qair_check", /
```

Chapter 6

NetCDF file Input

6.1 Overview of this secrion

In this section, how to prepare a NetCDF file and how to set the configuration to input new data to a model will be described. Same as output case, it is assumed that the model is already coupled to the ILS system. When you need to know how to couple the model, please refer the coupling manual.

6.2 Prepare a NetCDF File

6.2.1 File Format

The format of NetCDF files that ILSIO can handle conforms to CF Convensions. The main standards are as follows.

- One file one data
- Data has spatial and tempral dimension
- Spatial dimension is 2D (horizontal) or 3D (horizontal + vertical)
- Horizontal demension is defined by X Axis and Y Axis
- Spatial dimension type is defined by the attribute "axis"
- The size of temporal dimension must be UNLIMITED
- Data must have start time
- Start time is defined by the attribute "units" of time variable
- Data time can have a range
- Whether or not the time variable has a time bounds is specified by the attribute "bounds" of the time variable

An example of NetCDF Input file is shown in List 6.1.

List 6.1: example of NetCDF Input file

```
netcdf leaf_area_index {
dimensions:
        time = UNLIMITED ; // (12 currently)
        lat = 360 ;
        lon = 720 ;
        bnds = 2 ;
variables:
        double time(time) ;
                time:long_name = "Time" ;
                time:standard_name = "time" ;
                time:calendar = "gregorian" ;
                time:units = "hours since 1871-01-01 00:00:00" ;
                time:bounds = "time_bnds" ;
        float lat(lat) ;
                lat:long_name = "Latitude" ;
                lat:standard_name = "latitude" ;
                lat:axis = "Y" ;
                lat:units = "degrees_north" ;
        float lon(lon) ;
                lon:long_name = "Longitude" ;
                lon:standard_name = "longitude" ;
                lon:axis = "X" ;
                lon:units = "degrees_east" ;
        float leaf_area_index(time, lat, lon) ;
                leaf_area_index:_fillvalue = 1.e+20 ;
                leaf_area_index:long_name = "" ;
                leaf_area_index:standard_name = "" ;
                leaf_area_index:units = "" ;
        double time_bnds(time, bnds) ;
data:

 time = 1131156, 1131876, 1132596, 1133328, 1134060, 1134792, 1135524,
    1136268, 1137000, 1137732, 1138464, 1139196 ;

 lat = 89.75, 89.25, 88.75, 88.25, 87.75, 87.25, 86.75, 86.25, 85.75, 85.25,
    84.75, 84.25, 83.75, 83.25, 82.75, 82.25, 81.75, 81.25, 80.75, 80.25,
     :::
    -83.75, -84.25, -84.75, -85.25, -85.75, -86.25, -86.75, -87.25, -87.75,
    -88.25, -88.75, -89.25, -89.75 ;

 lon = -179.75, -179.25, -178.75, -178.25, -177.75, -177.25, -176.75,
    -176.25, -175.75, -175.25, -174.75, -174.25, -173.75, -173.25, -172.75,
      :::
    173.25, 173.75, 174.25, 174.75, 175.25, 175.75, 176.25, 176.75, 177.25,
    177.75, 178.25, 178.75, 179.25, 179.75 ;

 leaf_area_index =
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     :::
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ;

```

```
 time_bnds =
  1130784, 1131528,
  1131528, 1132224,
  1132224, 1132968,
  1132968, 1133688,
  1133688, 1134432,
  1134432, 1135152,
  1135152, 1135896,
  1135896, 1136640,
  1136640, 1137360,
  1137360, 1138104,
  1138104, 1138824,
  1138824, 1139568 ;
}
```

6.2.2 Horizontal axis

Here, the horizontal grid of the data is assumed to be the same as the other data. If the grid is different from the other, it is necessary to start by creating a MAPPING TABLE. Please refer the SPRING manual for this procedure. The input file must conform to CF conventions, which specify coordinates in variable attribute "axis". The "axis" can be set to "X", "Y", "Z", or "T". For ILSIO input, at least the specification of X axis Y axis is mandatory for ILSIO input.

6.2.3 Start time

The data start time is set by the time variable attribute "units". Time unit must be "seconds", "minits", "hours", "days" or "months". Time string is allowd whether "YYYY-MO-DD" or "YYYY-MO-DD HH:MM:SS"

6.2.4 Time and Time bounds

Limitations on data time and time bounds are as described in the previous chapter. In addition, a variable of size (time, 2) with the name of the attribute value of the attribute "bounds" must be entered in the file.

6.3 Setting of ilsio.conf

For reading the file and sending the data, user must set the data configuration into IO configuration file ilsio.conf. Input data is divided into two types: (1) initial data, which is read only once before time integration, and (2) normal data, which is read during the time integration. When the data is initial data, add the configuration init_file_name as shown in List 6.2. When the data is normal data, add the configration read_file_name as shown in List 6.3.

List 6.2: configuration for initial data

```
1  &nam_mio data_dir = "../../ILS_data/test/bnd/", /
2  &nam_mio hgrid_name = "io_grid1",
3           vgrid_name = "", /
4
5  &nam_mio init_file_name = "lon.bin", data_name = "lon", io_mode="BIN" /
6  &nam_mio init_file_name = "lat.bin", data_name = "lat", io_mode="BIN" /
```

List 6.3: configuration for normal data

```
1  &nam_mio data_dir = "../../ILS_data/test/frc/", /
2  &nam_mio hgrid_name = "io_grid2",
3           vgrid_name = "", /
4
5  &nam_mio read_file_name = "GSWP3.BC.SWdown.3hrMap.ILS.2000.nc", data_name = "
       SWdown", /
6  &nam_mio read_file_name = "GSWP3.BC.LWdown.3hrMap.ILS.2000.nc", data_name = "
       LWdown", /
```

6.4 Setting of coupling.conf

For sending the data, user must set the configuration for data exchange to coupling.conf file. The example is as shown in List 6.4. Here the name of var_put must be same as the name of data_name in ilsio.conf, and "flag" must be set to "SNP". "time_intpl_tag" is only required if you are implementing your own time interpolation code, and normally does not need to be set.

List 6.4: configuration for data exchange

```
1  &nam_ici comp_put = "ILSIO", comp_get = "MATSIRO",
2           grid_put ='io_grid2', grid_get ='matsiro_grid',
3           intpl_map_size = 311304,
4           map_file_name = "../../ILS_data/test/map/metnc2mat/grid.bin",
5           coef_file_name = "../../ILS_data/test/map/metnc2mat/coef.bin",
6  /
7  &nam_ici var_put = 'SWdown' , var_get ='SWdown' , time_intpl_tag = 1,
       grid_intpl_tag = 1, intvl=3600 , lag=-1, flag='SNP' /
8  &nam_ici var_put = 'LWdown' , var_get ='LWdown' , time_intpl_tag = 2,
       grid_intpl_tag = 1, intvl=3600 , lag=-1, flag='SNP' /
9  &nam_ici var_put = 'CCover' , var_get ='CCover' , time_intpl_tag = 2,
       grid_intpl_tag = 1, intvl=3600 , lag=-1, flag='SNP' /
10 &nam_ici var_put = 'Rainf' , var_get ='Rainf' , time_intpl_tag = 2, grid_intpl_tag
       = 1, intvl=3600 , lag=-1, flag='SNP' /
11 &nam_ici var_put = 'Snowf' , var_get ='Snowf' , time_intpl_tag = 2, grid_intpl_tag
       = 1, intvl=3600 , lag=-1, flag='SNP' /
12 &nam_ici var_put = 'Wind' , var_get ='Wind' , grid_intpl_tag = 1, intvl=3600 , lag
       =-1, flag='SNP' /
13 &nam_ici var_put = 'Tair' , var_get ='Tair' , grid_intpl_tag = 1, intvl=3600 , lag
       =-1, flag='SNP' /
14 &nam_ici var_put = 'Qair' , var_get ='Qair' , grid_intpl_tag = 1, intvl=3600 , lag
       =-1, flag='SNP' /
15 &nam_ici var_put = 'PSurf' , var_get ='Psurf' , grid_intpl_tag = 1, intvl=3600 ,
       lag=-1, flag='SNP' /
```

6.4.1 Code Implementation

For getting the data, call ICI's API subroutine. The subroutine to be used depends on whether the data is initial data or normal data. ici_get_initial_data is used for initial data, and ici_get_data is used for normal data. "ici_get_initial_data" is called only once for each data before time integration as shown in List 6.5. "ici_get_data" is called only once for each data in the time integration loop (List 6.6).

List 6.5: example of ici_get_initial_data

```
    ! Longitudes
    call ici_get_initial_data("lon", lon)
    !
    ! Latitudes
    call ici_get_initial_data("lat", lat)
    !
    pi = atan(1.d0) * 4.d0
    alon(:) = lon(:) / 180.d0 * pi
    alat(:) = lat(:) / 180.d0 * pi
    !
    call ici_get_initial_data("lkfrac", lkfrac)
    lkfrac_prev(:) = lkfrac(:)
    call ici_get_initial_data("lkdep", hhclm)
    hhclm(:) = hhclm(:) * 1.d2 !! cm
    do ij = 1, ijmax
      hhclm(ij) = min(lksh_hhmin, hhclm(ij))
    enddo
```

List 6.6: example of ici_get_data

```
    select case(trim(met_type))
      case('gcm')
        call ici_get_data('Rdown', rfsfcd)
        call ici_get_data('Cosz' , rcosz)
      case('offline', 'offline_prcp')
        call ici_get_data('SWdown', swdown)
        call ici_get_data('LWdown', lwdown)
        call ici_get_data('CCover', cloud)
        do ij = 1, ijmax
          call shtins ( & ! incidence angle
             & sins, rcosz(ij), & ! out
             & idate(1), idate(2), idate(3), & ! in
             & idate(4), idate(5), 0, delt, & ! in
             & alon(ij), alat(ij), & ! in
             & nhsub , & ! in
             & solcon ) ! in
          call radder(rfsfcd_1d, swdown(ij), lwdown(ij), cloud(ij), rcosz(ij))
          rfsfcd(ij,:) = rfsfcd_1d(:)
        enddo
    end select
```