Chapter 12

Structure of the model code

12.1 Source code files

The main code files, located in the **source** directory, can be classified into distinct groups, using their file name.

- Files with suffix *.f90* are FORTRAN source code files, files with *.F90* are FORTRAN files containing C-language code (#ifdef) statements.
- Files whose name start with a lower case character and include no underscore character, contain "declaration modules" with declarations of variables and arrays as described in Section 8.1.5.
- Files whose name start with a lower case character and include an underscore character, contain module routines, as described in Section 8.1.5. The routines are often generic and of a general nature.
- Files whose name start with an upper case character (except *Usrdef_** files) are external subprograms with "actual" code.
- Files whose name start with *Usrdef_*, contain routines for user setup. (The ones in the **source** directory are empty and intended for proper compilation only, user-defined versions are defined elsewhere).

Table 12.1: List of declaration module files.

file name	contents
currents.f90	(2-D and 3-D) current arrays
datatypes.f90	definitions of all derived types, used in the program
density.f90	density arrays
depths.f90	water depth and surface elevations arrays
diffusion.f90	horizontal and vertical diffusion coefficient arrays
fluxes.f90	arrays of surface and bottom fluxes, drag and exchange coefficients, roughness lengths
grid.f90	model grid arrays
gridpars.f90	model grid and related parameters
iopars.f90	parameters and arrays for all kind of I/O (including user-defined)
meteo.f90	surface meteorological arrays
modids.f90	key id definitions of physical model variables
nestgrids.f90	parameters and arrays for sub-grid nesting applications
obconds.f90	arrays for 2-D and 3-D open boundary conditions, arrays for 1-D surface forcing
optics.f90	optical arrays (including irradiance)
paralpars.f90	parameters and arrays needed for parallel applications
physpars.f90	physical model parameters
relaxation.f90	arrays for applying relaxation conditions
sedarrays.f90	sediment model arrays
sedids.f90	key id definitions of sediment model variables
sedpars.f90	sediment model parameters
sedswitches.f90	sediment model switches
structures.f90	parameters and arrays for the structure and discharge units
switches.f90	physical model switches
syspars.f90	"system" parameter constants
tide.f90	parameters and arrays for tidal applications
timepars.f90	date and time parameters
turbpars.f90	turbulence model constants
turbulence.f90	turbulence model arrays

Table 12.2: List of module routine files.

file name	contents
array_interp.f90	routines for interpolation on the model grid
cf90_routines.F90	library of netCDF routine calls
check_model.f90	routines for checking of user-defined parameters and arrays in the physical model
check_sediments.f90	routines for checking of user-defined parameters and arrays in the sediment model
cif_routines.f90	utility routines used for reading and writing a CIF
comms_MPI.F90	library of MPI routine calls
datatypes_init.f90	initialisation of derived type scalar and array variables
default_model.f90	default settings for the physical model
default_sediments.f90	default settings for the sediment model
diagnostic_routines.F90	utility routines calculating terms in the energy equation, total energy, potential energy, enstrophy and vorticity
error_routines.F90	routines performing error checking
fft_library.f90	routines for performing fast Fourier transforms
grid_interp.f90	routines for performing interpolation from and to external grids and locations
grid_routines. f90	utility routines performed on the model grid
inout_paral.f90	routines for preparing input/output in parallel mode
inout_routines.f90	routines for performing input/output in standard format
math_library.f90	library of diverse mathematical routines (e.g. root finder)
model_output.f90	routines for defining standard output data in the physical model
modvars_routines.f90	attributes of variables and files in the physical model
nla_library.F90	linear algebra library
paral_comms.f90	parallel communication library
paral_utilities.f90	utility routines for parallel applications
reset_model.F90	reset setup parameters and arrays in the physical model defined by the user
reset_sediments.f90	reset setup parameters and arrays in the sediment model defined by the user
rng_library.f90	random generator library
sediment_output.f90	routines for defining standard output data in the sediment model
sedvars_routines.f90	attributes of variables and files in the sediment model
time_routines.f90	(calendar) date and time utility routines
turbulence_routines.F90	routines used by the turbulence subprogram
utility_routines.f90	various utility routines

Table 12.3: List of files with external procedures.

file name	contents
Advection_Terms.F90	advective terms in the transport equations
Allocate_Sediment_Arrays.f90	allocate/deallocate variables with a global scope for the sediment model
Allocate_Sediment_Arrays.f90	allocate/deallocate variables with a global scope for the sediment model
Bottom_Fluxes.f90	bottom drag coefficient and shear stress
Coherens_Program.f90	COHERENS main program
Corrector_Terms.F90	corrector terms in the transport equations
Density_Equations.F90	salinity and temperature equations, optical module, equation of state, baroclinic pressure gradient
Diffusion_Coefficients.F90	horizontal and vertical diffusion coefficients
Diffusion_Terms.F90	diffusion terms in the transport equations
Grid_Arrays.F90	model grid parameters and arrays (grid spacings, pointer arrays, open boundary locations, water depths)
Harmonic_Analysis.f90	harmonic analysis
Hydrodynamic_Equations.F90	hydrodynamic equations (currents, 2-D mode, elevations)
Model_Finalisation.f90	finalise physical model
Model_Initialisation.F90	initialise physical model
Model_Parameters.f90	read/write a CIF for the physical model
Nested_Grids.F90	sub-grid nesting
Open_Boundary_Conditions.f90	apply open boundary conditions
Open_Boundary_Data_2D.f90	define 2-D open boundary conditions and update data
Open_Boundary_Data_Prof.f90	define 3-D open boundary conditions and update data
Parallel_Initialisation.f90	initialise parallel mode (parameters, domain decomposition, ...)
Relaxation_Zones.f90	define and apply relaxation conditions
Sediment_Bottom_Fluxes.F90	near bed boundary conditions, (skin) shear stress and roughness length, and critical shear stress in the sediment model
Sediment_Density_Equations.F90	sediment contributions in the calculations within the physical model involving density
Sediment_Equations.F90	COHERENS sediment module (main part)

(*Continued*)

Table 12.3: *Continued*

Sediment_Finalisation.f90	finalise sediment model
Sediment_Initialisation.F90	initialise sediment model
Sediment_Parameters.f90	read/write a CIF for the sediment model
Structures_Model.f90	structure (dry cells, thin dams, weirs, barriers) and discharge model units
Surface_Bounday_Data_1D.f90	define and apply surface forcing conditions (slope and elevation) for the water column mode
Surface_Data.f90	update 2-D external forcing data
Surface_Fluxes.F90	surface fluxes
Surface_Grids.f90	define 2-D external grids
Tidal_Forcing.F90	astronomical argument, nodal factors, tidal force
Time_Averages.f90	time-averaged output
Time_Series.f90	time series output
Transport_Equations.F90	solve transport equations
Turbulence_Equations.F90	turbulence models

Table 12.4: List of user-defined routine files.

file name	contents
Usrdef_Harmonic_Analysis.f90	parameters and data for harmonic analysis and output
Usrdef_Model.f90	"basic" model setup (model parameters, bathymetry, domain decomposition, initial conditions, open boundary conditions)
Usrdef_Nested_Grids.f90	setup of sub-grids for nesting
Usrdef_Output.f90	output completely specified by the user
Usrdef_Sediment.f90	setup of the sediment model
Usrdef_Surface_Data.f90	definition of external 2-D grids and update of 2-D external data
Usrdef_Time_Averages.f90	parameters and data for time-averaged output
Usrdef_Time_Series.f90	parameters and arrays for time series output

12.2 Structure diagrams

12.2.1 General structure

The general structure of the program is given in Figure 12.1. The program

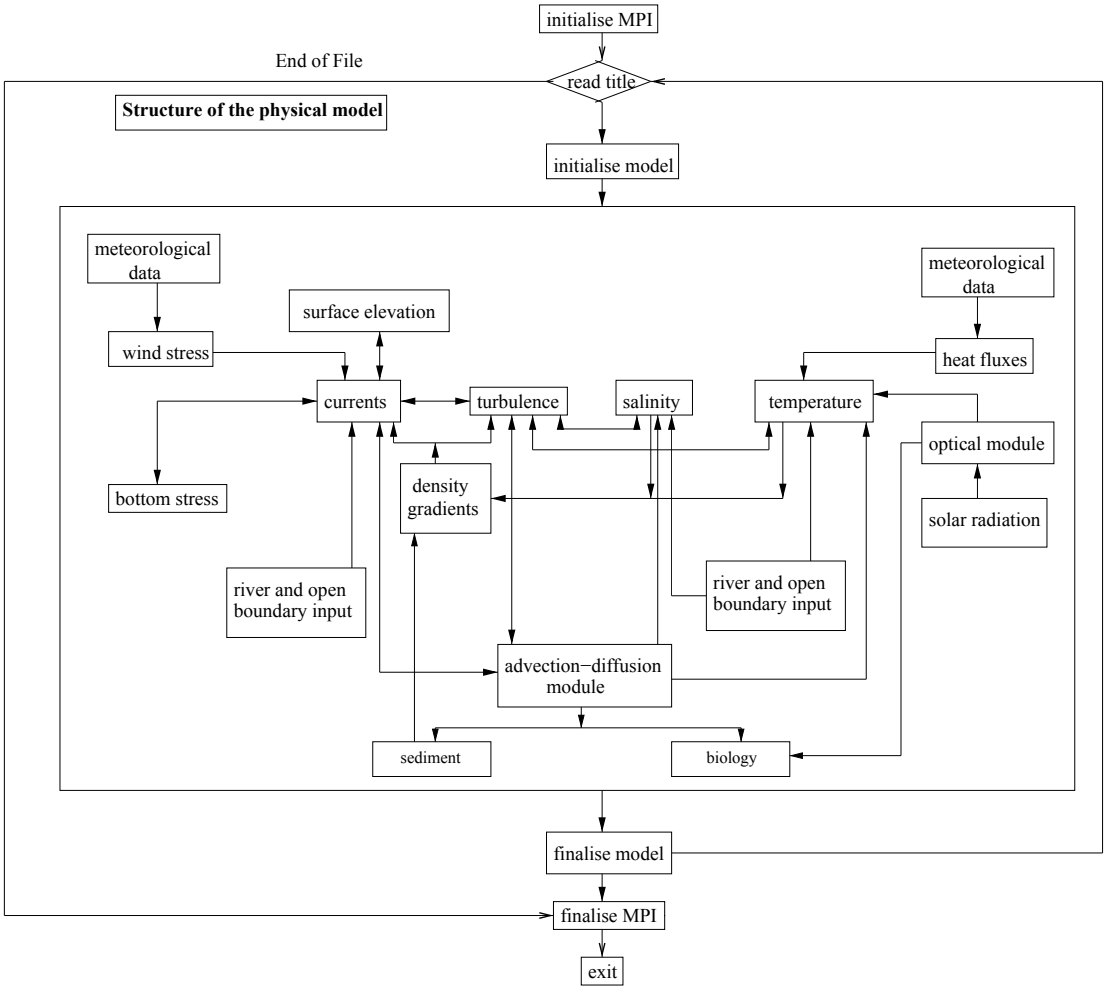

Figure 12.1: General structure of COHERENS.

contains two major loops.

- The first (outer) loop is contained within the large (semi-)rectangle. Each cycle corresponds to a new simulation, initiated by reading a next input line from the file *defruns*. This is further discussed in Section 14.1.
- The second (inner) loop within the smaller rectangle denotes the time-stepping.

Each simulation is composed of three parts: initialisation, time-stepping (where the actual calculations are performed) and finalisation. Details are given below. Initialisation and finalisation of MPI are the only procedures outside the outer loop. This means that, although multiple simulations can be performed within one run, they must all be conducted either in serial or in parallel mode.

The advection-diffusion module forms the central "core" part of the time-loop section. The module is coded in a generic way and solves the advective-diffusive transport equations of any scalar variable (temperature, salinity, sediment concentrations, biological state variables, ...), 3-D and 2-D currents, and turbulence tranport equations. The inputs for the routine are prepared in separate routines for each variable (temperature, salinity, 3-D currents, 2-D currents, sediments, turbulent energy, ...) and are composed of source terms, open boundary and surface forcing data. The boundary data are obtained via general routine calls. For example, there is one program module (discussed in Section 12.2.4 below) dealing with reading and updating of open boundary profile data for any 3-D quantity.

12.2.2 Initialisation procedures

The initialisation procedures are schematically presented in Figure 12.2. Except for user output, all initialisation routines are called from initialise_model.

The first task to be performed by the program is the definition of all parameters and arrays needed to setup the application. This is organised in different sections.

1. Model parameters:
    - model switches
    - date and time parameters (start and end date, time steps)
    - model grid (dimensions, resolution, number of open sea and river boundaries)
    - various parameters like number and type of tidal constituents, number of nested sub-grids, ...
    - parameters for setting up the model in parallel mode
    - physical model parameters
    - numerical model parameters
    - parameters for the turbulence sub-module(s)
    - attributes of external 2-D grids

- attributes of the forcing files
- parameters to define the type and form of monitoring
- parameters and switches for the sediment transport module

For details see Chapter 14.

2. Model grid and bathymetry:

   - coordinates of the model grid
   - bathymetry
   - location of open boundaries

   For details see Sections 15.1.

3. Domain decomposition (parallel mode, see Section 14.9). Once the model grid and domain decomposition have been defined, memory is allocated to all model grid arrays and a series of additional arrays (grid spacings, pointer arrays, ...) are defined.

4. Initial conditions (see Section 15.2 for the physics and 19.2 for the sediments).

5. Definition of the areas for application of the open boundary relaxation scheme (Section 16.3).

6. Locations of nested sub-grid(s) (see Section 17.3).

7. Positions of external 2-D data grids (Section 17.2).

8. Type and form of open boundary conditions for the 2-D mode (Section 16.1.1), baroclinic currents and all 3-D scalars (Section 16.2.1).

9. Initialise surface and bottom fluxes.

10. Parameters for setting up user defined output (see Chapter 20).

Open boundary and surface forcing data are usually given as time series. If the first data time coincides with the initial time, the data file is opened and the first and (eventually second) time records are read from the file during the initialisation phase of the program. This has the further advantage that error checking can be performed (existence of the file, data formats, ...) before the program enters the time loop.

Model parameters and switches for the different mofel compartments (physics, sediments) can be defined either in a usrdef_ routine or by reading from a Central Input File (CIF).

Each section of the initialisation contains (or may contain) the following sub-tasks:

1. Defaults values are given to several model parameters and arrays. In many cases, these defaults are meaningfull and should be maintained. In other case, they are not meaningfull and must be replaced by the user. The advantage of such procedure is a more efficient error checking.

2. Values are (re)-defined. Two methods are available:

   - The definitions are programmed by the user in a usrdef_ routine. Options are foreseen in the program to write these definitions to an external file in COHERENS standard format.
   - All values are obtained as input from an external file in COHERENS standard format.

3. Depending on the definitions given by the user, parameters are reset from their default values.

4. The setup of the model (definitions of model parameters and arrays) are checked for errors. If errors are detected, appropriate messages are written to an *errlog* file and the program aborts.

12.2.3 Time loop

Figure 12.3 shows a diagram of the time loop. The order of routine calls is in line with the solution procedure described in Section 5.8.

The routines where the 2-D mode, 3-D current, temperature, salinity and sediment transport equations are solved, are schematically presented in Figures 12.4–12.8. Each routines is composed of an initialisation section, a main part where the variable(s) is (are) updated and a finalisation section.

initialisation Actions performed during the initialisation phase (time t=0): allocation of local arrays, definition of open boundary conditions, opening of data files and reading of first time records.

- The initialisation of the 2-D mode is actually perfomed in routines update_2dobc_data called from current_2d and define_2dobc_spec called from update_2dobc_data.
- Open boundary conditions for 3-D currents are defined in current_cor.

- If the temperature equation is forced with SST, the SST grid and data are defined and initialised in temperature_equation.

main section

- update open boundary data
- apply open boundary conditions
- apply surface and bottom boundary conditions (3-D variables only)
- calculate source terms
- solve the transport equations by calling the appropriate transport routine
- exchange array sections with neighbouring sub-domains (parallel mode only)
- write interpolated data for nested sub-grids (if requested)

finalisation Deallocate arrays at the current or final time step.

The following remarks are to be given:

- Surface elevations are updated in current_2d before the depth-integrated transports.
- Open boundary conditions are applied in current_2d after solving the 2-D depth-integrated momentum equations.
- The 3-D current calculations are split over two routines called at different (predictor and corrector) time steps: surface and bottom boundary conditions are applied, source terms calculated and transport equations solved in current_pred; open boundary conditions and corrector step are applied, vertical currents calculated and nested output written in current_corr.
- Besides routines for solving the transport equations for all sediment fractions, the sediment model provides separate routines for update of the bed or total load transport.
- Meteorological forcing data are defined by a separate call to meteo_input from the main program.

The update of a 2-D or 3-D quantity by an advection-diffusion type equation is performed in one of the transport_at_* routines, which integrates the model equations in time. Exceptions are surface elevation and vertical current which are obtained from the 2-D and 3-D continuity equations. The procedures closely follow the numerical descriptions given in Chapter 5 so that no diagrams need to be given here.

12.2.4 Open boundary and surface forcing data input

The procedure for applying open boundary conditions for the 2-D mode is summarised in Figure 12.9:

1. The routine update_2dobc_data is called from current_2d:
   - At the initial time the routine define_2d_obc_spec is called where
     - A series of arrays to be specified by the user, are allocated.
     - The arrays are defined by calling either the user-defined routine usrdef_2dobc_spec or as input from a standard COHERENS file by calling read_2dobc_spec. If requested, the arrays are written to a standard file by calling write_2dobc_spec.
     - Error checking is performed.
   - If there are external data files, it is checked first for each data file, whether the data are still up to date, which means that the last date and time for which data are available is later than the current one. If this is not the case (such as at the initial time), define_2d_obc_data is called where:
     - New data are obtained by calling either the user-defined routine usrdef_2dobc_data or as input from a standard COHERENS file by calling read_2dobc_data. If requested, the arrays are written to a standard file by calling write_2dobc_data.
     - If an end of file condition occurs, further action is determined by the endfile attribute. This is further discussed in Section 14.7.2.
   - The new data (if any), representing the ψ_0^e term in equation (4.354), are stored in the appropriate open boundary arrays and interpolated in time (if requested).
   - Harmonic tidal expansions are evaluated. If needed (which is usually the case at the initial time), astronomical arguments and nodal factors are calculated by calling astro_params. The harmonic terms are added to the data values.
2. The open boundary conditions are applied by calling open_boundary_conds_2d.

User-defined setup for 2-D open boundary conditions is further discussed in Section 16.1.

The procedure for applying open boundary conditions for the 3-D mode is given in Figure 12.10. The code is written in a generic form so that the routines can be used for any 3-D quantity (currents, temperature, ...).

1. At the initial time the routine define_profobc_spec is called from the "main" routine (current_corr, temperature_equation, ...):
   - A series of arrays to be specified by the user, are allocated.
   - The arrays are defined by calling either the user-defined routine usrdef_profobc_spec or as input from a standard COHERENS file by calling read_profobc_spec. If requested, the arrays are written to a standard file by calling write_profobc_spec.
   - Error checking is performed.
2. The routine update_profobc_data is called where:
   - It is checked first for each data file, whether the data are still up to date, which means that the last date and time for which data are available is later than the current one. If this is not the case (such as at the initial time), define_profobc_data is called where:
     - New data are obtained by calling either the user-defined routine usrdef_profobc_data or as input from a standard COHERENS file by calling read_profobc_data. If requested, the arrays are written to a standard file by calling write_profobc_data.
     - If an end of file condition occurs, further action is determined by the endfile attribute. This is further discussed in Section 14.7.2.
   - The new data (if any) are stored in the appropriate open boundary profile arrays and interpolated in time (if requested).
   - If any of the interpolating values has a flagged value, the interpolated open boundary profile data value will be flagged as well. A flagged value at a certain vertical level within a vertical profile means that a zero gradient condition will be applied at that particular level.
3. The open boundary conditions are applied by calling open_boundary_conds_3d for baroclinic currents or open_boundary_conds_prof for scalars.

User-defined setup for baroclinic open boundary conditions is further discussed in Section 16.2.

The application of 2-D external data requires firstly the definition of the data grid, which is implemented as follows (no diagram shown):

1. A derived type "grid" array is created by allocation in the "main" routine (meteo_input for the meteo, temperature_equation for the SST

grid or wave_input for surface waves), for storing the relative coordinates of the data grid with respect to the model grid (see Section 8.1.4).

2. The grid is defined by calling define_surface_input_grid. Definition depends on the value nhtype grid attribute (see Section 10.4.2):

   0: No grid needs to be defined

   1: The grid is uniform rectangular and is defined by calling construct_regular_grid.

   2: The grid is non-uniform rectangular. Coordinate arrays are obtained by calling either the user-defined routine usrdef_surface_absgrd or as input from a standard COHERENS file by calling read_surface_absgrd. If requested, the coordinates are written to a standard file by calling write_surface_absgrd. The relative coordinates are obtained by calling model_to_data_coords.

   3: The grid is non-uniform and non-rectangular. The relative coordinate arrays are directly obtained by calling either the user-defined routine usrdef_surface_relgrd or as input from a standard COHERENS file by calling read_surface_relgrd. If requested, the coordinates are written to a standard file by calling write_surface_relgrd.

   4: The grid coincides with the model grid and does not need to be defined here.

Setup of 2-D data grids is discussed in Sections 17.2.1–17.2.2.

The procedure for the input of forcing data from a 2-D external data grid is shown in Figure 12.11. The code is written in a generic form so that the routines can be used for meteorological, SST, wave, ... data. The routine update_surface_data is called from the "main" routine (meteo_input for the meteo or temperature_equation for the SST grid):

1. It is checked first for each data file, whether the data are still up to date, which means that the last date and time for which data are available is later than the current one. If this is not the case (such as at the initial time), define_surface_data is called where:

   - New data are obtained by calling either the user-defined routine usrdef_surface_data or as input from a standard COHERENS file by calling read_surface_data. If requested, the arrays are written to a standard file by calling write_surface_data.
   - If an end of file condition occurs, further action is determined by the endfile attribute. This is further discussed in Section 14.7.2.

2. The data are interpolated in time.

3. If any of the interpolating values has a flagged value, the interpolated open boundary profile data value will be flagged as well. In case of SST data, the flagged value is replaced by the modelled temperature at the highest level. There is currently no procedure foreseen for flagged meteorological data values.

4. The data are interpolated in space by calling intpol_data_to_model_2d if 0<nhtype<4.

12.2.5 Finalisation procedures

After termination of the time loop the simulation is finalised as follows:

- All files, which are still open with exception of monitoring files, are closed.
- A timer report is written on request.
- Model arrays, which are still allocated, are deallocated.
- All monitoring files are closed.
- A new input line is read from the *defruns* file. If there is an end of file condition, the program checks whether MPI was switched on at the start, finalises MPI if needed and terminates. Otherwise, a new simulation starts and all model parameters and arrays are reset to their default values as part of the re-initialisation procedures discussed in Section 12.2.2.

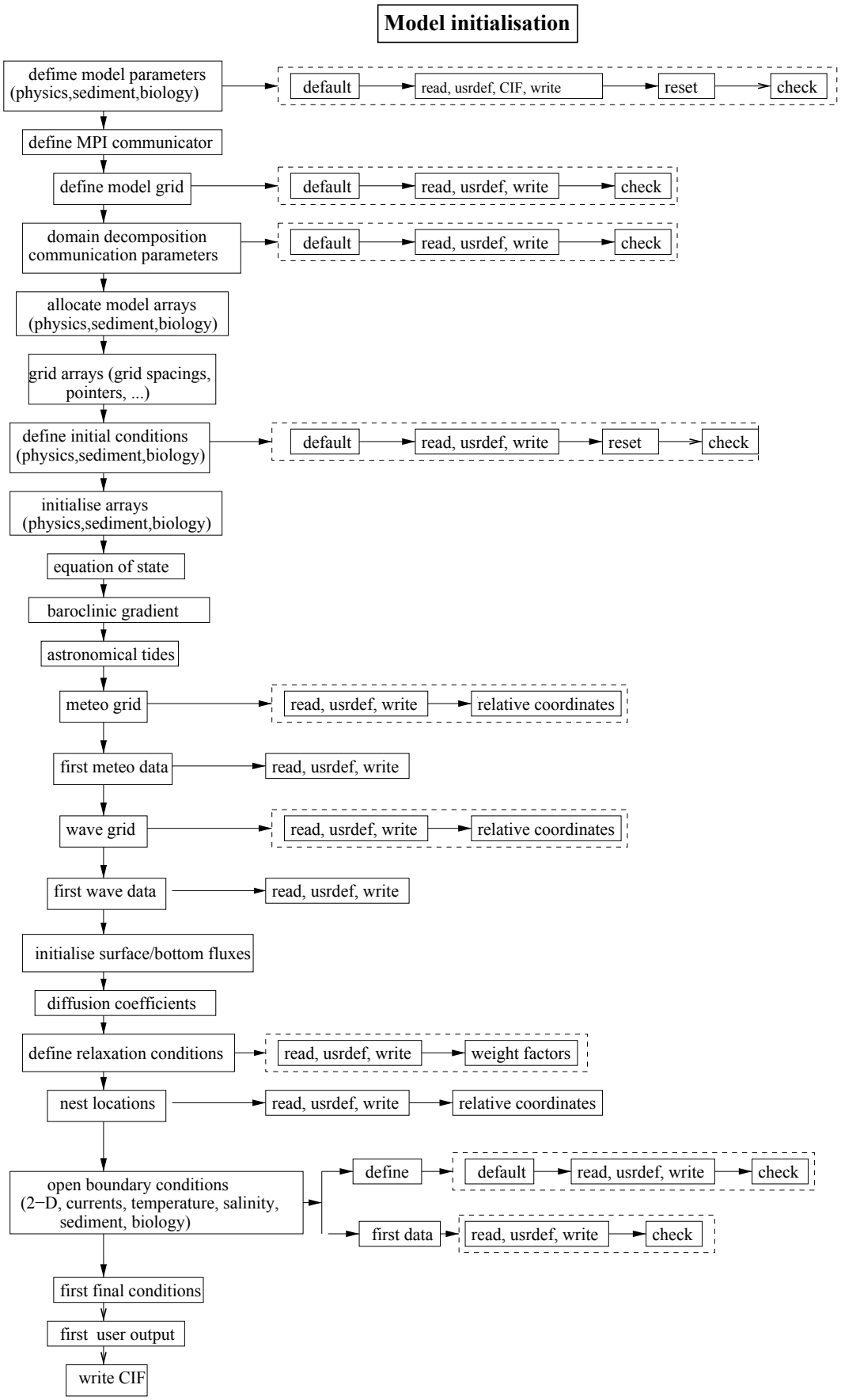

Figure 12.2: Schematic diagram of all initialisation procedures.

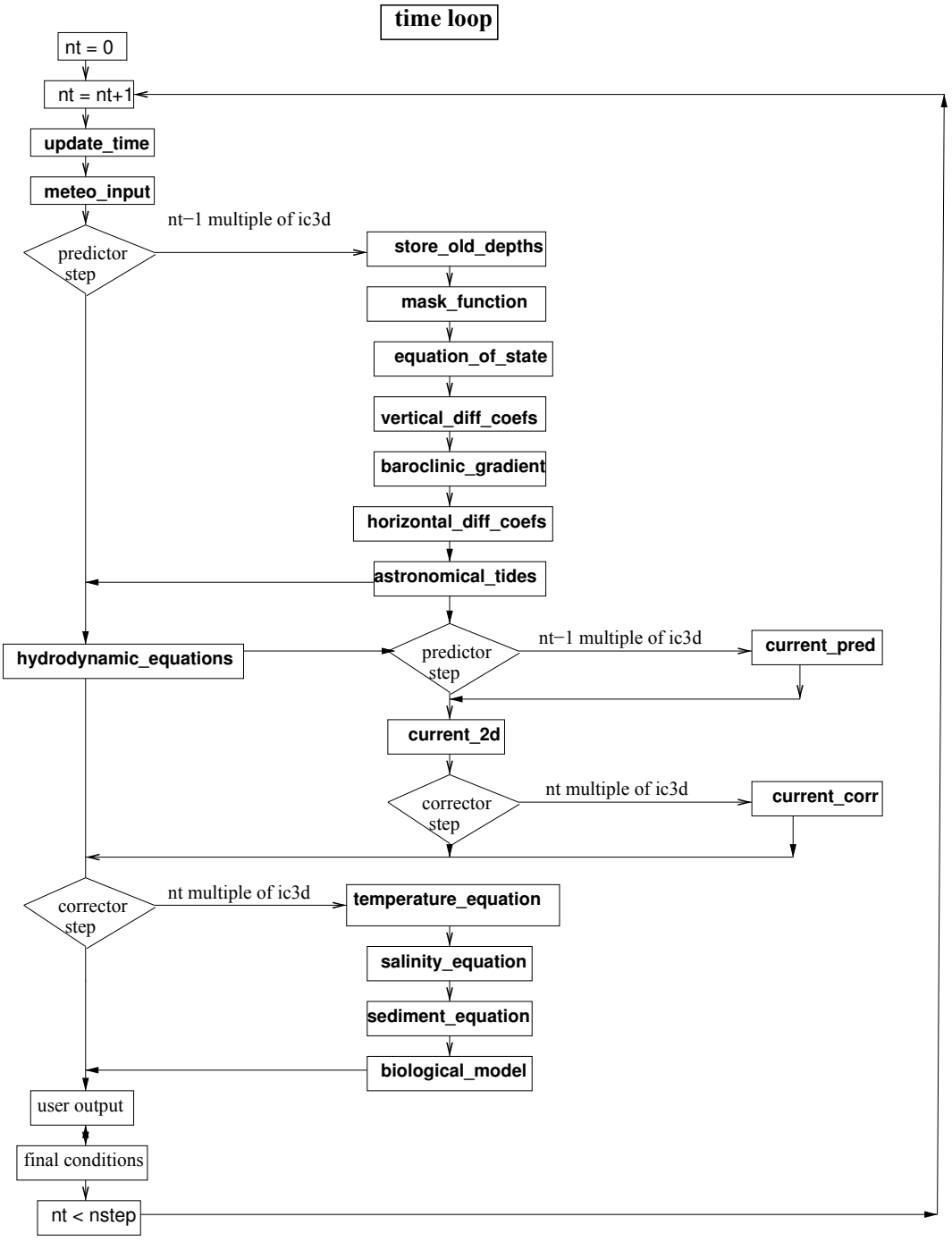

Figure 12.3: Structure diagram of the time loop.

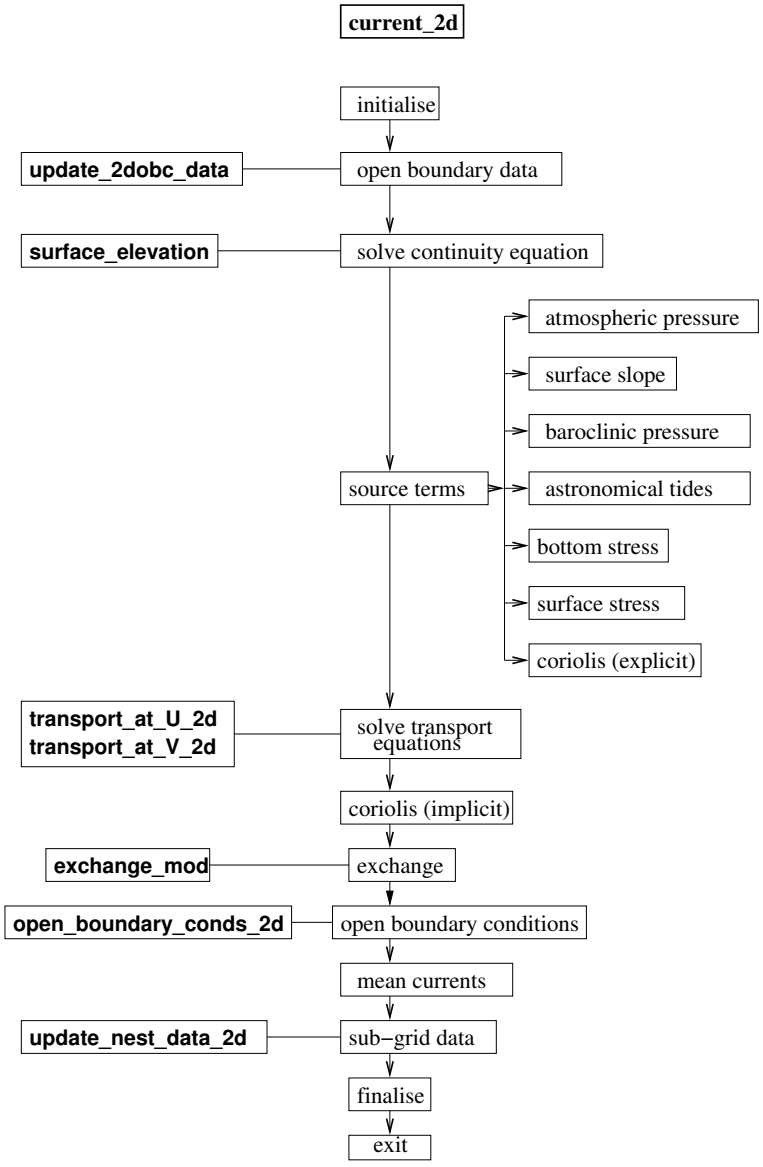

Figure 12.4: Diagram of routine current_2d which solves the 2-D mode equations.

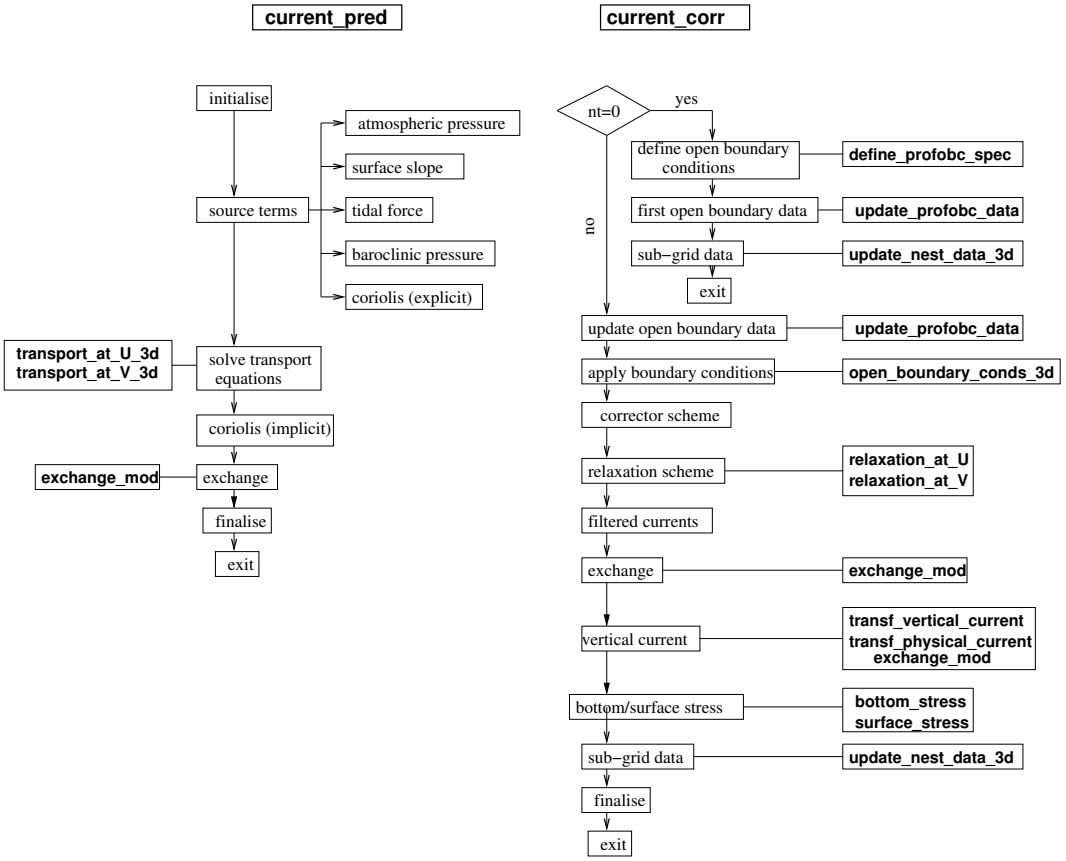

Figure 12.5: Diagrams of the routines current_pred and current_corr which solve the 3-D momentum equation at the predictor and corrector step.

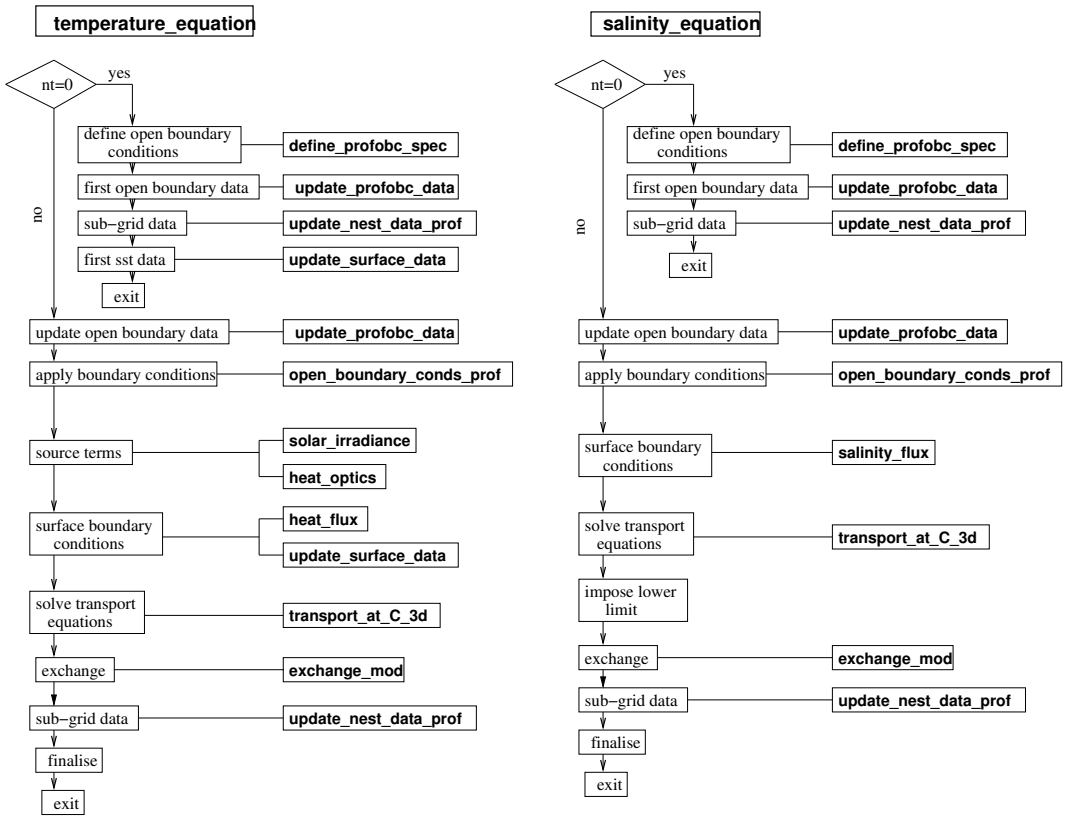

Figure 12.6: Diagrams of the routines temperature_equation and salinity_equation which solve the temperature and salinity equations.

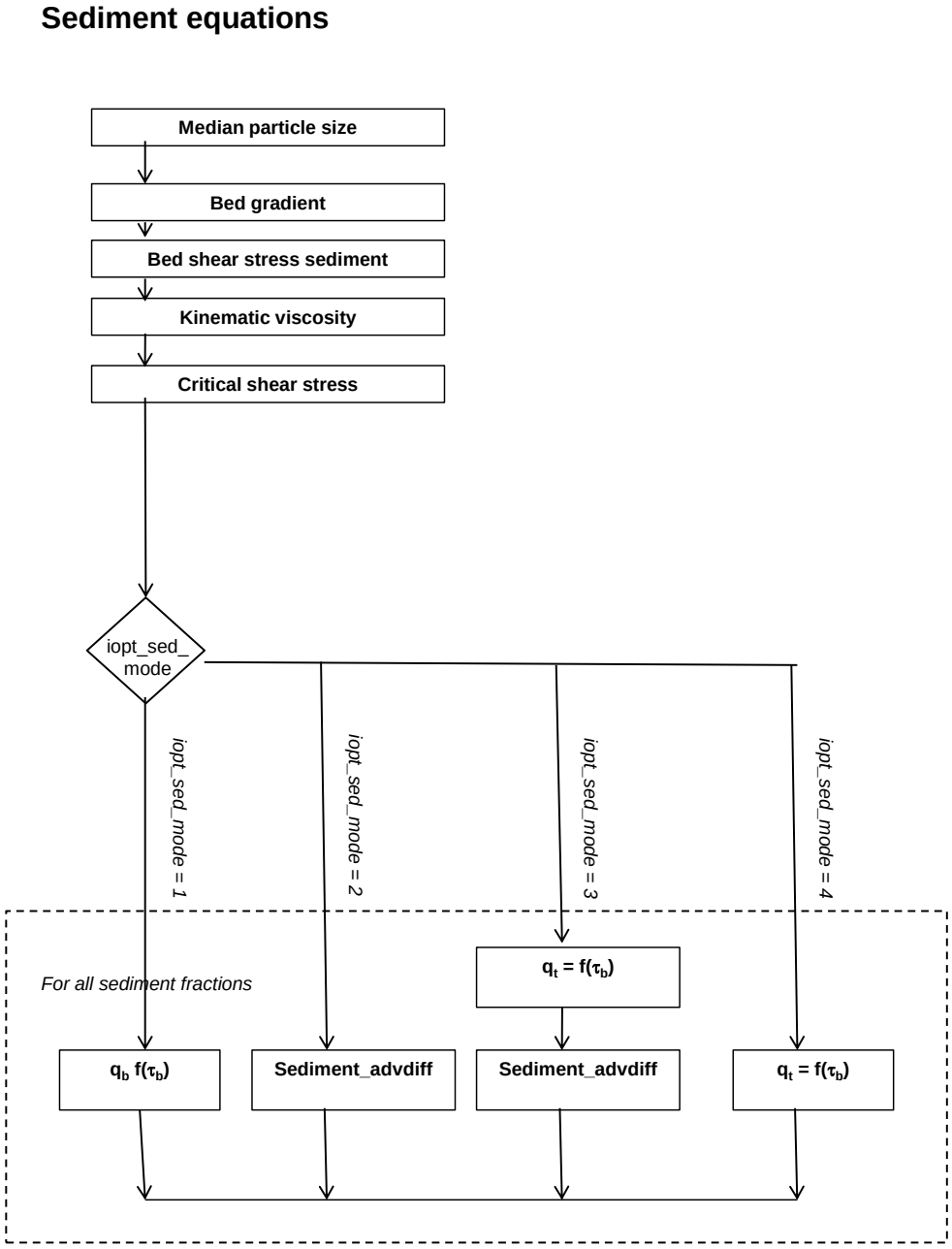

Figure 12.7: Diagram of the routine sediment_equation which is the "main" routine of the sediment transport model.

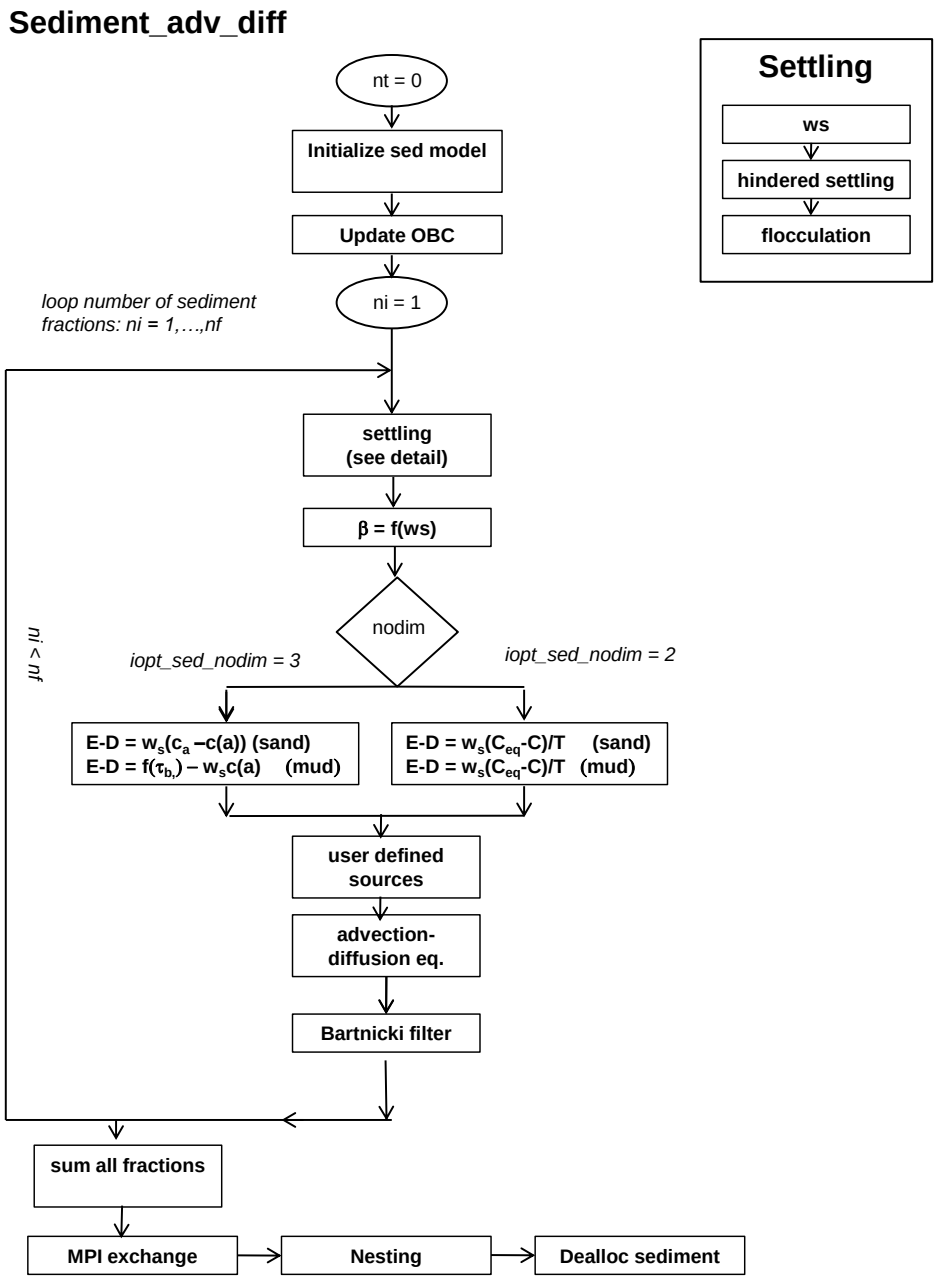

Figure 12.8: Diagram of the routine sediment_advdiff_equation which solves the transport equations for sediments.

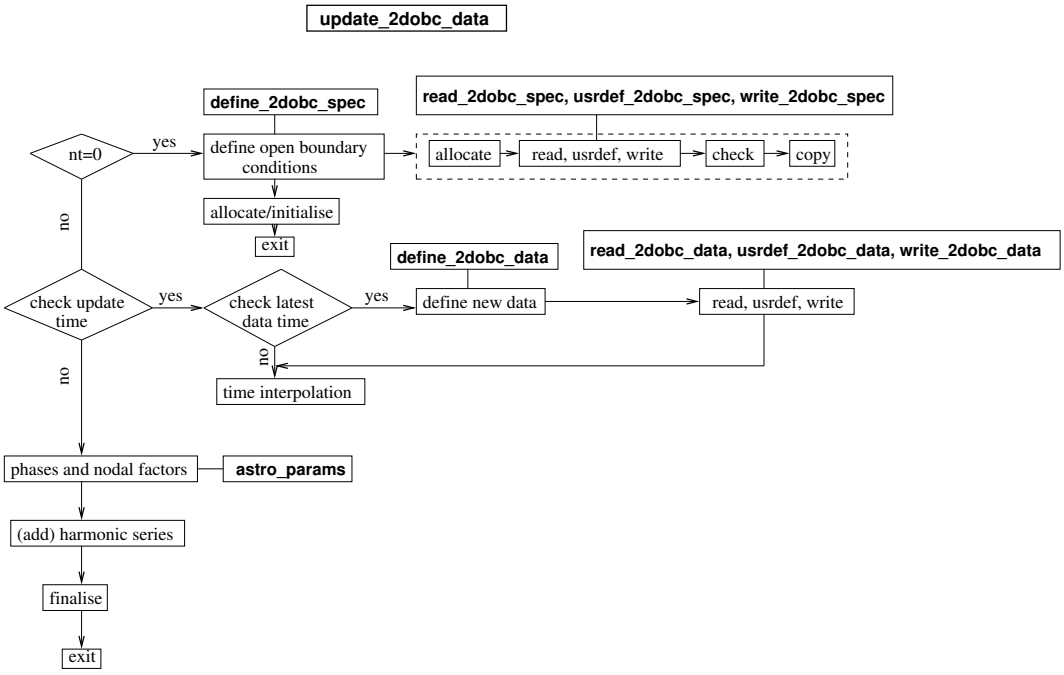

Figure 12.9: Diagrams of the routines used for defining and updating 2-D open boundary conditions and data.

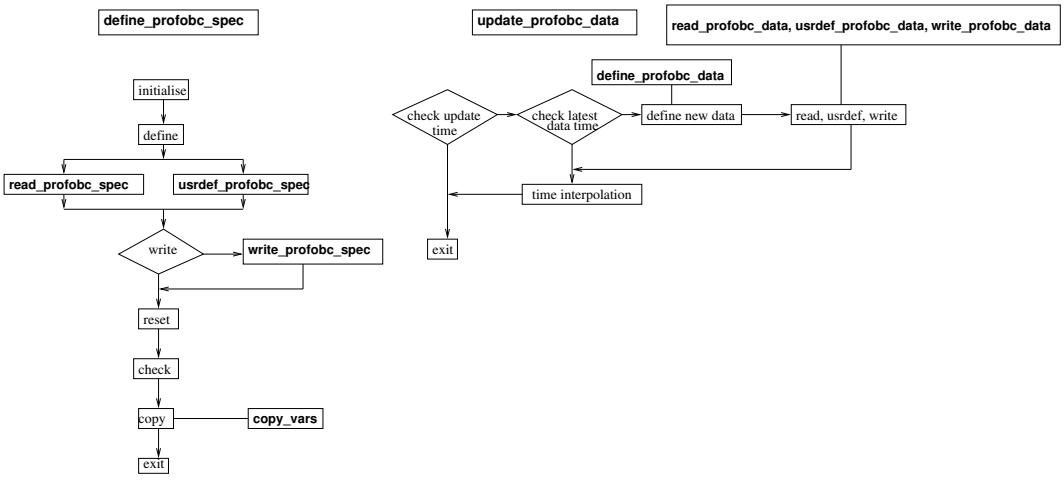

Figure 12.10: Diagrams of the routines used for defining and updating 3-D open boundary conditions and data.

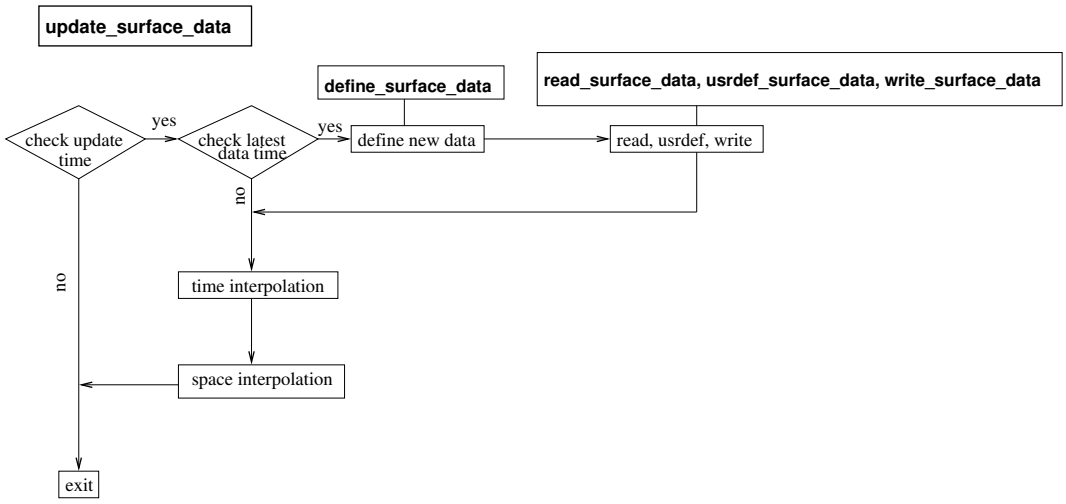

Figure 12.11: Diagram of the routines used for defining and updating data from an external 2-D grid.