Chapter 9

Model input and output

9.1 Classification of model files

The files, which can be created by the model fall to four different categories. Exception is the file *defruns* (see Section 14.1).

1. Monitoring files
    - log file
        - Writes "tracing" information during the simulation.
        - Informs about progress of the run upto a user-defined level. A zero level means that the log file is not created. Note that this is the default setting.
        - For parallel runs a log file can be created by each process.
        - The utility is useful for debugging the model code or for tracing errors in model setup.
    - error file
        - Writes error messages.
        - If an error is detected by an error checking routine, an error message is written. In most cases the program aborts immediately afterwards while in other cases several checks are made before the program aborts.
        - For parallel runs an error file is created by each process.
    - warning file
        - Writes "warning" messages about suspicious values of setup parameters or variables.
        - The program will not abort.

- timer report
  This is a file with information about the total execution time and the percentages of time spent by different model "compartments" (e.g. advection, 2-D mode, I/O, parallel communications, ...).

2. Central input file (CIF)

   - This is a file with a complete list of parameter values used for the setup of the application. This includes all the parameters which can be defined by the user in the routines

     `usrdef_init_params, usrdef_mod_params, usrdef_tsr_params, usrdef_avr_params, usrdef_anal_freqs, usrdef_anal_params`

     For further details see Chapters 14 and 20.
   - The file is created on request by the user and can be used as input to the program in a subsequent run.
   - If the CIF is used for input, the above routines are no longer called by the program.

3. Forcing files

   - Those files include data arrays used for model setup, e.g. model grid data, type of open boundary conditions, initial conditions, forcing data (open boundaries and meteorological). The files with open boundary data for nested sub-grids also fall within this category.
   - All files are used as input to the model, except the open boundary data files for nested sub-grids which are defined as output files.
   - All input files may be given in any user-defined format, but can optionally be converted to a COHERENS standard format. The output files are always in COHERENS standard format. The forcing file can be made "virtual", if the forcing data are directly defined by the user without using an external data file.

4. User output files

   - These are output data files created by the program on request by the user.
   - Spatial and temporal resolution and type of output data are selected by the user.
   - The files are always in COHERENS standard format.

- Output specifications are defined in the user-defined routines usrdef_tsr_params for time series, usrdef_avr_params for time averaged and usrdef_anal_params, usrdef_anal_freqs for harmonic data output.

9.2 Default file names

Each file has a default name which can be reset by the user. A default file name (in FORTRAN string format) is defined as:

```
title//'.'//filedesc//filenum//form//pid
```

for monitoring files, central input file(s) and forcing files, and

```
title//'_'//filenum//'.'//freqnum//filedesc//dim//form
```

for output files, where title is a simulation-specific title, filedesc the file descriptor (representative for the type of the data in the file), filenum the file number in case there is more than one file with the same descriptor, form the format of the file, pid the process id number, dim the dimension of output data and freqnum the frequency number. Values of these sub-strings are discussed below.

9.2.1 title

Possible values are

runtitle Simulation title defined in *defruns*. This is the value used for monitoring files and CIF(s)(Categories 1 and 2).

intitle User-defined parameter used as prefix name for forcing files (Category 3). Default value is runtitle.

outtitle User-defined parameter used as prefix name for user output files (Category 4). Default value is runtitle.

9.2.2 pid

Process id number, used only for log and error files in parallel mode, empty otherwise.

9.2.3 form

The file format can take the following values:

'A' ASCII format. This format is portable and readable but unsuitable for large data sets because of its great comsumption of disk space and its use of sequential storage.

'U' Unformatted binary format. Advantage is a more efficient use of disk space. Disadvantages are that the file is (mainly) non-portable, not directly readable and uses sequential storage.

'N' NetCDF format (recommended). This format is portable, directly readable (with the netCDF `ncdump` utility). Data can be accessed directly by specifying the appropriate record number. Only (possible) disadvantage is that the netCDF library needs to be compiled first.

'I' This does not represent a specific data format but indicates that the file is an ASCII information file containing the metadata of a forcing or output data file.

Monitoring and central input files are always in 'A'-format. Forcing and output files may be in any of the first three formats.

9.2.4 filedesc

The string filedesc, further denoted as the "file descriptor", is a character string taking one of the values below.

1. Monitoring files

   inilog log file with tracing information during the initialisation phase of the simulation

   runlog log file with tracing information during the main (time loop) phase of the simulation

   errlog error file

   warlog warning file

   timing timer report file

2. CIF file. Only one file is currently available.

   cifmod parameters for model setup

3. Forcing files. The string filedesc refers to the contents of the file.

mppmod	arrays defining the domain decomposition (parallel mode only)
inicon	initial conditions
fincon	final conditions
modgrd	model grid, bathymetry and location of open boundaries
metgrd	surface meteorological grid
sstgrd	sea surface temperature (SST) grid
wavgrd	surface wave grid
nstgrd	locations of open boundary locations of nested sub-grids. A file number needs to be supplied for each sub-grid.
sedspc	specifiers (i.e. particle attributes) for the sediment module
1uvsur	1-D surface forcing
2uvobc	2-D mode open boundary forcing
3uvobc	3-D mode (baroclinic currents) open boundary forcing
salobc	salinity open boundary forcing
tmpobc	temperature open boundary forcing
sedobc	sediment open boundary forcing
rlxobc	definitions of relaxation zones
nstspc	specifiers for sub-grid nesting
2uvnst	2-D open boundary data for the nested sub-grids. A file number is supplied for each sub-grid.
3uvnst	baroclinic current open boundary data for the nested sub-grids. A file number is supplied for each sub-grid.
salnst	salinity open boundary data for the nested sub-grids. A file number needs is supplied for each sub-grid.
tmpnst	temperature open boundary data for the nested sub-grids. A file number is supplied for each sub-grid.
sednst	sediment open boundary data for the nested sub-grids. A file number is supplied for each sub-grid.
metsur	meteorological surface data
sstsur	SST surface data
wavsur	surface wave data
drycel	dry cell locations
thndam	thin dam locations

weibar	weirs/barries locations and parameters
disspc	discharge specifiers
disloc	discharge locations
disvol	volume discharges
discur	momentum discharges
dissal	salinity discharge
distmp	temperature discharges

4. Output files.

tsout	time series output
avrgd	time averaged output
resid	output of residuals
amplt	output of harmonic amplitudes
phase	output of harmonic phases
ellip	output of tidal ellipse parameters

9.2.5 filenum

1. If the file descriptor equals 1uvsur, 2uvobc, 3uvobc, salobc, tmpobc or sedobc, the contents of the file depends on the value of filenum:

   - filenum=1: forcing specifiers (e.g. type of open boundary conditions)
   - filenum>1: forcing data itself. This allows to spread the data over several files. For example, one file may contain open sea and another river open boundary data. Each file can have its own temporal resolution.

2. In case the file descriptor equals 2uvnst, 3uvnst, salnst,tmpnst or sednst, the file contains nested data and filenum equals the number of the associated sub-grid (between 1 and nonestsets).

3. In all other cases (including surface forcing data) the file number is not used.

The maximum allowed file number is given by the system parameter MaxIOFiles, defined in *syspars.f90*.

9.2.6 freqnum

Frequency number only used for output of amplitudes, phases and elliptic parameters. Empty otherwise.

9.2.7 dim

The file dimension is only used for files of Category 4:

0 : 0-D output data grid

2 : 2-D output data grid

3 : 3-D output data grid

G: file containing the output grid coordinates but not the data themselves

9.3 Formats of monitoring files

9.3.1 Log files

Log files contain the following information:

- Some general information (e.g. current date, number of time steps, value of 2-D CFL limit, ...).
- Each time a file is opened within the program, a message is written with the name, unit number and format of the file. A similar message is printed when a file is closed.
- A line starting with a number followed by ':' and the name of a routine means that the program entered this routine. The number denotes the program routine level. Main program is at level 1, a routine called by the main program has level 2, a routine called by another routine at level n is at level n+1, ...
- The maximum number of levels traced by the log file, is defined by the user. Note that large log files may be written if this maximum equals or exceeds a value larger than 5. A value of 3 is recommended for normal runs, a value of 7 if the log file is used for debugging. When the program returns to the previous level (at the end of the called routine), a line is written with the same level number followed by ':R'.
- Two separate log files can be written by the program: the *inilog* file which traces information during the initialisation phase and is closed

when the program enters the time loop and the *runlog* file which is active during the time loop only. The two program phases are discussed in Section 12.2.

- If the maximum level for tracing is set to zero (default), no log file is written.
- The following parameters can be defined by the user:
  - The tracing level of the *inilog* and *runlog* files. In parallel mode, different values can be selected for different sub-domains. Note that, by default, all levels are set to zero so that no log file is created.
  - The names of the log files. In parallel mode, all *inilog* or *runlog* files have the same standard name (which can be redefined by the user) appended by a suffix with the process id number.
  - A number of time steps can be defined after which the *runlog* file is overwritten. Default is the total number of time steps (i.e. information is written at all time steps and the file is never overwritten).
  - The writing of an exit statement of the form 'num:R', where 'num' is the program level in the "log"-file on exit of a routine call if .TRUE. (which is also the default).

  For details see Section 14.2.2.

An example is given below.

```
Open file fredyA.runlogA on unit 1 type OUT (A)
2:update_time
3:add_secs_to_date_int
3:R
3:convert_date_to_char
3:R
3:day_number_int
3:R
2003/01/01;00:00:30,000
2:R
2:equation_of_state
2:R
2:baroclinic_gradient
3:Zcoord_arr
```

```
3:R
3:Carr_at_W: sal
3:R
...
2:R
2:hydrodynamic_equations
3:current_pred
3:R
3:current_2d
umax =   0.1765635E-02 (28,30)
vmax =   0.1765635E-02 (30,28)
3:R
...
Close file on unit 8 (A)
2:R
2:simulation_end
3:rng_finalize
3:R
3:timer_report
Open file fredyA.timingA on unit 4 type OUT (A)
3:R
3:deallocate_mod_arrays
3:R
2:End of simulation: fredyA
Close file fredyA.warlogA on unit 3 (A)
Close file fredyA.errlogA on unit 2 (A)
Close file fredyA.runlogA on unit 1 (A)
```

Example 9.1: Part of the *runlog* file written by running test case ***fredyA***. Tracer level is 3.

The tracing information in the log files is implemented within the code by defining the name of the routine and calling the routine log_timer_in on entry and log_timer_out just before exiting the routine, e.g.

```
procname(pglev+1) = 'open_boundary_arrays'
CALL log_timer_in()
...
CALL log_timer_out()
```

The first line is the name of the routine which has been entered and will be written to the log file. Routine log_timer_in sets the current program level by

increasing the counter pglev by 1 and writes the entry message. The call to log_timer_out decreases pglev by 1. Note that if the first call is programmed in the code, the second one must be inserted as well just before the RETURN statement.

9.3.2 Error files

Error checking routines have been implemented in the model code. Errors are always considered as fatal which means that the program aborts. The program exit is usually executed after a series of checks. In this way several error messages can be written to the *errlog* file. The file is created in the beginning of the program. If no errors are found, the file is deleted at the end of the simulation. Error checking is controlled by the following parameters which can be set by the user:

- Level of error checking

  0: Error checking is disabled and no file is created. This is the default.

  1: Error checking is performed during initialisation only.

  2: Error checking is enabled throughout the whole program (initialisation, time loop and finalisation). This level is very useful for the detection of read errors (e.g. end of file conditions) during an input operation, but should be selected only to check the program for the first time, since it may affect the CPU performance.

  In parallel mode, different levels can be taken for different sub-domains.

- The name of the error file. In parallel mode, all *errlog* files have the same standard name (which can be redefined by the user) appended by a suffix with the process id number.

- The maximum number of error messages. Default is the system parameter MaxErrMesgs. The reason for limiting the number of messages, is to avoid that an unnecessary amount of error messages is written, since the program performs checks on model arrays as well.

The format of an *errlog* file is illustrated with the example below. Line numbers are added for illustration purposes only.

```
1:Unable to open non-existing file: rhonegrid.dat
2:A total of 1 errors occurred in open_filepars
3:Error type 1 : Not possible to open file
4:PROGRAM TERMINATED ABNORMALLY
```

Table 9.1: Key ids for error coding and associated error messages.

key_id	message
ierrno_fopen	Not possible to open file
ierrno_fclose	Unable to close file
ierrno_read	Read error
ierrno_write	Write error
ierrno_fend	End of file condition
ierrno_input	Wrong input values
ierrno_inival	Invalid initial values for model parameters or arrays
ierrno_runval	Invalid values for variables at run time
ierrno_alloc	Not possible to allocate arrays
ierrno_arg	Missing or invalid argument in a routine call
ierrno_comms	Communication error
ierrno_MPI	Error in a MPI call
ierrno_CDF	Error in a netCDF call

Example 9.2: Contents of an *errlog* file.

The first line gives a description of the error. When more than one error is found, a message line is written for each error. The second line gives the name of the routine where the error is found. The third line writes a message code describing the general type of the error(s). Each message code is presented in the program by a key id of the form ierrno_*. A list of available key ids is given in Table 9.1. The last line in example 9.2 is standard for all *errlog* files.

The error code is programmed as follows:

- The number of detected errors is given by the parameter nerrs. Its initial value is zero.
- A series of routines are implemented for checking setup variables. For example, the model parameters, defined in usrdef_mod_params, are checked in check_mod_params. These routines are located in *check_model.f90* and fully described in Section 31.3. Error checking is also performed when a file is opened or closed, metadata are read from a forcing file, or if a READ or an end-of-file condition occurs.
- Within these "checking" routines, calls are made to one or more routines, defined in *error_routines.F90*. For example, error_limits_var to verify whether a switch has allowed values between certain limits. If

the routine detects an error, the error message is written and nerrs is increased. These routines in *error_routines.f90* are fully described in Section 31.8.

- The routine error_abort is called with a typical error code, in the form of a key id. If nerrs>0, lines 2–4 of the previous example are written with an error message which corresponds to the given key id (see Table 9.1), and the program aborts immediately afterwards. In parallel mode, the error message will only be written by the processes where error(s) were detected.

For example

```
CALL error_lbound_var(nc,'nc',0,.FALSE.)
...
CALL error_limits_var(iopt_grid_htype,'iopt_grid_htype',1,3)
...
CALL error_lbound_var_date(CEndDateTime,'CEndDateTime',CStartDateTime,&
                        & .FALSE.)
...
CALL error_limits_var(dlat_ref,'dlat_ref',-90.0,90.0)
...
CALL error_abort('check_mod_params',ierrno_inival)
```

Example 9.3: Excerpts of routine check_mod_params illustrating the use of error coding.

The first four calls are error checking routines. The first tests whether nc is positive, the second whether the switch iopt_grid_htype has a value between 1 and 3, the third whether the end date is later than the start date, the fourth whether the reference latitude is between -90^0 and 90^0. If a test turns .FALSE., an error message is written. The routine error_abort is called with the key id ierrno_inival, representative for invalid setup parameters.

9.3.3 Warning file

Besides error messages which are always fatal, the program may also write warning messages which do not cause termination of the program. The utility can be useful for debugging. The aim is to provide information for the user about suspicous selection of model parameters (usually switches) or arrays or to inform the user that certain setup parameters (defined by the user or default) have been reset. For example, if a 2-D simulation is selected with

iopt_grid_nodim=2 and the vertical resolution parameter nz is set to a value larger than 1, a warning message is issued that this parameter is reset to 1. The following control parameters can be (re)set by the user:

- The warning utility is switched on by default, but can be disabled by the user.
- The name of the warning file. In parallel mode, only one file is permitted, written by the master process.

```
WARNING: value of integer parameter iopt_mode_2D is set from 1 to 0
WARNING: value of integer parameter iopt_dens_grad is set from 1 to 0
WARNING: value of integer parameter iopt_bstres_drag is set from 3 to 0
WARNING: value of integer parameter nprocsx is set from 0 to 1
WARNING: value of integer parameter nprocsy is set from 0 to 1
```

Example 9.4: Contents of the *warlog* file produced by running test case ***pycnoA***.

9.3.4 Timer report file

A timer report is a file which contains information about the total execution time and the percentages of time spent by different model "compartments". Each compartment has an associated time key id, listed in Table 9.2. The following control parameters can be defined by the user:

- The type of information contained in the report.
  - 0: No timer report is written. This is the default.
  - 1: Only the total execution time is written.
  - 2: Time information (in percentage of total time) is written for all "timers". In case of a parallel simulation, the percentages are given for the process with the largest amount of time, the lowest amount of time, as an average over all processes and for the master process.
  - 3: The same as the previous case, but the time percentages are now given for each individual process in addition. In the serial case, behaviour is as for case 2.
- The name of the timer report file. In parallel mode, only one file is permitted, written by the master process.
- The unit of time for writing the total execution can be written in seconds, minutes, hours or days. Default is seconds.

Table 9.2: Timer key ids and their meaning.

key_id	description
itm_hydro	hydrodynamics
itm_1dmode	water column mode calculations
itm_2dmode	2-D mode calculations
itm_3dmode	3-D mode calculations
itm_dens	total of density (including temperature and salinity) calculations
itm_temp	temperature
itm_sal	salinity
itm_init	initialisation procedures
itm_trans	transport routines
itm_adv	advection routines
itm_hdif	horizontal diffusion
itm_vdif	vertical diffusion (including turbulence modules)
itm_phgrad	baroclinic pressure gradient
itm_input	input operations
itm_output	output operations
itm_inout	total of input and output operations
itm_ com_coll	collect communication calls
itm_com_comb	combine communication calls
itm_com_copy	copy communication calls
itm_com_dist	distribute communication calls
itm_com_exch	exchange communication calls
itm_com_util	utility communication calls
itm_coms	total of parallel communications
itm_MPI	total of MPI calls
itm_CDF	netCDF calls
itm_arrint	interpolation of model grid arrays
itm_user	usrdef routine calls
itm_nest	nesting procedures
itm_libs	internal library routine calls
itm_astro	astronomical tide
itm_bconds	boundary conditions
itm_meteo	meteorological routines
itm_structs	structures and discharges
itm_wait	wait calls
itm_sed	sediment model
itm_bio	biological model

The utility is useful for testing the CPU efficiency of a parallel decomposition. An example of such test is given in Figure 11.1. Two examples are given below using the same test case ***plumeC***. The first shows the contents of the timer report obtained from a serial run:

```
plume1C: 392s.822
Hydrodynamics       :  39.152
2D mode             :  12.125
3D mode             :  27.939
Density             :  26.150
Salinity            :  22.338
Initialisation      :   0.025
Transport           :  50.284
Advection           :  25.721
Horizontal diffusion:  13.541
Vertical diffusion  :  18.450
Baroclinic pressure :   3.118
Input               :   0.001
Output              :   1.762
Input/output        :   1.763
Array interpolation :  12.917
User calls          :  14.979
Library calls       :   4.284
Boundary conditions :   0.660
```

Example 9.5: Timer report for test case ***plumeC*** on a serial machine.

The second example is for the same test case now obtained on a parallel machine with four processors and full timer information. For each timer process there are now two lines. The first one gives statistical information (maximum, minimum, mean and master). The second gives the times for each individual processor. The report now contains additional information about parallel communication calls, given by the compartments 'Combine comms', ..., 'MPI calls'. Note that the numbers given in 'Parallel comms' are the sum of the corresponding ones for the combine, copy, exchange and utility operations. This information is not given in the serial case since the times related to parallel communications are, obviously zero, and zero times are not printed in the table.

```
plume1C: 169s.783
Hydrodynamics       :  44.064 44.009 44.037 44.044
```

```
   44.044 44.064 44.009 44.039
2D mode             :  18.971 18.889 18.939 19.172
   19.172 18.958 18.889 18.971
3D mode             :  25.648 25.584 25.620 25.568
   25.568 25.648 25.584 25.627
Density             :  27.339 27.210 27.285 27.176
   27.176 27.339 27.210 27.306
Salinity            :  23.711 23.659 23.677 23.677
   23.677 23.711 23.659 23.660
Initialisation      :   0.094  0.088  0.092  0.088
    0.088  0.088  0.094  0.094
Transport           :  45.924 45.189 45.574 45.046
   45.046 45.189 45.610 45.924
Advection           :  28.831 28.505 28.639 28.648
   28.648 28.505 28.582 28.831
Horizontal diffusion:  12.218 11.796 12.077 12.074
   12.074 11.796 12.218 12.216
Vertical diffusion  :  10.446 10.359 10.412 10.189
   10.189 10.359 10.446 10.431
Baroclinic pressure :   2.963  2.891  2.917  2.845
    2.845  2.898  2.891  2.963
Input               :   0.024  0.024  0.024  0.071
    0.071  0.024  0.024  0.024
Output              :   0.000  0.000  0.000  2.951
    2.951  0.000  0.000  0.000
Input/output        :   0.024  0.024  0.024  3.022
    3.022  0.024  0.024  0.024
Combine comms       :   3.351  3.239  3.290  0.353
    0.353  3.239  3.280  3.351
Copy comms          :   0.035  0.029  0.031  0.000
    0.000  0.035  0.029  0.029
Exchange comms      :  12.032 10.301 10.995 11.444
   11.444 12.032 10.652 10.301
Utility comms       :   0.524  0.424  0.465  0.530
    0.530  0.424  0.448  0.524
Parallel comms      :  15.731 14.206 14.782 12.328
   12.328 15.731 14.409 14.206
MPI calls           :  12.998 11.485 12.016  9.271
    9.271 12.998 11.565 11.485
Array interpolation :  12.404 11.920 12.165 11.939
   11.939 11.920 12.171 12.404
```

```
User calls          :   9.842  9.694  9.776 10.525
   10.525  9.694  9.792  9.842
Library calls       :   2.544  2.385  2.463  2.645
    2.645  2.544  2.461  2.385
Boundary conditions :   0.536  0.477  0.514  0.459
    0.459  0.477  0.530  0.536
```

Example 9.6: Timer report for test case ***plumeC*** on a parallel machine with four processors.

The following example shows how timing is implemented in the program

```
CALL log_timer_in(npcc)
...
CALL log_timer_out(npcc,itm_adv)
```

A timer vector array is created at the start of the program and initialised to zero. The routine log_timer_in is called in the beginning of the routine and stores the current clock count value of the processor clock in the optional argument npcc. The value is obtained by calling the FORTRAN 90 intrinsic routine SYSTEM_CLOCK. The call to log_timer_out is made at the last line of the subprogram. The routine calls SYSTEM_CLOCK again and subtracts the new clock count from the previous one. This gives the time spent in the routine, measured in clock counts. The result is added to the value stored in the corresponding element of the "timer" vector array. The array index is given by the timer key id itm_adv. At the end of the program the array values are converted to seconds, divided by the total execution time and multiplied by 100. This gives the computation times associated with different timer key ids in percentage.

9.4 Central input file

9.4.1 Syntax of a CIF

As shown in the example 9.7 below, each data line in the CIF has the following syntax

```
varname = value 1, value 2, ..., value n
```

where varname is the FORTRAN name of a model parameter and value 1 to value n are the input values of the parameter, separated by the data separator ','. The file is read line-wise. The data strings value 1, value 2,

. . . are converted to the appropriate (numeric, logical, character) data format associated with the variable FORTRAN variable varname. The following rules apply

- If a comment character '!' appears in the string, all characters in the string, starting from this character are ignored. However, the comment character can only appear at the first position of the data line (in which case the entire line is ignored) or after the last character of the last data string.

- If varname corresponds to a scalar, it is obvious that only one value needs to be given and there is no data separator. In case of a vector, the number of data can be lower than the size of the vector in which case the non-defined values are set to their defaults. However if a vector has a specified "physical" size, all expected data must be given. Examples are the arrays index_obc (physical size given by nconobc) or ntrestart (physical size given by norestarts).

- If the model parameter represents a multi-dimensional array (of rank m), the first m-1 data strings represent the vector index for the first m-1 dimensions, the subsequent the values for each array index of the last dimension. As before, the number of values does not need to be equal to the size of the last dimension, unless a "physical" size is expected.

- If the variable is a derived type scalar variable, the data strings represent the components in the order given by the TYPE definition in *datatypes.f90*. Derived type arrays are initialised element-wise, i.e. a separate line for each array element. The first data string(s) are the array indices of the first, . . . , last array dimension.

- The first array index for the variable modfiles (see Section 14.7) is not given by a numeric value but by its file descriptor in string format, e.g. the string modgrid corresponds to the key id io_modgrd whose numeric value is set by the program to 3.

- If a data string contains only blanks or equals the null string, the value of the corresponding model parameter is undefined, in which case its default value is retained. When the CIF is written by the program, all variables (even defaults) are defined in the data strings.

- No error occurs if a model scalar or array parameter does not appear on any input line in which case the default value is retained.

- The characters in the string varname are case insentitive. If the CIF is written by the program, the names are always given in upper case characters.
- When a CIF is written by the program, all setup parameters are included in the file. The values are either the default settings or the re-defined values from a call to the appropriate usrdef_ routine or the ones reset by the program after a call to a reset_ routine. Only exception to this rule is the parameter cold_start which is always written as .FALSE. and can only be changed by editing the CIF manually.

9.4.2 CIF blocks

A CIF file is composed of six blocks which much be given in a specific order. Each block corresponds to a usrdef_ routine (given in parentheses below) where the parameters could be defined in absence of the CIF.

1: monitoring parameters (usrdef_init_params)
2: general model setup parameters (usrdef_mod_params)
3: parameters for the setup of time series output (usrdef_out_params)
4: parameters for the setup of time averaged output (usrdef_avr_params)
5: definitions for making harmonic analyses (usrdef_anal_freqs)
6: parameters for harmonic output (usrdef_anal_params)

The following rules apply for CIF blocks

- A CIF block is terminated by a line whose first character is the block separator '#' (the rest of the line is ignored).
- A block may be empty but the separator lines must always be there. This means that the file must contain 6 lines (including the last one) starting with a '#'. An empty block is represented by two consecutive separator lines.
- Empty blocks are written by the program in the following cases
  - block 3: no time series output (iopt_out_tsers=0)
  - block 4: no time averaged output (iopt_out_avrgd=0)
  - blocks 5 and 6: no harmonic output (iopt_out_anal=0)
- On the other hand, the above blocks may be non-empty even when the appropriate switch is zero. In that case the input lines are read by the program, but no assignment is made.

CIF special characters

The CIF utility uses the following special characters

‘,’ separates the data strings on an input line

‘=’ separates the string varname from the data strings. Must be on all input lines except those starting with a ‘!’ or ‘#’ character

‘!’ indicates the start of a comment. All characters on the input line at and beyond this character are ignored.

‘#’ block separator. Must always be the first character on a separator line.

These special characters cannot be used in the string varname or in a data string representing a string variable. For this reason the ‘,’ character, used in previous versions as separator between seconds and milliseconds in a date/time string is now replaced by a ‘:’.

9.4.3 Order of definitions

Each scalar or array parameter must be defined within its specific block. However, the order of definition within a block is, in principle, irrelevant. However, if the number of data on an input line depends on a “physical size” dimension parameter defined by another model parameter, this size parameter must appear on a previous data line.

```
COLD_START = F
LEVPROCS_INI = 3
LEVPROCS_RUN = 3
INILOG_FILE = plume1A.inilogA
RUNLOG_FILE = plume1A.runlogA
RUNLOG_COUNT = 8640
MAX_ERRORS = 50
LEVPROCS_ERR = 1
ERRLOG_FILE = plume1A.errlogA
WARNING = T
WARLOG_FILE = plume1A.warlogA
LEVTIMER = 3
TIMING_FILE = plume1A.timingA
TIMER_FORMAT = 1
#
IOPT_ADV_SCAL = 3
IOPT_ADV_TURB = 0
```

```
IOPT_ADV_TVD = 1
IOPT_ADV_2D = 3
IOPT_ADV_3D = 3
IOPT_ARRINT_HREG = 0
IOPT_ARRINT_VREG = 0
IOPT_ASTRO_ANAL = 0
IOPT_ASTRO_PARS = 0
IOPT_ASTRO_TIDE = 0
...
NC = 121
NR = 41
NZ = 20
NOSBU = 80
NOSBV = 120
NRVBU = 0
NRVBV = 1
NONESTSETS = 0
NORLXZONES = 0
NPROCS = 1
NPROCSX = 1
NPROCSY = 1
IDMASTER = 0
CSTARTDATETIME = 2003/01/03;00:00:00:000
CENDDATETIME = 2003/01/06;00:00:00:000
DELT2D = 30.
IC3D = 10
ICNODAL = 0
TIME_ZONE = 0.
NTOBCRLX = 0
ATMPRES_REF = 101325.
BDRAGCOEF_CST = 0.
BDRAGLIN = 0.
...
NCONOBC = 1
INDEX_OBC = 46
NCONASTRO = 0
ALPHA_BLACK = 0.2
ALPHA_MA = 10.
ALPHA_PP = 5.
BETA_MA = 3.33
BETA_XING = 2.
```

```
...
NORESTARTS = 1
NTRESTART = 8640
INTITLE = plume1A
OUTTITLE = plumeA
MAXWAITSECS = 3600
NOWAITSECS = 0
NRECUNIT = 4
NOSETSTSR = 4
NOSTATSTSR = 0
NOVARSTSR = 9
NOSETSAVR = 0
NOSTATSAVR = 0
NOVARSAVR = 0
NOSETSANAL = 1
NOFREQSANAL = 1
NOSTATSANAL = 0
NOVARSANAL = 7
MODFILES = inicon,1,1,U,R,plumeA.phsicsU,0,0,0,0,F,F,
MODFILES = modgrd,1,1,A,R,plumeA.modgrdA,0,0,0,0,F,F,
MODFILES = 2uvobc,1,1,U,R,plume1A.2uvobc1U,0,0,0,0,F,F,
MODFILES = 3uvobc,1,1,A,R,plume1A.3uvobc1A,0,0,0,0,F,F,
MODFILES = salobc,1,1,A,R,plume1A.salobc1A,0,0,0,0,F,F,
MODFILES = 2uvobc,2,1,U,R,plume1A.2uvobc2U,0,0,1,0,F,F,
MODFILES = 3uvobc,2,1,A,R,plume1A.3uvobc2A,0,0,1,0,F,F,
MODFILES = salobc,2,1,A,R,plume1A.salobc2A,0,0,1,0,F,F,
SURFACEGRIDS = 1,1,0,0,1000.,1000.,0.,0.
#
TSRVARS = 1,0,0,0,0,0.,C,width,Plume width,km,
TSRVARS = 2,0,0,0,0,0.,C,hfront,Plume length,km,
TSRVARS = 3,92,2,0,0,0.,C,umvel,X-component of depth-mean current,m/s,
          Depth-mean current
TSRVARS = 4,101,2,0,0,0.,C,vmvel,Y-component of depth-mean current,m/s,
          Depth-mean current
TSRVARS = 5,81,2,0,0,0.,C,zeta,Surface elevation,m,
TSRVARS = 6,93,3,0,0,0.,C,uvel,X-component of current,m/s,Current
TSRVARS = 7,102,3,0,0,0.,C,vvel,Y-component of current,m/s,Current
TSRVARS = 8,106,3,0,0,0.,C,wphys,Physical vertical velocity,m/s,
          Physical current
TSRVARS = 9,111,3,0,0,0.,C,sal,Salinity,PSU,
IVARSTSR = 1,6,7,8,9
```

```
IVARSTSR = 2,6,7,8,9
IVARSTSR = 3,6,7,8,9
IVARSTSR = 4,1,2,3,4,5
TSR3D = 1,T,U,plumeA_1.tsout3U,T,,2
TSR3D = 2,T,U,plumeA_2.tsout3U,T,,2
TSR3D = 3,T,U,plumeA_3.tsout3U,T,,2
TSR0D = 4,T,A,plumeA_4.tsout0A,T,,2
TSR2D = 4,T,U,plumeA_4.tsout2U,T,,2
TSRGPARS = 1,T,F,F,F,2003/01/03;00:00:00:000,3,0,0,1,120,1,1,40,1,20,20,1,0,
           8640,360
TSRGPARS = 2,T,F,F,F,2003/01/03;00:00:00:000,3,0,0,30,30,1,1,40,1,1,20,1,0,
           8640,360
TSRGPARS = 3,T,F,F,F,2003/01/03;00:00:00:000,3,0,0,1,120,1,5,5,1,1,20,1,0,
           8640,360
TSRGPARS = 4,T,F,F,F,2003/01/03;00:00:00:000,2,0,0,30,30,1,1,1,1,1,1,1,0,
           8640,12
#
#
INDEX_ANAL = 46
NOFREQSHARM = 1
IFREQSHARM = 1,1
#
ANALVARS = 1,92,2,0,0,0.,C,umvel,X-component of depth-mean,current,m/s,
           Depth-mean current
ANALVARS = 2,101,2,0,0,0.,C,vmvel,Y-component of depth-mean,current,m/s,
           Depth-mean current
ANALVARS = 3,81,2,0,0,0.,C,zeta,Surface elevation,m,
ANALVARS = 4,93,3,0,0,0.,C,uvel,X-component of current,m/s,Current
ANALVARS = 5,102,3,0,0,0.,C,vvel,Y-component of current,m/s,Current
ANALVARS = 6,106,3,0,0,0.,C,wphys,Physical vertical velocity,m/s,
           Physical current
ANALVARS = 7,111,3,0,0,0.,C,sal,Salinity,PSU,
IVARSANAL = 1,1,2,3,4,5,6,7
IVARSELL = 1,1,10
IVECELL2D = 1,1,2
IVECELL3D = 1,1,2
RES2D = 1,T,A,plumeA_1.resid2A,T,,2
RES3D = 1,T,A,plumeA_1.resid3A,T,,2
AMP2D = 1,1,T,A,plumeA_1.1amplt2A,T,,2
PHA2D = 1,1,T,A,plumeA_1.1phase2A,T,,2
ELL2D = 1,1,T,A,plumeA_1.1ellip2A,T,,2
```

```
ELL3D = 1,1,T,A,plumeA_1.1ellip3A,T,,2
ANALGPARS = 1,T,F,F,F,2003/01/03;06:00:00:000,3,0,0,1,120,1,1,40,1,1,20,1,0,
            8640,1440
#
```

Example 9.7: (Parts) of the CIF produced for test case ***plumeA***.

9.5 Forcing files

9.5.1 General aspects

The forcing files, used in the program code, can be divided into two categories:

1. Files, which have no time dependence, are called "initialisation" files and may contain the following information:
   - definitions of a domain decomposition
   - model grid and bathymetry
   - definitions of 2-D external grids
   - locations of the open boundary points of a nested sub-grid
   - specifiers for 2-D and 3-D open boundary conditions or 1-D water column forcing
   - specifiers for nesting
   - definition of relaxation zones.
2. Time series files provide forcing data at one or more specific times, given as a sequence of time record(s).
   - initial conditions[1]
   - open boundary data
   - 1-D water column forcing data
   - 2-D and 3-D open boundary data written for nested sub-grids
   - data defined on a 2-D external (meteorological, SST) grid.

The standard structure of forcing files is composed of

[1]Initial conditions can be given at one or more specific times.

- a metadata (header) section with global information about the file (called "global attributes" in netCDF language) and information about the data variables within the file ("variable attributes" in netCDF language)

- a data section with the values of the data. The time coordinate (if present) is considered as an additional data variable.

The general structure of the file is then[2]

```
[Names and values of dimensions]
Global attributes
...
[Attributes of the time coordinate]
Attributes of variable 1
...
Attributes of variable 2
...
Attributes of variable n
[First data time]
Values of variables 1
...
Values of variables n
[Second data time
...]
```

Example 9.8: General layout of a forcing file.

The time coordinate and data time(s) need, obviously, only to be present in case of time series files. Note that the data times must be stored in chronological order, but may be given at non-regular time intervals. The detailed formats of forcing files are discussed in the subsections below.

As mentioned in Section 8.1.4, the properties (or attributes) of data files are stored into the derived type variable FileParams (see Example 8.7 for a full list of file attributes). The attributes of the forcing files are stored into the 3-D derived type array modfiles:

```
TYPE (FileParams), DIMENSION(MaxIOTypes,MaxIOFiles,2) :: modfiles
```

where

[2]The lines in [ ] are not always present. For example, metadata and data for the time coordinate variable are only needed in case of a forcing file containing time series.

MaxIOTypes is the maximum number of file descriptors

MaxIOFiles is the maximum allowed number of files per file descriptor.

An element of the array modfiles can be generically written as modfiles(iddesc,ifil,iotype) where

iddesc denotes the file descriptor key id. The program name of the key id has the form io_filedesc where filedesc is one of the descriptor strings given in Section 9.2.4. For example, io_2uvobc is the key id for all forcing files related to 2-D open boundary conditions.

ifil is the file number. In some cases, this number is always 1. For example, the input data for defining the model grid are stored in one file with key id io_modgrd. On the other hand, if a main grid contains several nested sub-grids, a data file has to be written for each requested parameter and each sub-grid. The parameter is represented by its key id, while the file number denotes the number of the sub-grid. A similar approach is followed for open boundary data. For example, the temperature profiles at open boundaries can be obtained from several data files having the same key id io_tmpobc.

iotype equals 1 for an input and 2 for an output file. All forcing files used by the model are input files, except for the files written for sub-grid nesting. However, the program provides the possibility to re-write the data obtained from an input file to a separate output file in COHERENS format. This is further discussed in Section 14.7.

In the current implementation, the global attributes of the files are defined by components of the derived type variable modfiles and are given below. A (*) at the end of the description means that the attribute can be defined by the user.

status* Status of the file

- '0' : The file is not activated.
- 'N' : The file is activated but its contents are defined by the user in a usrdef_ routine. The user needs to decide whether the data are obtained from some external file or that the file only exists virtually and the data are defined without making a file connection. The option is only available for input files.
- 'R' : The file is activated and its contents are read from a data file in COHERENS standard format. The option is only allowed for input data (iotype=1).

'W': The data are written in COHERENS standard format. This is always the case for nesting data. The option can be used to re-write data, previously obtained in a user format. The file can then be used in a subsequent simulation with the 'R' status. Since the file is created for output, the iotype index must be 2.

form* Format of the data file.

'A': ASCII

'U': unformatted binary

'N': netCDF

filename* Name of the file. If the file status is 'R' or 'W' and the name of the file is not defined by the user, a default name is given (see Section 9.2). If the status is 'N', the name is either defined by the user or unknown. In the latter case the file name is set to 'N' (unknown) which may mean that the data are not obtained from a file.

info* An info ('I') file is produced with the metadata information only.

end_type* Switch to decide what action needs to be taken when an end of file condition occurs during a read. See Section 14.7.2 for details.

0: The program aborts with an error message

1: The program continues, no further attempt will be made to read data.

2: The program continues, a next attempt to read the data will be made after nowaitsecs seconds.

tlims(1:3)* Start/end/step time indices (i.e. times measured in units of the 2-D time step delt2d). The data will be updated after ABS(tlims(3)) ×delt2d seconds. If tlims(3)>0, time interpolation will be performed. If tlims(3)<0, the data values are set to the one obtained from the latest data record earlier than the current time.

iunit File unit number. This parameter is set internally and cannot be reset by the user.

iostat The I/O status of the file.

-1: An error occurred when the program attempted to open the file.

0 : The file is not open.

1 : The file is open and the file pointer is located at the start or before the end of the file.

2 : The file pointer is located at the end of the file (i.e. an end of file condition will occur on a next read).

3 : An end of file condition did occur.

nocoords Number of coordinate arrays within the file, equal to 0 for an initialisation file and 1 for a time series file.

novars Number of (non-coordinate) data variables within the file.

filedesc A string with a description of the file.

timeid netCDF variable id of the time coordinate.

The variable attributes are stored into a temporary derived type array

```
TYPE (VariableAtts), DIMENSION(nocoords+novars) :: varatts
```

where nocoords and novars are the global attributes stored in the modfiles array. The following attributes are defined:

f90_name A string with the FORTRAN name of the variable as used in the program. Maximum length is set by the system parameter lenname.

long_name A string with a description of the variable. Maximum length is set by the system parameter lendesc.

vector_name If the variable represents a vector component, a string with a description of the vector. Its value must be the same for all components of the same vector. Maximum length is set by the system parameter lendesc.

units A string describing the variable's unit. The string has a format recognised by UNIDATA's Udunits package (UDUNITS, 1997) which can be considered as an internationally recognised standard. Maximum length is set by the system parameter lenunit.

data_type The type of variable as given in the second column of Table 8.1 (e.g. real_type for a REAL variable).

ivarid The variable's key id.

nrank If the variable is an array, the rank of the array. Otherwise, the variable is a scalar and the rank is zero.

shape If the variable is an array, a vector with the array size(s) in each dimension. In case of a scalar the shape vector has one element with the value 1.

9.5.2 Data contents of forcing files

Table 9.3 provides a general listing of the data contents for each type of forcing file. Note that the exact number of data variables depends on how the model has been set up. A detailed discussion is given in the Part IV.

The integer parameters used in the table for array dimensioning have the following meaning:

nprocs	number of processes in case the program is applied in parallel mode
ncloc	X-dimension of the local grid
nrloc	Y-dimension of the local grid
nc	X-dimension of the global (computational) grid
nr	Y-dimension of the global (computational) grid
nz	number of vertical layers
nhalo	halo size (=2)
nf	number of sediment fractions
nconobc	number of tidal constituents at open boundaries
nconastro	number of astronomical constituents used for the tidal force
nobu	number of open boundary points at U-nodes
nobv	number of open boundary points at V-nodes
nodat	number of data (e.g. discharge locations)
numdis	number of discharge locations
numdry	number of dry cells
numthinu	number of thin dams at U-nodes
numthinv	number of thin dams at V-nodes
numwbaru	number of weirs/barriers at U-nodes
numwbarv	number of weirs/barriers at V-nodes
n1dat	X-dimension of an external 2-D data grid
n2dat	Y-dimension of an external 2-D data grid
nonestsets	number of nested sub-grids
nhdat	number of sub-grid open boundary locations in the horizontal used for nesting of a sub-grid
nzdat	number of vertical levels at sub-grid open boundary locations used for nesting of a sub-grid

nonodes number of "nodal" grids used for interpolation in nesting procedures (see Section 17.3.3)

novars number of data variables (meteorological, wave, discharge) variables in a data file

numprofs number of vertical profiles in a 3-D open boundary forcing file

nofiles number of data files (plus one) for open boundary or 1-D surface forcing

norlxzones number of zones for application of the relaxation scheme near open boundaries

Table 9.3: Data contents for each type of input forcing file. In the last column 'R', 'I', 'C', 'D' denote respectively real, integer, character and derived type data.

key id	file number	variable	shape	type
io_mppmod	1	nc1procs	nprocs	I
		nc2procs	nprocs	I
		nr1procs	nprocs	I
		nr2procs	nprocs	I
io_inicon	1	udvel	(1-nhalo:ncloc+nhalo,1-nhalo:nrloc+nhalo)	R
		vdvel	(1-nhalo:ncloc+nhalo,1-nhalo:nrloc+nhalo)	R
		zeta	(0:ncloc+1,0:nrloc+1)	R
		uvel	(1-nhalo:ncloc+nhalo,1-nhalo:nrloc+nhalo,nz)	R
		vvel	(1-nhalo:ncloc+nhalo,1-nhalo:nrloc+nhalo,nz)	R
		wvel	(0:ncloc,0:nrloc,nz+1)	R
		temp	(1-nhalo:ncloc+nhalo,1-nhalo:nrloc+nhalo,nz)	R
		sal	(1-nhalo:ncloc+nhalo,1-nhalo:nrloc+nhalo,nz)	R
		tke	(1-nhalo:ncloc+nhalo, 1-nhalo:nrloc+nhalo,nz+1)	R
		zlmix	(1-nhalo:ncloc+nhalo, 1-nhalo:nrloc+nhalo,nz+1)	R
		dissip	(1-nhalo:ncloc+nhalo, 1-nhalo:nrloc+nhalo,nz+1)	R
		bdragcoefatc	(0:ncloc,0:nrloc)	R
		zroughatc	(0:ncloc,0:nrloc)	R
		fnode_obc	nconobc	R
		phase_obc	nconobc	R
		fnode_astro	nconastro	R
		phase_astro	nconastro	R
		obcsalatu	(nobu,nz,0:2)	R

(*Continued*)

Table 9.3: *Continued*

		obcsalatv	(nobv,nz,0:2)	R
		obctmpatu	(nobu,nz,0:2)	R
		obctmpatv	(nobv,nz,0:2)	R
		obc2uvatu	(nobu,2)	R
		obc2uvatv	(nobv,2)	R
		obc3uvatu	(nobu,nz,2)	R
		obc3uvatv	(nobv,nz,2)	R
io_inicon	2	cvol	(1-nhalo:ncloc+nhalo,1-nhalo:nrloc+nhalo,nz,nf)	R
		zroughatc_sed	(0:ncloc,0:nrloc)	R
		bed_fraction	(0:ncloc,0:nrloc,nf)	R
		obcsedatu	(nobu,nz,0:2,nf)	R
		obcsedatv	(nobv,nz,0:2,nf)	R
io_modgrd	1	gxcoordglb	(nc,nr)	R
		gycoordglb	(nc,nr)	R
		gscoordglb	(nc-1,nr-1,nz+1)	R
		depmeanglb	(nc-1,nr-1)	R
		iobu	nobu	I
		jobu	nobu	I
		iobv	nobv	I
		jobv	nobv	I
io_metgrd	1	xcoord	(n1dat,n2dat)	R
		ycoord	(n1dat,n2dat)	R
	or	surfgridglb	(nc,nr)	D
io_sstgrd	1	xcoord	(n1dat,n2dat)	R
		ycoord	(n1dat,n2dat)	R
	or	surfgridglb	(nc,nr)	D
io_wavgrd	1	xcoord	(n1dat,n2dat)	R
		ycoord	(n1dat,n2dat)	R
	or	surfgridglb	(nc,nr)	D
io_nstgrd	1:nonestsets	xcoord	nhdat	R
		ycoord	nhdat	R
	or	hnests	(nhdat,nonodes)	D
		zcoord	(nhdat,nzdat)	R
io_sedspc	1	dp	nf	R
		rhos	nf	R
		tau_cr_cst	nf	R
		ws_cst	nf	R

(*Continued*)

Table 9.3: *Continued*

io_1uvsur	1	gxslope_amp	nconobc	R
		gxslope_pha	nconobc	R
		gyslope_amp	nconobc	R
		gyslope_pha	nconobc	R
		zeta_amp	nconobc	R
		zeta_pha	nconobc	R
	2:nofiles	ciodatetime	–	C
		data1d	novars	R
io_2uvobc	1	ityp2dobu	nobu	I
		iloczobu	nobu	I
		itypintobu	nobu	I
		ityp2dobv	nobv	I
		iloczobv	nobv	I
		itypintobv	nobv	I
		no2dobc	2:nofiles	I
		iobc2dtype	2:nofiles	I
		index2dobc	(nobu+nobv:2:nofiles)	I
		ud2obu_amp	(nobu,nconobc)	R
		zetobu_amp	(nobu,nconobc)	R
		ud2obu_pha	(nobu,nconobc)	R
		zetobu_pha	(nobu,nconobc)	R
		vd2obv_amp	(nobv,nconobc)	R
		zetobv_amp	(nobv,nconobc)	R
		vd2obv_pha	(nobv,nconobc)	R
		zetobv_pha	(nobv,nconobc)	R
	2:nofiles	ciodatetime	–	C
		data2d	(nodat,novars)	R
io_3uvobc	1	itypobu	nobu	I
		iprofobu	nobu	I
		itypobv	nobv	I
		iprofobv	nobv	I
		noprofsd	2:nofiles	I
		indexprof	(nobu+nobv,2:nofiles)	I
		iprofrlx	norlxzones	I
	2:nofiles	ciodatetime	–	C
		psiprofdat	(numprofs,nz)	R
io_salobc	1	itypobu	nobu	I
		iprofobu	nobu	I

(*Continued*)

Table 9.3: *Continued*

		itypobv	nobv	I
		iprofobv	nobv	I
		noprofsd	2:nofiles	I
		indexprof	(nobu+nobv,2:nofiles)	I
		iprofrlx	norlxzones	I
	2:nofiles	ciodatetime	–	C
		psiprofdat	(numprofs,nz)	R
io_tmpobc	1	itypobu	nobu	I
		iprofobu	nobu	I
		itypobv	nobv	I
		iprofobv	nobv	I
		noprofsd	2:nofiles	I
		indexprof	(nobu+nobv,2:nofiles)	I
		iprofrlx	norlxzones	I
	2:nofiles	ciodatetime	–	C
		psiprofdat	(numprofs,nz)	R
io_sedobc	1	itypobu	nobu	I
		iprofobu	nobu,nf	I
		itypobv	nobv	I
		iprofobv	nobv,nf	I
		noprofsd	2:nofiles	I
		indexprof	(nf*(nobu+nobv),2:nofiles)	I
		indexvar	(nf*(nobu+nobv),2:nofiles)	I
		iprofrlx	norlxzones	I
	2:nofiles	ciodatetime	–	C
		psiprofdat	(numprofs,nz)	R
io_rlxobc	1	inodesrlx	2	I
		idirrlx	norlxzones	I
		ityprlx	norlxzones	I
		iposrlx	norlxzones	I
		jposrlx	norlxzones	I
		ncrlx	norlxzones	I
		nrrlx	norlxzones	I
io_nstspc	1	nestcoords	nonestsets	I
		nohnstglbc	nonestsets	I
		nohnstglbu	nonestsets	I
		nohnstglbv	nonestsets	I
		novnst	nonestsets	I

(*Continued*)

Table 9.3: *Continued*

		inst2dtype	nonestsets	I
io_metsur	1	ciodatetime	–	C
		surdata	(n1dat,n2dat,novars)	R
io_sstsur	1	ciodatetime	–	C
		surdata	(n1dat,n2dat,1)	R
io_wavsur	1	ciodatetime	–	C
		surdata	(n1dat,n2dat,novars)	R
io_drycel	1	idry	numdry	I
		jdry	numdry	I
io_thndam	1	ithinu	numthinu	I
		jthinu	numthinu	I
		ithinv	numthinv	I
		jthinv	numthinv	I
io_weibar	1	iwbaru	numwbaru	I
		jwbaru	numwbaru	I
		oricoefu	numwbaru	R
		oriheightu	numwbaru	R
		orisillu	numwbaru	R
		wbarcoefu	numwbaru	R
		wbarcrestu	numwbaru	R
		wbarmodlu	numwbaru	R
		iwbarv	numwbarv	I
		jwbarv	numwbarv	I
		oricoefv	numwbarv	R
		oriheightv	numwbarv	R
		orisillv	numwbarv	R
		wbarcoefv	numwbarv	R
		wbarcrestv	numwbarv	R
		wbarmodlv	numwbarv	R
io_disspc	1	kdistype	numdis	I
		mdistype	numdis	I
io_disvol	2:nofiles	ciodatetime	–	C
		disdata	(nodat,novars)	R
io_discur	2:nofiles	ciodatetime	–	C
		disdata	(nodat,novars)	R
io_dissal	2:nofiles	ciodatetime	–	C
		disdata	(nodat,novars)	R
io_distmp	2:nofiles	ciodatetime	–	C

(*Continued*)

Table 9.3: *Continued*

disdata	(nodat,novars)	R

9.5.3 Standard format of forcing files

9.5.3.1 ASCII files

The ASCII format is illustrated with two examples. They are taken from a case study for the North Sea (not discussed in the current version of the manual). Examples 9.9 and 9.10 show the contents of the files *nsp89.2uvobc1A* and *nsp89.metsurA*. The line numbers have been added in the header section for illustrative purposes only and do not appear in the actual file.

```
1 :           1
2 :V2.0
3 :2010/06/17;09:22:34
4 :Type of open boundary conditions for 2-D mode
5 :           0
6 :          17
7 :         620           3           1          29
8 :ityp2dobu
9 :Type of 2-D open boundary condition at U-nodes
10:-
11:      607        3        1       29
12:iloczobu
13:Position of elevation points with respect to U-open boundaries
14:-
15:      616        3        1       29
16:itypintobu
17:Switch to allow momentum advection near U-open boundaries
18:-
19:      621        3        1      111
20:ityp2dobv
21:Type of 2-D open boundary condition at V-nodes
22:-
23:      608        3        1      111
24:iloczobv
25:Position of elevation points with respect to V-open boundaries
26:-
27:      617        3        1      111
28:itypintobv
```

```
29:Switch to allow momentum advection near V-open boundaries
30:-
31:      624        3        1        2
32:no2dobc
33:Number of input data per input file
34:-
35:      612        3        1        2
36:iobc2dtype
37:Type 2-D open boundary data input
38:-
39:      611        3        2      140        2
40:index2dobc
41:Map of data points to open boundary locations
42:-
43:      637        5        2       29        9
44:ud2obu_amp
45:Amplitude of X-component of depth-integrated current at U-open
   boundaries
46:m^2/s
47:      648        5        2       29        9
48:zetobu_amp
49:Amplitude of surface elevation at U-open boundaries
50:m
51:      638        5        2       29        9
52:ud2obu_pha
53:Phase of X-component of depth-integrated current at U-open boundaries
54:radian
55:      649        5        2       29        9
56:zetobu_pha
57:Phase of surface elevation at U-open boundaries
58:m
59:      640        5        2      111        9
60:vd2obv_amp
61:Amplitude of Y-component of depth-integrated current at V-open
   boundaries
62:m^2/s
63:      651        5        2      111        9
64:zetobv_amp
65:Amplitude of surface elevation at V-open boundaries
66:m
67:      641        5        2      111        9
```

```
68:vd2obv_pha
69:Phase of Y-component of depth-integrated current at V-open boundaries
70:radian
71:      652        5        2      111        9
72:zetobv_pha
73:Phase of surface elevation at V-open boundaries
74:m
ityp2dobu
         11         11 ...
iloczobu
          1          1 ...
itypintobu
          0          0 ...
ityp2dobv
         11         11 ...
iloczobv
          1          1 ...
itypintobv
          0          0 ...
no2dobc
        127         13
iobc2dtype
          1          3
index2dobc
          1          2  ...
ud2obu_amp
   0.4421976       0.5311887 ...
zetobu_amp
   0.3082999E-01   0.3055999E-01 ...
ud2obu_pha
    4.054574        4.223073 ...
zetobu_pha
    5.249485        5.269729 ...
vd2obv_amp
   0.5446934       0.7153571 ...
zetobv_amp
   0.4053500E-01   0.3966500E-01 ...
vd2obv_pha
    2.541209        2.624729 ...
zetobv_pha
   0.2956865       0.3112219 ...
```

Example 9.9: Contents of the file *nsp89.2uvobc1A* in standard ASCII COHERENS format.

- Lines 1–6 give the values of the attributes:

header_type	the type of header which (in the current version) is always 1 for a forcing file
coherens_version	the current program version number, which is the same for all forcing files
creation_date	the exact date and time when the file was created
filedesc	a description of the file
nocoords	the number of coordinate variables (as defined in modfiles)
novars	the number of (non-coordinate) variables (as defined in modfiles)

- In case of a time series file, the next four lines list the attributes of the time coordinate (see Example 9.10).
- The remaining lines 7–74 show the attributes of the data variables.
- The attributes of each variable are given on four lines
  - line 1: the attributes ivarid, data_type, nrank, shape, giving 3+nrank integer parameters on the input line
  - line 2: the f90_name attribute
  - line 3: the long_name attribute
  - line 4: the units attribute
- The total number of header lines is then given by 6+4*(nocoords+novars).
- The data section gives the values of the variables in the order in whichv they have been defined in the header section.
  - In case of an initialisation file, the name of the variable is written on one line, followed by its values.
  - In case of a time series file, the value of the time coordinate is written first using the date/time string format (see Section 8.2.2), the data values are written as in the previous case except that the data values are preceded by an empty line (for illustration this line is presented in the example by the variable's name in [ ]). The

line next to the values of the last variable is the date/time of the next time record, ...

The contents of an ASCII forcing file are to be read sequentially, i.e. line by line.

- The lines in the header either contain a character string or integer parameters which makes it easier to read them using the character ('A') format (strings) or free (*) format (integers). The header information does not provide additional information for the program, since the attributes are already known internally, but are used for error checking only. For users who like to read the data from some external program, the metadata provides useful information (number, rank and shape of the data variables).

- The data values (except the date/time string in time series files) are either of type INTEGER or REAL and are read in FORTRAN array order, e.g.

  ```
  READ (iunit,IntegerFormat) itypintobu
  READ (iunit,RealFormat) uwindatc
  ```

  where the string format specifications IntegerFormat and RealFormat are system parameters, defined in *syspars.f90*. Values are

  ```
  IntegerFormat='(50I11)'; RealFormat='(50G16.7)'
  ```

  It is advised not to change these values (except for the repeat specification) since they allow to represent the data with the highest possible precision.

```
1 :           1
2 :V2.0
3 :2010/06/17;09:22:34
4 :Meteo input surface data
5 :           1
6 :           7
7 :      952        1        2       23       -1
8 :time
9 :Time
10:date/time
11:      410        5        2       50       28
```

```
12:uwindatc
13:X-component of surface wind
14:m/s
15:      411        5        2       50       28
16:vwindatc
17:Y-component of surface wind
18:m/s
19:      402        5        2       50       28
20:atmpres
21:Atmospheric pressure
22:N/m^2
23:      401        5        2       50       28
24:airtemp
25:Air temperature
26:degC
27:      407        5        2       50       28
28:relhum
29:Relative humidity
30:-
31:      403        5        2       50       28
32:cloud_cover
33:Cloud cover
34:-
35:      404        5        2       50       28
36:evapminprec
37:Evaporation minus precipitation rate
38:kg/m^2/s
1989/01/01;00:00:00,000
[uwindatc]
   0.8044688E-06  -0.2852140 ...
[vwindatc]
    9.202050        8.167472 ...
[atmpres]
    102672.9        102768.9 ...
[airtemp]
    13.85400        13.92300 ...
[relhum]
   0.9958699       0.9328000 ...
[cloud_cover]
   0.6250000       0.6250000 ...
[evapminprec]
```

```
   0.7811863E-05   0.7811863E-05 ...
1989/01/01;03:00:00,000
[uwindatc]
  -0.3340597      -0.5992587 ...
[vwindatc]
    9.566232        8.569798 ...
[atmpres]
    102548.8        102661.5 ...
[airtemp]
    13.95900        13.99600 ...
[relhum]
   0.7983300       0.8664200 ...
[cloud_cover]
   0.6250000       0.6250000 ...
[evapminprec]
   0.7811863E-05   0.7811863E-05 ...
1989/01/01;06:00:00,000
...
```

Example 9.10: Contents of the file *nsp89.metsurA* in standard ASCII COHERENS format.

9.5.3.2 unformatted binary files

When the unformatted binary or ASCII format is selected, the same metadata and data are written in the forcing file. Differences are that in case of a binary format:

- String attributes, numerical attributes and data values are written in binary format. The form depends on the internal binary presentation of each (character, integer, real) data type. Common types are known as 'NATIVE', 'LITTLE_ENDIAN', 'BIG_ENDIAN', 'IBM'.
- The file can be read only on machines which use the same binary presentation[3].
- Reading of the file or conversion to a readable format can only be performed using some external (post-processing) program.
- The data are written sequentially as in the ASCII case but without a format specification.

[3]Some compilers provide a CONVERT specifier in a FORTRAN OPEN call allowing to make a conversion between two different binary formats.

- Data values are no longer preceded by a blank line.
- Prime advantage of this format is that each numerical value (defined in single precision) only uses (in most cases) only 4 bytes of disk space, reducing the required storage space by a factor 3 to 4 compared to the ASCII format.

The program provides an utility (using the info file attribute) to view the metadata contents of a binary file without additional programming tools. If the info attribute is set to .TRUE., the program will create an additional "info" file with all metadata information in ASCII. An example is given below for the rhone test case.

```
header type:                    1
coherens version:              V2.0
creation date:                 2010/06/18;16:49:53
file description:              Model grid
nocoords:                       0
novars:                         6
   32 5 1        31
gsigcoord
Sigma coordinates on uniform grid
-
  105 5 2       108        50
depmeanglb
Global mean water depth at C-nodes
m
   43 3 1        65
iobu
Global X-index of U-open boundaries
-
   51 3 1        65
jobu
Global Y-index of U-open boundaries
-
   45 3 1       110
iobv
Global X-index of V-open boundaries
-
   53 3 1       110
jobv
Global Y-index of V-open boundaries
```

–

Example 9.11: Contents of *rhoneA.modgrdI* info file, giving all metadata information included in the rhone grid file.

9.5.3.3 netCDF files

The netCDF format is binary and non-sequential. The data are organised in records which can be read by specifying the appropriate record number as argument to a netCDF routine call. Moverover, the contents of the file, or part of the contents (metadata only, values of one or more data variables) can easily by converted to an ASCII format, which is the so-called CDL (network Common Data form Language) format. For a full description of CDL, see Russ *et al.* (2004).

- Data and metadata are stored, in a platform-independent way, as (non-sequential) internal records.
- Reading and writing is performed by netCDF routine calls described in the netCDF manual (Pincus & Rew, 2008). Specific routines are available in the program to write metadata and data into a standard COHERENS format, similar to the one used in the ASCII and unformatted binary formats.
- An alias (starting with cf_90) is defined in the code for each netCDF routine call (starting with NF90_). The aim is a more efficient implementation of future netCDF versions.
- Although the file is in a compressed binary format, the contents or part of the contents of a netCDF file can be converted to ASCII form with the `ncdump` utility:

  ```
  ncdump file.cdf
  ncdump -h file.cdf
  ncdump -v var file.cdf
  ```

  The first form rewrites the file *file.cdf* to standard output in ASCII format. The second rewrites the metadata section only, the third form rewrites the metadata section and the values of the variable var.

.

Example 9.12 shows the metadata file in CDL format of the initial condition file for the North Sea case study. Differences with the ASCII a binary formats are:

- Variable dimensions must be defined with a given name.
- The f90_name, nrank and shape attributes are no longer explicitly defined, but provided implicitly when a netCDF variable is defined with the NF_90_def_var library call.
- Global attributes are the same as before except for the cdfversion attribute giving the current netCDF version used by the model and the timerec attribute giving the number of time records in the file.

Reading and writing of netCDF metadata and data is performed using the routines of the netCDF library. A detailed description is found in the netCDF FORTRAN 90 manual (Pincus and Rew, 2008).

```
netcdf nsp89
dimensions:
tlendim = 23 ;
T = UNLIMITED ; // (2 currently)
X002 = 141 ;
Y002 = 128 ;
        ...
Z020 = 2 ;
variables:
char time(T, tlendim) ;
time:ivarid = 952 ;
time:long_name = "Time" ;
time:units = "date/time" ;
float udvel(T, Y002, X002) ;
udvel:ivarid = 157 ;
udvel:long_name = "X-component_of_depth-integrated_current" ;
udvel:units = "m^2/s" ;
float vdvel(T, Y003, X003) ;
vdvel:ivarid = 166 ;
vdvel:long_name = "Y-component_of_depth-integrated_current" ;
vdvel:units = "m^2/s" ;
float zeta(T, Y004, X004) ;
zeta:ivarid = 113 ;
zeta:long_name = "Surface_elevation" ;
zeta:units = "m" ;
float uvel(T, Z005, Y005, X005) ;
uvel:ivarid = 162 ;
uvel:long_name = "X-component_of_current" ;
```

```
uvel:units = "m/s" ;
float vvel(T, Z006, Y006, X006) ;
vvel:ivarid = 171 ;
vvel:long_name = "Y-component_of_current" ;
vvel:units = "m/s" ;
float wvel(T, Z007, Y007, X007) ;
wvel:ivarid = 176 ;
wvel:long_name = "Transformed_vertical_velocity" ;
wvel:units = "m/s" ;
float temp(T, Z008, Y008, X008) ;
temp:ivarid = 205 ;
temp:long_name = "Temperature" ;
temp:units = "degC" ;
float sal(T, Z009, Y009, X009) ;
sal:ivarid = 204 ;
sal:long_name = "Salinity" ;
sal:units = "PSU" ;
float tke(T, Z010, Y010, X010) ;
tke:ivarid = 304 ;
tke:long_name = "Turbulent_kinetic_energy" ;
tke:units = "J/kg" ;
float fnode_obc(T, X011) ;
fnode_obc:ivarid = 353 ;
fnode_obc:long_name = "Nodal_factors_of_tidal_constituents_at_open_boundaries" ;
fnode_obc:units = "-" ;
float phase_obc(T, X012) ;
phase_obc:ivarid = 360 ;
phase_obc:long_name = "Astronomical_phases_at_open_boundaries" ;
phase_obc:units = "radian" ;
float obcsalatu(T, Z013, Y013, X013) ;
obcsalatu:ivarid = 625 ;
obcsalatu:long_name = "Storage_array_for_salinity_at_U-open_boundaries" ;
obcsalatu:units = "PSU" ;
float obcsalatv(T, Z014, Y014, X014) ;
obcsalatv:ivarid = 626 ;
obcsalatv:long_name = "Storage_array_for_salinity_at_V-open_boundaries" ;
obcsalatv:units = "PSU" ;
float obctmpatu(T, Z015, Y015, X015) ;
obctmpatu:ivarid = 627 ;
obctmpatu:long_name = "Storage_array_for_temperature_at_U-open_boundaries" ;
obctmpatu:units = "degC" ;
```

```
float obctmpatv(T, Z016, Y016, X016) ;
obctmpatv:ivarid = 628 ;
obctmpatv:long_name = "Storage_array_for_temperature_at_V-open_boundaries" ;
obctmpatv:units = "degC" ;
float obc2uvatu(T, Y017, X017) ;
obc2uvatu:ivarid = 629 ;
obc2uvatu:long_name = "Storage_array_for_2-D_mode_at_U-open_boundaries" ;
obc2uvatu:units = "-" ;
float obc2uvatv(T, Y018, X018) ;
obc2uvatv:ivarid = 630 ;
obc2uvatv:long_name = "Storage_array_for_2-D_mode_at_V-open_boundaries" ;
obc2uvatv:units = "-" ;
float obc3uvatu(T, Z019, Y019, X019) ;
obc3uvatu:ivarid = 631 ;
obc3uvatu:long_name = "Storage_array_for_3-D_mode_at_U-open_boundaries" ;
obc3uvatu:units = "m/s" ;
float obc3uvatv(T, Z020, Y020, X020) ;
obc3uvatv:ivarid = 632 ;
obc3uvatv:long_name = "Storage_array_for_3-D_mode_at_V-open_boundaries" ;
obc3uvatv:units = "m/s" ;

// global attributes:
:header_type = 1 ;
:coherens_version = "V2.0" ;
:creation_date = "2010/06/21;17:49:36" ;
:filedesc = "Physical initial conditions" ;
:cdfversion = "3̈.6.2¨ of" ;
:nocoords = 1 ;
:novars = 19 ;
:timerec = 2 ;
data:

 time =
  "1989/01/01;00:00:00,000",
  "1989/12/25;00:00:00,000" ;
 udvel =
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -0.08447912, 0.3973033, 0, ...
 vdvel =
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
 zeta =
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.607709, 1.618565, 1.631894, 0, ...
```

```
uvel =
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -0.006357468, 0.006396886, 0, ...
vvel =
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
wvel =
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
temp =
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 10.34, 10.34, 10.34, 0, 0, 0, ...
sal =
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 35.4, 35.4, 35.4, 0, 0, 0, 35.33, ...
tke =
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.114186e-05, 5.660635e-06, ...
fnode_obc =
 1.165938, 1.165991, 1.10667, 0.9655771, 0.9624547, 0.9655771, ...
phase_obc =
 5.17733, 2.496852, 0.2368603, 5.49324, 5.436536, 2.75393, 3.223492, ...
obcsalatu =
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
obcsalatv =
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
obctmpatu =
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
obctmpatv =
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
obc2uvatu =
 -20.18008, -20.35524, -18.70897, -17.60509, -17.31932, -16.37582, ...
obc2uvatv =
 -0.6839569, 2.442981, 1.570055, 0.996551, 1.981653, 2.819804, ...
obc3uvatu =
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
obc3uvatv =
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
```

Example 9.12: Metadata of the initial condition file for the North Sea test case in netCDF CDL format.

9.6 User output files

9.6.1 General aspects

The procedures for defining user output are discussed at length in Chapter 20. Only a summary, needed to understand the formatting of the data file discussed in the next subsection, will be given here.

The following forms of user output can be defined

- Time series: output of model data at regular time intervals.
- Time averaged: output of model data averaged over a specific period and repeated at regular time intervals.
- Harmonic: output of harmonically analysed model data over a specific period and repeated at regular time intervals. Output data may consist of residuals, amplitudes, phases and elliptic parameters.

User output is organised by defining so-called "output sets". Each set is characterised by a specific spatial and time resolution and a number of selected variables. Each output set (except harmonic output) may contain 3 files at most:

- 0-D file: "globally" evaluated data (e.g. domain averaged temperature) or data without a spatial dimension (e.g. width of a river plume)
- 2-D file: 2-D (horizontal) data without a vertical dimension (e.g. water levels)
- 3-D file: 3-D data having both horizontal and vertical dimensions

For each file a derived type variable is defined of TYPE(FileParams). The following attributes can be defined by the user:

defined	Activates or disactivates the file if set to .TRUE. or .FALSE. respectively.
form	Format of the output file 'A': ASCII 'U': unformatted binary 'N': netCDF
info	An info ('I') file is produced with the metadata information if .TRUE.

header_type No header (metadata) will be written if set to zero. The option is not available for netCDF files.

The attributes iunit, iostat, nocoords, novars and timeid are defined internally and have the same meaning as for forcing files (see Section 9.5.1).

The user is free to choose the output grid which may be the full model grid, a subsection of the model grid or a completely irregular grid as is the case for data at station locations. In case of a 2-D or 3-D grid, the horizontal grid may be taken as one point so that the 2-D data reduce to one data value and the 3-D data to one vertical profile per variable and time step.

The attributes of the output grid are stored in a derived type variable of TYPE(OutGridParams). The definitions of its attributes are given in Section 20.1.1.3, but repeated here for clarity.

gridded If .TRUE. (default), the output data are defined on a sub-grid of the model grid (or the whole model grid). If .FALSE., the data are taken at a number of irregularly spaced locations ("station" data) defined by the user.

grid_file If .TRUE., the coordinates of the output grid will be written on a separate output file. Otherwise, they are written within the data file itself (default).

land_mask A land mask will be applied if .TRUE. and gridded=.TRUE.. This means that the gridded data will be stored as a vector excluding land points. Advantage may be a significant reduction in disk space. Disadvantage is that the data need to be restored on the original output grid by a postprocessing program. Default is .FALSE.

time_grid If .TRUE., the data grid is taken as time-dependent (since the vertical positions in a σ-grid depend on time). Surface elevations will be written as an additional coordinate variable at each time step. Default is .FALSE., in which case the vertical positions are referred to the mean water level. This option is only available for time series output.

time_format Format of the time coordinate.

0: date/time in string format (default value)

1: seconds

2: minutes

3: hours

4: days

5: months

6: years

7: date in years

Cases 1-6 are numerical formats. Cases 0 and 7 are absolute times, while the others are times relative to the reference date/time refdate.

refdate — Reference date/time for calculating relative times. If not given, refdate equals the first output date/time, rounded to the nearest minute, hour, ... depending on the value of time_format.

tlims — Start/end/step time indices for data output. This means that output will be written at intervals of delt2d*tlims(3) seconds.

nodim — Dimension of the output grid (0/2/3). For example, the dimension must be set to 3 to enable 3-D output.

nostats — Number of data stations in case of non-gridded (station) data.

xlims — Start/end/step X-index in case of gridded data. This defines the output sub-grid in the X-direction. (Option not available for 0-D or station output).

ylims — Start/end/step Y-index in case of gridded data. This defines the output sub-grid in the Y-direction. (Option not available for station or 0-D output).

zlims — Start/end/step Z-index in case of gridded data. This defines the output sub-grid in the Z-direction and applies for gridded and non-gridded output. (Option only available for 3-D output).

Attributes of variables and coordinate arrays of the output grid are stored in a derived type variable of TYPE(VariableAtts) (see Section 9.5.1). The program makes distinction between two types of variables:

- Standard variables whose attributes are known to the program. In that case the variable's key id is supplied by the user and only a limited number of definitions need to be made. A number of operators (minimum, maximum or mean value, value at a given vertical level or depth) can be applied to the output data. The output values are automatically generated by the program.
- User-defined variables in which case all attributes and output values are to be defined by the user.

The procedures for defining time series output can be summarised as follows.

1. Define the three general parameters

   nosetstsr number of sets

   novarstsr total number of output variables

   nostatstsr total number of output stations

2. Define the attributes of all variables used for time series output. Attributes can be automatically generated if the ivarid attribute is defined by the user.

3. For each set

   - Select which files are to be written (0-D, 2-D, 3-D) using the define attribute.
   - Define the file attributes form, filename, info, header_type. Defaults are available.
   - Select the variables for each output file.
   - Define the attributes of the output grid. Defaults are available.

The procedures for time averaged output are the same, except that an averaging period needs to be defined instead of an output time interval.

For harmonic output additional parameters need to be defined. Assume then that N frequencies are selected within one set. The following output files may be defined within that set:

- 0-D: 1 residual, N amplitude and N phase files
- 2-D: 1 residual, N amplitude, N phase and N elliptic (tidal ellipse parameters) files
- 3-D: 1 residual, N amplitude, N phase and N elliptic files

giving a maximum of $3+8N$ files per set.

In addition to the the procedures for time series output, harmonic output requires to:

1. Define the total number of frequencies nofreqsanal.

2. Define all harmonic frequencies.

3. For each set

   - select which of the $3+8N$ files are written

- select output frequencies
- select elliptic parameters (optional)
- select the period for analysis.

9.6.2 Structure of user output files

The general structure is similar to the one used for forcing files, except that the file additionally contains the coordinates of the output grid.

```
[Names and values of dimensions]
Global attributes
...
Attributes of spatial coordinate 1
...
Attributes of spatial coordinate 2
...
Attributes of time coordinate
...
Attributes of variable 1
...
Attributes of variable 2
...
Attributes of variable n
...
Values of spatial coordinate 1
...
Values of spatial coordinate 2
...
First output time
[Surface elevation coordinate]
Values of variables 1
...
Values of variables n
Second output time
...
```

Example 9.13: General layout of a user output file.

In principle, the number of coordinate variables should be equal to or lower than four (three spatial and one time variable). However, the model uses a σ-grid which is fixed in time in the transformed coordinate system,

but not in physical space due to the up and down movements of the water column. In most graphical applications these changes are negligible and the total water depth is approximated by its mean value. However, time-varying grids can be taken into account in the current version of COHERENS. The output grid can then be defined through the following six coordinates (where the spatial dimension is given in parentheses):

xout X-coordinate in m or fractional degrees longitude (2-D)

yout Y-coordinate in m or fractional degrees latitude (2-D)

zout Z-coordinate in m using mean water depths, i.e. between $-h$ and 0 (3-D)

depout mean water depth in m (2-D)

zetout surface elevation in m (2-D)

time output time (0-D). Unit is specified by the time_format attribute.

It is clear that

- In the case of a 0-D output only the time coordinate is included.
- In the case of a 2-D output only xout, yout, depout and time are included.
- Elevation data are only included if the user selects a time varying 3-D output grid using the time_grid attribute.

The first four arrays have no associated time dimension. The last two are stored within the file in chronological order at regular time intervals. In the ASCII and unformatted binary format, storage is sequential. In the netCDF format, data are stored with increasing record numbers at subsequent times.

9.6.3 Format of files with user-defined output

9.6.3.1 ASCII files

The ASCII format is illustrated with examples taken from the plume test case. The first shows the contents of the output file *plumeA_2.out3A*. The line numbers have been added in the header section for illustrative purposes only and do not appear in the actual file.

```
1 :            2
2 :V2.0
3 :2010/06/23;12:19:42
```

```
4 :Time series data: plume1A
5 :           3
6 :F
7 : T F F
8 :        0        1        1
9 :        1       40       20        0        0       25
10:   10800.00
11:2003/01/03;00:00:00,000
12:2003/01/06;00:00:00,000
13:2003/01/03;00:00:00,000
14:           0
15:           5
16:           5
17:           4
18:      957        5        2        1       40
19:xout
20:X-coordinate
21: m
22:      962        5        2        1       40
23:yout
24:Y-coordinate
25: m
26:      968        5        3        1       40       20
27:zout
28:Z-coordinate
29: m
30:      951        5        2        1       40
31:depout
32:Mean water depth
33: m
34:      952        5        2       23       25
35:time
36:Time
37: date/time
38:      162        5        4        1       40       20       25
39:uvel
40:X-component of current
41:m/s
42:Current
43:      171        5        4        1       40       20       25
44:vvel
```

```
45:Y-component of current
46:m/s
47:Current
48:      175        5        4        1       40       20       25
49:wphys
50:Physical vertical velocity
51:m/s
52:Physical current
53:      204        5        4        1       40       20       25
54:sal
55:Salinity
56:PSU
57:
[xout]
    29500.00        29500.00    ...
[yout]
    500.0000        1500.000    ...
[zout]
   -19.50000       -19.50000    ...
[depout]
    20.00000        20.00000    ...
2003/01/03;00:00:00,000
[uvel]
  -0.2228398      -0.2281678      ...
[vvel]
  -0.8393249E-02  -0.1428325E-01  ...
[wphys]
    0.000000        0.000000      ...
[sal]
    33.00000        33.00000      ...
2003/01/03;03:00:00,000
...
```

Example 9.14: Contents of the output data file *plumeA_2.tsout3A* from the plume test case.

- Line 1: the header_type attribute. If set to 0, no further header information will be given (except this line). Otherwise, its value is 2.

- Line 2: the coherens_version attribute with the currently used COHERENS version.

- Line 3: the creation_date giving the exact date and time when the file was created.
- Line 4: the filedesc attribute with a description of the simulation.
- Line 5: the nodim attribute giving the spatial dimension of the output grid.
- Line 6: the grid_file attribute. If .TRUE. (.FALSE.), grid data and metadata are (are not) written to a separate data file.
- Line 7: the gridded, land_mask and time_grid attributes.
- Line 8: values of the switches of iopt_grid_sph, iopt_grid_htype and iopt_grid_vtype which define the type of grid (see Section 14.4.1).
- Line 9: the attributes ncout, nrout, nzout, which define the size of the output grid (gridded case), nowetout, giving the number of sea points in the output grid (if the land_mask attribute is .TRUE., zero otherwise), nostats (the number of stations in the non-gridded case) and nstepout (number of time records).
- Line 10: the output time step in seconds (deltout attribute).
- Line 11: date/time of first output (startdate attribute).
- Line 12: date/time of last output (enddate attribute).
- Line 13: reference date/time (refdate attribute). If the time coordinate has a numerical format, the time is given as the time elapsed from this date.
- Line 14: the time_format attribute.
- Line 15: the nocoords attribute giving the number of grid coordinates.
- Line 16: the timeid attribute giving the variable id of the time coordinate. This attribute is only used for reading the time coordinate in a netCDF file.
- In case of a time varying grid (only), the zetaid attribute giving the variable id of the zetout coordinate, used for reading the surface elevation coordinate in a netCDF file.
- Line 17: the novars attribute giving the number of data variables.

- In case of station data, the labels and names of the stations are written on subsequent lines with one line per station.
- Lines: 18-37: attributes of each coordinate variable using four lines per variable.
  - line 1: the attributes ivarid, data_type, nrank, shape, represented by 3+nrank integer parameters
  - line 2: the f90_name attribute
  - line 3: the long_name attribute
  - line 4: the units attribute
- Lines 38–57: attributes of each data variable using five lines per variable
  - line 1: the attributes ivarid, data_type, nrank, shape, represented by 3+nrank integer parameters
  - line 2: the f90_name attribute
  - line 3: the long_name attribute
  - line 4: the units attribute
  - line 5: the vector_name attribute
- The total number of header lines in the current example is then given by 17+4*nocoords+5*novars.
  - If nostats>0, nostats lines have to be added.
  - If time_grid is .TRUE., there is 1 additional line for the zetaid attribute. In that case nocoords is also increased by 1.
  - If grid_file is .TRUE., lines 7–16 and 18–37 are moved to the header of the grid file. The total number of header lines is then decreased by 10+4*nocoords.
- The remaining lines in the example form the data section, which is composed of two parts:
  - The first part lists (if the time_grid attribute is .TRUE.) the values of the time-independent coordinate arrays, preceded by an empty line and written in the same order as defined in the header.
  - The second lists (if the time_grid attribute is .TRUE.) the values of the time-dependent coordinate arrays (and data variables in the following order:

* time coordinate
* zetout variable (if time_grid is .TRUE. and preceded by an empty line)
* values of each data variable in the same order as defined in the header. Each variable list is preceded by an empty line.
* For clarity, the name of the variable is substituted in [ ] within the example.

– The same procedure is followed for the next time records.

The contents of an ASCII forcing file are to be read sequentially, i.e. line by line.

- The parameters listed on each line all have the same data type (except for the station parameters) which makes it easy to read them using the character ('A') format (strings) or in free (*) format (integer, real, logical data).
- The data values (except the date/time string if the time coordinate is given in a non-numeric format) are all of type REAL, read in FORTRAN array order. Format specifications are the same as for forcing files (see Section 9.5.3.1), i.e. '(A)' for the date/time string and RealFormat for the other variables.

If grid_file is set to .TRUE., all coordinate information and data are moved to a separate grid file (i.e. lines 7–16, 18–37 and the values of the coordinate arrays).

The next example is a 0-D output file from the test case ***fredyA***.

```
           2
V2.0
2010/06/24;14:07:32
Time series data: fredyA
           0
F
 T F F
        0        1        1
        0        0        0        0        0      865
   600.0000
2003/01/01;00:00:00,000
2003/01/07;00:00:00,000
2003/01/01;00:00:00,000
           0
```

```
           1
           1
           8
      952        5        2       23      865
time
Time
 date/time
        0        5        1       60
ekin
Kinetic energy
GJ

        0        5        1       60
epot
Available potential energy
GJ

        0        5        1       60
theta
Energy ratio


        0        5        1       60
enstr
Enstrophy
m^3/s^2

        0        5        1       60
a1pt
A1%
10^8 m^2

        0        5        1       60
salmin
Minimum salinity
PSU

        0        5        1       60
salmax
Maximum salinity
PSU
```

```
        0        5        1       60
smean
Mean salinity deviation
PSU

2003/01/01;00:00:00,000

    0.000000         9.042427         0.000000         0.000000
0.2500000         33.75000         34.85000      -0.3296069E-02
2003/01/01;00:10:00,000

   0.2259467E-01     8.989937        0.2513329E-02    0.1228950E-04
0.3700000         33.75000         34.85001      -0.3295073E-02
...
2003/01/07;00:00:00,000

   0.6530415         7.097948        0.9200427E-01    0.2413782
3.370000         34.27942         34.85036      -0.3307598E-02
```

Example 9.15: Contents of the output data file *fredyA_2.tsout0A* from the ***fredyA*** test case.

- Since the data have no spatial dimension in a 0-D file, the time variable is the only coordinate.
- Contrary to the general case, 0-D data are not stored indivually but as a vector of size novars. They are read using

  ```
  REAL, DIMENSION(novars) :: out0ddat

  READ (iunit,'(A)') ciodatetime
  READ (iunit,RealFormat) out0ddat
  ekin = out0ddat(1); ...; smean = out0ddat(novars)
  ```

- The ivarid attribute is zero for all data variables. This means that the f90_name, long_name and units attributes are non-standard and defined by the user.

9.6.3.2 unformatted binary files

When the unformatted binary format is selected for a forcing file, the same metadata and data are written as in the case of an ASCII format. The differences are the same as described in Section 9.5.3.2 for forcing files.

9.6.3.3 netcdf files

The description of the netCDF format is similar to the one given in Section 9.5.3.3 and will not repated here. Two examples are given below. The first one lists the contents of the file *plumeA_1.ellip3N*, with harmonic output from the test case ***plumeA***.

```
netcdf plumeA_1
dimensions:
xdim = 120 ;
ydim = 40 ;
zdim = 20 ;
tlendim = 23 ;
tdim = UNLIMITED ; // (6 currently)
variables:
float xout(ydim, xdim) ;
xout:ivarid = 957 ;
xout:long_name = "X-coordinate" ;
xout:units = "m" ;
float yout(ydim, xdim) ;
yout:ivarid = 962 ;
yout:long_name = "Y-coordinate" ;
yout:units = "m" ;
float zout(zdim, ydim, xdim) ;
zout:ivarid = 968 ;
zout:long_name = "Z-coordinate" ;
zout:units = "m" ;
float depout(ydim, xdim) ;
depout:ivarid = 951 ;
depout:long_name = "Mean_water_depth" ;
depout:units = "m" ;
char time(tdim, tlendim) ;
time:ivarid = 952 ;
time:long_name = "Time" ;
time:units = "date/time" ;
float ellmin3d(tdim, zdim, ydim, xdim) ;
ellmin3d:ivarid = 0 ;
ellmin3d:long_name = "M2-Ellipticity" ;
ellmin3d:units = "-" ;
ellmin3d:vector_name = "_" ;

// global attributes:
```

```
:header_type = 2 ;
:coherens_version = "V2.0" ;
:creation_date = "2010/06/23;12:19:42" ;
:filedesc = "Elliptic parameters" ;
:cdfversion = "3̈.6.2¨ of" ;
:iopt_CDF_fill = 0 ;
:grid_dimension = 3 ;
:grid_file = 0 ;
:gridded = 1 ;
:land_mask = 0 ;
:time_grid = 0 ;
:grid_type = 0, 1, 1 ;
:dimensions = 120, 40, 20, 0, 0, 6 ;
:time_step = 43200.f ;
:startdate = "2003/01/03;06:00:00,000" ;
:enddate = "2003/01/05;18:00:00,000" ;
:refdate = "2003/01/03;06:00:00,000" ;
:time_format = 0 ;
:nocoords = 5 ;
:timeid = 5 ;
:novars = 1 ;
:timerec = 6 ;
data:

 xout =
  500, 1500, ...

 yout =
  500, 500, ...

 zout =
  -19.5, -19.5, ...

 depout =
  20, 20, ...

 time =
  "2003/01/03;06:00:00,000",
  "2003/01/03;18:00:00,000",
  "2003/01/04;06:00:00,000",
  "2003/01/04;18:00:00,000",
```

```
  "2003/01/05;06:00:00,000",
  "2003/01/05;18:00:00,000" ;

 ellmin3d =
  0.03929285, 0.03556633, ...
    -0.1459797, -0.1480648, -0.1499712, -0.1511258, -0.1511276, -0.1443668 ;
```

Example 9.16: Contents of the output data file *plumeA_1.ellip3N* from the ***plumeA*** test case.

The attributes are similar to the ones listed in the ASCII format. Some are defined with a different name (e.g. the dimensions attribute which combines the values of the previous parameters ncout, nrout, nzout, nowetout, nostats, nstepout into one vector). Other attributes are defined implicitly such as f90_name, nrank and shape. The timerec giving the current number of (time) records and cdfversion with the current version of netCDF, do not exist in the ASCII and unformatted binary formats.

The second example is the same as Example 9.17 now given in CDL format.

```
netcdf fredyA_2
dimensions:
tlendim = 23 ;
tdim = UNLIMITED ; // (865 currently)
variables:
char time(tdim, tlendim) ;
time:ivarid = 952 ;
time:long_name = "Time" ;
time:units = "date/time" ;
float ekin(tdim) ;
ekin:ivarid = 0 ;
ekin:long_name = "Kinetic_energy" ;
ekin:units = "GJ" ;
ekin:vector_name = "_" ;
float epot(tdim) ;
epot:ivarid = 0 ;
epot:long_name = "Available_potential_energy" ;
epot:units = "GJ" ;
epot:vector_name = "_" ;
float theta(tdim) ;
theta:ivarid = 0 ;
```

```
theta:long_name = "Energy_ratio" ;
theta:units = "_" ;
theta:vector_name = "_" ;
float enstr(tdim) ;
enstr:ivarid = 0 ;
enstr:long_name = "Enstrophy" ;
enstr:units = "m^3/s^2" ;
enstr:vector_name = "_" ;
float a1pt(tdim) ;
a1pt:ivarid = 0 ;
a1pt:long_name = "A1%" ;
a1pt:units = "10^8_m^2" ;
a1pt:vector_name = "_" ;
float salmin(tdim) ;
salmin:ivarid = 0 ;
salmin:long_name = "Minimum_salinity" ;
salmin:units = "PSU" ;
salmin:vector_name = "_" ;
float salmax(tdim) ;
salmax:ivarid = 0 ;
salmax:long_name = "Maximum_salinity" ;
salmax:units = "PSU" ;
salmax:vector_name = "_" ;
float smean(tdim) ;
smean:ivarid = 0 ;
smean:long_name = "Mean_salinity_deviation" ;
smean:units = "PSU" ;
smean:vector_name = "_" ;

// global attributes:
:header_type = 2 ;
:coherens_version = "V2.0" ;
:creation_date = "2010/06/24;09:55:22" ;
:filedesc = "Time series data: fredyA" ;
:cdfversion = "3̈.6.2¨ of" ;
:iopt_CDF_fill = 0 ;
:grid_dimension = 0 ;
:grid_file = 0 ;
:gridded = 1 ;
:land_mask = 0 ;
:time_grid = 0 ;
```

```
:grid_type = 0, 1, 1 ;
:dimensions = 0, 0, 0, 0, 0, 865 ;
:time_step = 600.f ;
:startdate = "2003/01/01;00:00:00,000" ;
:enddate = "2003/01/07;00:00:00,000" ;
:refdate = "2003/01/01;00:00:00,000" ;
:time_format = 0 ;
:nocoords = 1 ;
:timeid = 1 ;
:novars = 8 ;
:timerec = 865 ;
data:

 time =
  "2003/01/01;00:00:00,000",
  ...
  "2003/01/07;00:00:00,000" ;

 ekin = 0, 0.02259467, 0.08382063, ...

 epot = 9.042427, 8.989937, ...

 theta = 0, 0.002513329, ...

 enstr = 0, 1.22895e-05, ...

 a1pt = 0.25, 0.37, ...

 salmin = 33.75, 33.75, ...

 salmax = 34.85, 34.85001, ...

 smean = -0.003296069, -0.003295073, ...
```

Example 9.17: Contents of *fredyA_2.tsout0N* output data file from the ***fredyA*** test case in CDL format.

Contrary to the ASCII case, a separate netCDF call has to be made to read each of the 0-D output data, i.e. the data are not stored in vector form but as separate records in the netCDF file.