Chapter 32

Description of user defined routines

A series of usrdef_ routines have been created where the user can define all parameters and arrays needed for the setup of an application. The contents of these routines are discussed in Part IV. A summary is given in this chapter.

32.1 Harmonic analysis and output

usrdef_anal_freqs

`SUBROUTINE usrdef_anal_freqs`

File
: *Usrdef_Harmonic_Analysis.f90*

Type
: Subroutine

Purpose
: Define frequencies and related parameters for performing harmonic analysis.

Reference
: Section 20.3.1

Calling procedures
: harmonic_analysis_init

usrdef_anal_params

```
SUBROUTINE usrdef_anal_params
```

File
Usrdef_Harmonic_Analysis.f90

Type
Subroutine

Purpose
Define specifiers for harmonic output.

Reference
Section 20.3.2

Calling procedures
harmonic_analysis_init

usrdef_anal0d_vals

```
SUBROUTINE usrdef_anal0d_vals(out0ddat,n0vars)
INTEGER, INTENT(IN) :: n0vars
REAL, INTENT(OUT), DIMENSION(n0vars) :: out0ddat
```

File
Usrdef_Harmonic_Analysis.f90

Type
Subroutine

Purpose
Define the 0-D variables for harmonic analysis.

Reference
Section 20.3.3.1

Arguments

out0ddat	0-D data. The variables are given in the same order as they are defined in analvars.
n0vars	The number of 0-D harmonic variables

Calling procedures
harmonic_analysis_update

usrdef_anal2d_vals

```
SUBROUTINE usrdef_anal2d_vals(out2ddat,i,j,n2vars)
INTEGER, INTENT(IN) :: i, j, n2vars
REAL, INTENT(OUT), DIMENSION(n2vars) :: out2ddat
```

File
Usrdef_Harmonic_Analysis.f90

Type
Subroutine

Purpose
Define the 2-D variables for harmonic analysis.

Reference
Section 20.3.3.2

Arguments

out2ddat	2-D output data. The variables are given in the same order as they are defined in the array analvars.
i	X-index of the data location on the *local* model grid
j	Y-index of the data location on the *local* model grid
n2vars	The number of 2-D harmonic variables

Calling procedures
harmonic_analysis_update

usrdef_anal3d_vals

```
SUBROUTINE usrdef_anal3d_vals(out3ddat,i,j,k,n3vars)
INTEGER, INTENT(IN) :: i, j, k, n3vars
REAL, INTENT(OUT), DIMENSION(n3vars) :: out3ddat
```

File
Usrdef_Harmonic_Analysis.f90

Type
Subroutine

Purpose
Define the 3-D variables for harmonic analysis.

Reference
Section 20.3.3.3

Arguments

out3ddat	3-D output data. The variables are given in the same order as they are defined in the array analvars.
i	X-index of the data location on the *local* model grid
j	Y-index of the data location on the *local* model grid
k	Vertical index of the data location on the model grid
n3vars	The number of 3-D harmonic variables

Calling procedures
harmonic_analysis_update

32.2 Model setup

usrdef_init_params

`SUBROUTINE usrdef_init_params`

File
Usrdef_Model.f90

Type
Subroutine

Purpose
Define parameters for monitoring.

Reference
Section 14.2

Calling procedures
simulation_start

usrdef_mod_params

`SUBROUTINE usrdef_mod_params`

File
Usrdef_Model.f90

Type
Subroutine

Purpose
The following setup parameters are defined:

- number of processes and dimensions of the parallel grid (Section 14.3)
- model switches (Section 14.4)
- time parameters (Section 14.5.1)
- grid and other integer parameters (Sections 14.5.2–14.5.3)
- physical model parameters (Section 14.5.4)
- turbulence model parameters (Section 14.5.5)
- parameters for surface data grids (Section 14.6)
- attributes of forcing files (Section 14.7)
- parameters for user-defined output (Section 14.8)

Reference
Chapter 14

Calling procedures
initialise_model

usrdef_grid

`SUBROUTINE usrdef_grid`

File
Usrdef_Model.f90

Type
Subroutine

Purpose
Define model grid, bathymetry and locations of open boundaries.

Reference
Section 15.1

Calling procedures
initialise_model

usrdef_partition

`SUBROUTINE usrdef_partition`

File
Usrdef_Model.f90

Type
Subroutine

Purpose
Define domain decomposition for parallel applications.

Reference
Section 14.9

Calling procedures
domain_decomposition

usrdef_phsics

`SUBROUTINE usrdef_phsics`

File
Usrdef_Model.f90

Type
Subroutine

Purpose
Define physical initial conditions.

Reference
Section 15.2

Calling procedures
define_phsics

usrdef_1dsur_spec

`SUBROUTINE usrdef_1dsur_spec`

File
Usrdef_Model.f90

Type
Subroutine

Purpose
Define the surface forcing specifiers in case of a water column (1-D) application.

Reference
Section 17.1.1

Calling procedures
define_1dsur_spec

usrdef_2dobc_spec

```
SUBROUTINE usrdef_2dobc_spec
```

File
Usrdef_Model.f90

Type
Subroutine

Purpose
Define open boundary conditions for the 2-D mode.

Reference
Section 16.1.1

Calling procedures
define_2dobc_spec

usrdef_profobc_spec

```
SUBROUTINE usrdef_profobc_spec(iddesc,itypobu,itypobv,iprofobu,&
                            & iprofobv,iprofrlx,noprofsd,&
                            & indexprof,indexvar,novars,nofiles)
INTEGER, INTENT(IN) :: iddesc, nofiles, novars
INTEGER, INTENT(INOUT), DIMENSION(2:nofiles) :: noprofsd
INTEGER, INTENT(OUT), DIMENSION(nobu) :: itypobu
INTEGER, INTENT(OUT), DIMENSION(nobv) :: itypobv
INTEGER, INTENT(INOUT), DIMENSION(nobu,novars) :: iprofobu
INTEGER, INTENT(INOUT), DIMENSION(nobv,novars) :: iprofobv
```

```
INTEGER, INTENT(INOUT), DIMENSION(novars*(nobu+nobv),2:nofiles) :: indexprof
INTEGER, INTENT(INOUT), DIMENSION(novars*(nobu+nobv),2:nofiles) :: indexvar
INTEGER, INTENT(INOUT), DIMENSION(norlxzones) :: iprofrlx
```

File
Usrdef_Model.f90

Type
Subroutine

Purpose
Define open boundary conditions for baroclinic currents and 3-D scalars.

Reference
Section 16.2.1

Arguments

iddesc	The file descriptor key id of the 3-D quantity which may take the following values io_3uvobc baroclinic currents io_salobc salinity io_tmpobc temperature io_bioobc biological variables (currently not implemented)
itypobu	Type of the open boundary condition at the U-nodes (see Section 16.2.1.1)
itypobv	Type of the open boundary condition at the V-nodes (see Section 16.2.1.1)
iprofobu	Profile number used at the U-open boundaries for each data variable (0 is none)
iprofobv	Profile number used at the V-open boundaries for each data variable (0 is none)
iprofrlx	Disables/enables the application of the open boundary relaxation scheme within the zones defined in usrdef_rlxobc_spec (0/1).
noprofsd	Number of profiles per data file
indexprof	Mapping array of the profile numbers in the data files to the profile numbers assigned to the open boundaries. The physical size of the first dimension equals the number of profiles in a data file.

indexvar	Defines the variable number of the profiles in a data file. The physical size of the first dimension equals the number of profiles in a data file.
novars	Total number of variables. Value is 1 in the current implementation.
nofiles	The number of data files minus 1 (the file index for data files ranges from 2 to nofiles)

Calling procedures

define_profobc_spec

usrdef_1dsur_data

```
SUBROUTINE usrdef_1dsur_data(ciodatetime,data1d,novars)
CHARACTER (LEN=lentime), INTENT(INOUT) :: ciodatetime
INTEGER, INTENT(IN) :: novars
REAL, INTENT(INOUT), DIMENSION(novars) :: data1d
```

File

Usrdef_Model.f90

Type

Subroutine

Purpose

Define the surface forcing data in case of a water column (1-D) application.

Reference

Section 17.1.2

Arguments

ciodatetime	Returned date and time in string format
data1d	Returned forcing data. The array elements are: 1: X-component of the pressure gradient if isur1dtype=1 or 3, surface elevation if isur1dtype=2 2: Y-component of the pressure gradient if isur1dtype=1 or 3 3: surface elevation if isur1dtype=1
novars	The number of data variables depending on the value of isur1dtype

1: three data values (X- and Y-component of the pressure gradient and elevation)

2: one data value (elevation)

3: two data values (X- and Y-component of the pressure gradient)

Calling procedures

define_1dsur_data

usrdef_2dobc_data

```
SUBROUTINE usrdef_2dobc_data(ifil,ciodatetime,data2d,nodat,novars)
CHARACTER (LEN=lentime), INTENT(INOUT) :: ciodatetime
INTEGER, INTENT(IN) :: ifil, nodat, novars
REAL, INTENT(INOUT), DIMENSION(nodat,novars) :: data2d
```

File

Usrdef_Model.f90

Type

Subroutine

Purpose

Define the open boundary data for 2-D quantities.

Reference

Section 16.1.2

Arguments

ifil	File number index of the data file (>1)
ciodatetime	Returned date and time in string format
data2d	Returned values of the open boundary data
nodat	Number of data points as given by no2dobc(ifil)
novars	The number of data variables depending on the value of iobc2dtype(ifil)

1: two data values (depth integrated current and elevation)

2: one data value (elevation)

3: one data value (depth integrated current)

Calling procedures

define_2dobc_data

usrdef_profobc_data

```
SUBROUTINE usrdef_profobc_data(iddesc,ifil,ciodatetime,psiprofdat,numprofs)
CHARACTER (LEN=lentime), INTENT(INOUT) :: ciodatetime
INTEGER, INTENT(IN) :: iddesc, ifil, numprofs
REAL, INTENT(INOUT), DIMENSION(numprofs,nz) :: psiprofdat
```

File
Usrdef_Model.f90

Type
Subroutine

Purpose
Define the open boundary data for 3-D quantities.

Reference
Section 16.2.2

Arguments

iddesc	The file descriptor key id of the 3-D quantity which may take the following values:
	io_3uvobc baroclinic currents
	io_salobc salinity
	io_tmpobc temperature
	io_bioobc biological variables (currently not implemented)
ifil	File number index of the data file (>1)
ciodatetime	Returned date and time in string format
psiprofdat	Returned values of the open boundary profile data
numprofs	The number of profiles in the data file

Calling procedures
define_profobc_data

usrdef_rlxobc_spec

```
SUBROUTINE usrdef_rlxobc_spec
```

File
Usrdef_Model.f90

Type
: Subroutine

Purpose
: Define specifier arrays for the use of relaxation conditions near open boundaries.

Reference
: Section 16.3

Calling procedures
: define_rlxobc_spec

32.3 Setup of nested sub-grids

usrdef_nstgrd_spec

```
SUBROUTINE usrdef_nstgrd_spec
```

File
: *Usrdef_Nested_Grids.f90*

Type
: Subroutine

Purpose
: Define the number of nested open boundary locations and type of coordinates.

Reference
: Section 17.3.1

Calling procedures
: define_nstgrd_spec

usrdef_nstgrd_abs

```
SUBROUTINE usrdef_nstgrd_abs(iset,nhdat,nzdat,xcoord,ycoord,zcoord,cnode)
CHARACTER (LEN=lennode), INTENT(IN) :: cnode
INTEGER, INTENT(IN) :: iset, nhdat, nzdat
REAL, INTENT(OUT), DIMENSION(nhdat) :: xcoord, ycoord
REAL, INTENT(OUT), DIMENSION(nhdat,nzdat) :: zcoord
```

File
Usrdef_Nested_Grids.f90

Type
Subroutine

Purpose
Define the absolute geographical positions of the open boundaries for the sub-grid with index iset.

Reference
Section 17.3.2

Arguments

iset	The index number of the sub-grid
nhdat	Number of sub-grid points in the horizontal, equal to the value of either nohnstglbc(iset), nohnstglbu(iset) or nohnstglbv(iset) depending on the value of cnode
nzdat	Number of data points in the vertical equal to novnst(iset)
xcoord	X-coordinates of the sub-grid data points [meters or longitude]
ycoord	Y-coordinates of the sub-grid data points [meters or latitude]
zcoord	Z-coordinates of the sub-grid locations, defined as the negative distance to the mean water level (only when nzdat>0) [m]
cnode	Grid node of the sub-grid data points which may take the values 'C', 'U', 'V'

Calling procedures
define_nstgrd_locs

usrdef_nstgrd_rel

```
SUBROUTINE usrdef_nstgrd_rel(iset,nhdat,nzdat,nonodes,hnests,zcoord,cnode)
CHARACTER (LEN=lennode), INTENT(IN) :: cnode
INTEGER, INTENT(IN) :: iset, nhdat, nonodes, nzdat
REAL, INTENT(OUT), DIMENSION(nhdat,nzdat) :: zcoord
TYPE (HRelativeCoords), INTENT(OUT), DIMENSION(nhdat,nonodes) :: hnests
```

File
Usrdef_Nested_Grids.f90

Type
Subroutine

Purpose
Define the geographical positions of the open boundaries in relative coordinates for the sub-grid with index iset.

Reference
Section 17.3.3

Arguments

iset	The index number of the sub-grid
nhdat	Number of sub-grid points in the horizontal, equal to the value of either nohnstglbc(iset), nohnstglbu(iset) or nohnstglbv(iset) depending on the value of cnode
nzdat	Number of data points in the vertical equal to novnst(iset)
nonodes	The number of nodes on the model grid for which relative coordinates need to be provided. Its value equals 1 or 2 depending on the value of cnode.
hnests	Relative coordinates of the sub-grid ('C', 'U', 'V') locations with respect to either the C-, U- or V-node model grid
zcoord	Z-coordinates of the sub-grid locations, taken as the negative distance to the mean water level (only when nzdat>0) [m]
cnode	Grid node of the sub-grid data points which may take the values 'C', 'U', 'V'

Calling procedures
define_nstgrd_locs

32.4 User-formatted output

usrdef_output

```
SUBROUTINE usrdef_output
```

File
Usrdef_Output.f90

Type
Subroutine

Purpose
User-formatted output

Reference
Section 20.4

Calling procedures
coherens_main, initialise_model

32.5 Structures and discharges

usrdef_dry_cells

```
SUBROUTINE usrdef_dry_cells
```

File
Usrdef_Structures.f90

Type
Subroutine

Purpose
Define dry cell locations.

Reference
Section 6.1

Calling procedures
define_global_grid

usrdef_thin_dams

```
SUBROUTINE usrdef_thin_dams
```

File
Usrdef_Structures.f90

Type
Subroutine

Purpose
Define thin dam locations.

Reference
Section 6.2

Calling procedures
initialise_model

usrdef_weirs

`SUBROUTINE usrdef_weirs`

File
Usrdef_Structures.f90

Type
Subroutine

Purpose
Define weir and barrier locations and parameters.

Reference
Section 6.3

Calling procedures
initialise_model

usrdef_dischr_spec

`SUBROUTINE usrdef_dischr_spec`

File
Usrdef_Structures.f90

Type
Subroutine

Purpose
Define specifier arrays for the discharge module.

Calling procedures
define_dischr_spec

usrdef_dischr_data

```
SUBROUTINE usrdef_dischr_data(iddesc,ifil,ciodatetime,disdata,nodat,novars)
CHARACTER (LEN=lentime), INTENT(OUT) :: ciodatetime
INTEGER, INTENT(IN) :: iddesc, ifil, nodat, novars
REAL, INTENT(INOUT), DIMENSION(nodat,novars) :: disdata
```

File

Usrdef_Structures.f90

Type

Subroutine

Purpose

Define data (locations, volume, momentum and scalar discharge) for the discharge module.

Reference

Section 6.4

Arguments

iddesc	The file descriptor of the corresponding data file. The key id in parentheses below is the associated grid key id (idgrd).
	io_disvol volume discharges
	io_discur momentum (area and direction) discharge
	io_dissal salinity discharge data
	io_distmp temperatire discharge data
ifil	file index
ciodatetime	Returned date and time in string format
disdata	discharge data
nodat	Number of discharge locations (must be equal to numdis
novars	Number of input data variables which depends on the value of iddesc

Calling procedures

define_dischr_data

32.6 Surface grids and data

usrdef_surface_absgrd

```
SUBROUTINE usrdef_surface_absgrd(iddesc,ifil,n1dat,n2dat,xcoord,ycoord)
INTEGER, INTENT(IN) :: iddesc, ifil, n1dat, n2dat
REAL, INTENT(INOUT), DIMENSION(n1dat,n2dat) :: xcoord, ycoord
```

File
Usrdef_Surface_Data.f90

Type
Subroutine

Purpose
Define a surface grid in (absolute) geographical coordinates.

Reference
Section 17.2.1

Arguments

iddesc	The file descriptor of the corresponding data file. The key id in parentheses below is the associated grid key id (idgrd).
	io_metgrd surface meteo grid (igrd_meteo)
	io_sstgrd surface grid for sea surface temperature (igrd_sst)
	io_wavgrd surface wave grid (igrd_waves)
ifil	File index. In the current implementation its value is 1.
n1dat	X-dimension of the data grid equal to surfacegrids(idgrd,ifil)%n1dat
n2dat	Y-dimension of the data grid equal to surfacegrids(idgrd,ifil)%n2dat
xcoord	X-coordinates of the data grid [m or degrees longitude]
ycoord	Y-coordinates of the data grid [m or degrees latitude]

Calling procedures
define_surface_input_grid, define_surface_output_grid

usrdef_surface_relgrd

```
SUBROUTINE usrdef_surface_relgrd(iddesc,ifil,surfgridglb,nx,ny,nonodes)
INTEGER, INTENT(IN) :: iddesc, ifil, nonodes, nx, ny
TYPE (HRelativeCoords), INTENT(INOUT), DIMENSION(nx,ny,nonodes) :: surfgridglb
```

File
Usrdef_Surface_Data.f90

Type
Subroutine

Purpose
Define the relative coordinate arrays of a surface grid with respect to model grid.

Reference
Section 17.2.2

Arguments

iddesc	The file descriptor of the corresponding data file. The key id in parentheses below is the associated grid key id (idgrd).
	io_metgrd surface meteo grid (igrd_meteo)
	io_sstgrd surface grid for sea surface temperature (igrd_sst)
	io_wavgrd surface wave grid (igrd_waves)
ifil	File index. In the current implementation its value is 1.
surfgridglb	Returned relative coordinates of the model C-node grid with respect to a data grid represented by iddesc
nx	Currenly equal to the global X-dimension nc of the model grid.
ny	Currenly equal to the global Y-dimension nr of the model grid.
nonodes	Number of nodes. In the current implementation its value equals 1.

Calling procedures
define_surface_input_grid, define_surface_output_grid

usrdef_surface_data

```
SUBROUTINE usrdef_surface_data(iddesc,ifil,ciodatetime,surdata,&
                               & n1dat,n2dat,novars)
CHARACTER (LEN=lentime), INTENT(INOUT) :: ciodatetime
INTEGER, INTENT(IN) :: iddesc, ifil, novars, n1dat, n2dat
REAL, INTENT(INOUT), DIMENSION(n1dat,n2dat,novars) :: surdata
```

File
: *Usrdef_Surface_Data.f90*

Type
: Subroutine

Purpose
: Define the input surface forcing data.

Reference
: Section 17.2.3

Arguments

iddesc	The file descriptor of the corresponding data file io_metsur surface meteo data io_sstsur sea surface temperature data
ifil	File index. In the current version its value is 1.
ciodatetime	Returned date and time in string format
surdata	Returned surface forcing data as defined on the surface data grid
n1dat	X-dimension of the data grid equal to surfacegrids(idgrd,ifil)%n1dat
n2dat	Y-dimension of the data grid equal to surfacegrids(idgrd,ifil)%n2dat
novars	Number of data variables. In case of meteorological data its value ranges from 2 to 7, in case of SST data its value is 1.

Calling procedures
: define_surface_data

32.7 Time averaged output

usrdef_avr_params

```
SUBROUTINE usrdef_avr_params
```

File
Usrdef_Time_Averages.f90

Type
Subroutine

Purpose
Define specifiers for time averaged output.

Reference
Section 20.2.1

Calling procedures
time_averages_init

usrdef_avr0d_vals

```
SUBROUTINE usrdef_avr0d_vals(out0ddat,n0vars)
INTEGER, INTENT(IN) :: n0vars
REAL, INTENT(OUT), DIMENSION(n0vars) :: out0ddat
```

File
Usrdef_Time_Averages.f90

Type
Subroutine

Purpose
Define the 0-D variables for time averaged output.

Reference
Section 20.2.2.1

Arguments

out0ddat	0-D data. The variables are given in the same order as they are defined in avrvars.
n0vars	The number of 0-D output variables

Calling procedures
time_averages_update

usrdef_avr2d_vals

```
SUBROUTINE usrdef_avr2d_vals(out2ddat,i,j,n2vars)
INTEGER, INTENT(IN) :: i, j, n2vars
REAL, INTENT(OUT), DIMENSION(n2vars) :: out2ddat
```

File
Usrdef_Time_Averages.f90

Type
Subroutine

Purpose
Define the 2-D variables for time averaged output.

Reference
Section 20.2.2.2

Arguments

out2ddat	2-D output data. The variables are given in the same order as they are defined in the array avrvars.
i	X-index of the data location on the *local* model grid
j	Y-index of the data location on the *local* model grid
n2vars	The number of 2-D output variables

Calling procedures
time_averages_update

usrdef_avr3d_vals

```
SUBROUTINE usrdef_avr3d_vals(out3ddat,i,j,k,n3vars)
INTEGER, INTENT(IN) :: i, j, k, n3vars
REAL, INTENT(OUT), DIMENSION(n3vars) :: out3ddat
```

File
Usrdef_Time_Averages.f90

Type
Subroutine

Purpose
Define the 3-D variables for time averaged output.

Reference
Section 20.2.2.3

Arguments

out3ddat	3-D output data. The variables are given in the same order as they are defined in the array avrvars.
i	X-index of the data location on the *local* model grid
j	Y-index of the data location on the *local* model grid
k	Vertical index of the data location on the model grid
n3vars	The number of 3-D output variables

Calling procedures
time_averages_update

32.8 Time series output

usrdef_tsr_params

```
SUBROUTINE usrdef_tsr_params
```

File
Usrdef_Time_Series.f90

Type
Subroutine

Purpose
Define specifiers for time series output.

Reference
Section 20.1.1

Calling procedures
time_series_init

usrdef_tsr0d_vals

```
SUBROUTINE usrdef_tsr0d_vals(out0ddat,n0vars)
INTEGER, INTENT(IN) :: n0vars
REAL, INTENT(OUT), DIMENSION(n0vars) :: out0ddat
```

File
Usrdef_Time_Series.f90

Type
Subroutine

Purpose
Define the 0-D variables for time series output.

Reference
Section 20.1.2.1

Arguments

out0ddat	0-D data. The variables are given in the same order as they are defined in tsrvars.
n0vars	The number of 0-D output variables

Calling procedures
time_series

usrdef_tsr2d_vals

```
SUBROUTINE usrdef_tsr2d_vals(out2ddat,i,j,n2vars)
INTEGER, INTENT(IN) :: i, j, n2vars
REAL, INTENT(OUT), DIMENSION(n2vars) :: out2ddat
```

File
Usrdef_Time_Series.f90

Type
Subroutine

Purpose
Define the 2-D variables for time series output.

Reference
Section 20.1.2.2

Arguments

out2ddat	2-D output data. The variables are given in the same order as they are defined in the array tsrvars.
i	X-index of the data location on the *local* model grid

j	Y-index of the data location on the *local* model grid
n2vars	The number of 2-D output variables

Calling procedures
time_series

usrdef_tsr3d_vals

```
SUBROUTINE usrdef_tsr3d_vals(out3ddat,i,j,k,n3vars)
INTEGER, INTENT(IN) :: i, j, k, n3vars
REAL, INTENT(OUT), DIMENSION(n3vars) :: out3ddat
```

File
Usrdef_Time_Series.f90

Type
Subroutine

Purpose
Define the 3-D variables for time series output.

Reference
Section 20.1.2.3

Arguments

out3ddat	3-D output data. The variables are given in the same order as they are defined in the array tsrvars.
i	X-index of the data location on the *local* model grid
j	Y-index of the data location on the *local* model grid
k	Vertical index of the data location on the model grid
n3vars	The number of 3-D output variables

Calling procedures
time_series