Chapter 34

Sediment reference manual

34.1 External routines

34.1.1 *Allocate_Sediment_Arrays.f90*

allocate_sed_arrays

`SUBROUTINE allocate_sed_arrays`

File
Allocate_Sediment_Arrays.f90

Type
Subroutine

Purpose
Allocate sediment arrays

Arguments
None

Calling procedures
initialise_model

deallocate_sed_arrays

`SUBROUTINE deallocate_sed_arrays`

File
Allocate_Sediment_Arrays.f90

Type
Subroutine

Purpose
Deallocate sediment arrays

Arguments
None

Calling procedures
simulation_end

34.1.2 *Sediment_Bottom_Fluxes.F90*

bottom_flux_sed

```
SUBROUTINE bottom_flux_sed
```

File
Sediment_Bottom_Fluxes.f90

Type
Subroutine

Purpose
Bottom sediment flux

Reference
Sections 7.6.3 and 7.7.1

Arguments
None

Called external procedures
bottom_stress_waves

Calling procedures
sediment_advdiff, sediment_suspendedload, sediment_totalload

bottom_stress_sed

```
SUBROUTINE bottom_stress_sed
```

File
Sediment_Bottom_Fluxes.f90

Type
Subroutine

Purpose
Bottom (skin) stress as used in the sediment transport module

Arguments
None

Calling procedures
sediment_equation

bottom_stress_waves

```
SUBROUTINE bottom_stress_waves
CHARACTER (LEN=lennode), INTENT(IN):: cnode
```

File
Sediment_Bottom_Fluxes.f90

Type
Subroutine

Purpose
Wave-induced bottom stresses

Arguments

cnode	grid node where the total stress is evaluated

Calling procedures
bottom_flux_sed, sediment_bedload, sediment_totalload

critical_shear_stress

```
SUBROUTINE critial_shear_stress
```

File
Sediment_Bottom_Fluxes.f90

Type
Subroutine

Purpose
Critical shear stress

Reference
Section 7.6.4.2

Arguments
None

Calling procedures
sediment_equation

34.1.3 *Sediment_Density_Equations.F90*

baroclinic_gradient_sed_cubic

```
SUBROUTINE baroclinic_gradient_sed_cubic(zcoord,dzx,dzy,dzz,cdir)
CHARACTER (LEN=1) :: cdir
REAL, INTENT(IN), DIMENSION(0:ncloc+1,0:nrloc+1,nz) ::  dzx, dzy, dzz
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz) :: zcoord
```

File
Sediment_Density_Equations.F90

Type
Subroutine

Purpose
Sediment component of the baroclinic pressure gradient (cube-H method)

Reference
Sections 5.3.13.3 and equation (7.22)

Arguments

zcoord	Z-coordinate at W-nodes	[m]
dzx	Harmonic derivative of the X-coordinate	
dzy	Harmonic derivative of the Y-coordinate	
dzz	Harmonic derivative of the Z-coordinate	
cdir	Direction of the gradient ('X', 'Y')	

Called external procedures
hcube_deriv, hcube_fluxes

Calling procedures
baroclinic_gradient

baroclinic_gradient_sed_sigma

```
SUBROUTINE baroclinic_gradient_sed_sigma(zcoord,cdir)
CHARACTER (LEN=1), INTENT(IN) :: cdir
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz) :: zcoord
```

File
Sediment_Density_Equations.F90

Type
Subroutine

Purpose
Sediment component of the baroclinic pressure gradient (second order method)

Reference
Sections 5.3.13.1 and equation (7.22)

Arguments

zcoord	Z-coordinate array at the W-nodes	[m]
cdir	Direction of the gradient ('X', 'Y')	

Calling procedures
baroclinic_gradient

baroclinic_gradient_sed_z

```
SUBROUTINE barcolinic_gradient_sed_z(sigint,kint,cdir)
CHARACTER (LEN=1) :: cdir
INTEGER, INTENT(IN), DIMENSION(ncloc,nrloc,2:nz+1,2) :: kint
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,2:nz+1,2) :: sigint
```

File
Sediment_Density_Equations.F90

Type
Subroutine

Purpose
Sediment component of the baroclinic density gradient (Z-level method)

Reference
Sections 5.3.13.2 and equation (7.22)

Arguments

sigint	Interpolated sigma coordinates
kint	Counter for the interpolated coordinates
cdir	Direction of the gradient ('X', 'Y')

Calling procedures
baroclinic_gradient

buoyancy_frequency_sed

```
SUBROUTINE buoyancy_frequency_sed(bgrad)
REAL, INTENT(INOUT), DIMENSION(0:ncloc+1,0:nrloc+1,2:nz) :: bgrad
```

File
Sediment_Density_Equations.F90

Type
Subroutine

Purpose
Sediment contribution to the squared buoyancy frequency N^2

Reference
Equation (7.20)

Arguments

bgrad	(Non-averaged) buoyancy gradient	$[1/s^2]$

Calling procedures
buoyancy_frequency

equation_of_state_sed

```
SUBROUTINE equation_of_state_sed
```

File
Sediment_Density_Equations.F90

Type
Subroutine

Purpose
Density and expansion coefficients of a water-sediment mixture

Reference
Section 7.2.3

Arguments
None

Calling procedures
equation_of_state

34.1.4 *Sediment_Equations.F90*

ackerswhite_params

```
SUBROUTINE ackerswhite_params(dstar,maskvals,nw,mw,aw,cw)
LOGICAL,  INTENT(IN), DIMENSION(1:ncloc,1:nrloc), :: maskvals
REAL,  INTENT(INOUT), DIMENSION(1:ncloc,1:nrloc) :: dstar
REAL,  INTENT(OUT), DIMENSION(1:ncloc,1:nrloc) :: aw, cw, mw, nw
```

File
Sediment_Equations.F90

Type
Subroutine

Purpose
Parameters for the Ackers & White (1973) formula

Reference
Equation (7.91)

Arguments

dstar	Dimensionless particle diameter d_*
maskvals	Mask array to exclude dry points
nw, mw, aw, cw	Parameters for the Ackers & White (1973) formula

Calling procedures
sediment_suspendedload, sediment_totalload

bartnicki_filter

```
SUBROUTINE bartnicki_filter(psic,novars,ivarid)
INTEGER,INTENT(IN):: ivarid, novars
REAL, INTENT(INOUT), DIMENSION(1-nhalo:ncloc+nhalo,&
                            & 1-nhalo:nrloc+nhalo,nz,novars) :: psic
```

File
Sediment_Equations.F90

Type
Subroutine

Purpose
Apply the Bartnicki filter to a scalar field to eliminate negative values

Reference
Section 7.7.4

Arguments

psic	C-node scalar quantity	[psic]
novars	Size of last "variable" dimension	
ivarid	Key id of psic	

Calling procedures
sediment_advdiff

bed_slope_arrays

```
SUBROUTINE bed_slope_arrays
```

File
Sediment_Equations.F90

Type
Subroutine

Purpose
Bed slopes at different grid nodes

Arguments
None

Calling procedures
sediment_equation

beta_factor

`SUBROUTINE beta_factor`

File
Sediment_Equations.F90

Type
Subroutine

Purpose
Ratio of sediment diffusivity to eddy viscosity β

Reference
Equation (7.136)

Arguments
None

Calling procedures
sediment_advdiff

diff_coef_waves

`SUBROUTINE diff_coef_waves`

File
Sediment_Equations.F90

Type
Subroutine

Purpose
Sediment diffusivity including wave effects

Reference
Section 7.6.4.2

Arguments
None

Called external procedures
bottom_stress_waves

Calling procedures
sediment_advdiff

equilibrium_concentration

`SUBROUTINE equilibrium_concentration`

File
Sediment_Equations.F90

Type
Subroutine

Purpose
Determine equilibrium concentration c_a

Reference
Section 7.6.3

Arguments
None

Called external procedures
equilibrium_timescale, sediment_suspendedload

Calling procedures
sediment_advdiff

equilibrium_timescale

`SUBROUTINE equilibrium_timescale`

File
Sediment_Equations.F90

Type
Subroutine

Purpose
Determine dimensionless equilibrium time scale T_e

Reference
Equations (7.130)–(7.133)

Arguments
None

Calling procedures
equilibrium_concentration

median_particle_diameter

`SUBROUTINE median_particle_diameter`

File
Sediment_Equations.F90

Type
Subroutine

Purpose
Calculates the median particle diameter d_{50} from a size distribution

Arguments
None

Calling procedures
sediment_equation

sediment_advdiff

`SUBROUTINE sediment_advdiff`

File
Sediment_Equations.F90

Type
Subroutine

Purpose
Solves the advection-diffusion transport equations for each size fraction

Reference
Section 7.6

Arguments
None

Called external procedures
bartnicki_filter, beta_factor, bottom_flux_sed, define_profobc_spec, diff_coef_waves, equilibrium_concentration, open_boundary_conds_prof, settling_velocity, transport_at_C_4d1, transport_at_C_4d2, update_nest_data_prof, update_profobc_data

Calling procedures
sediment_equation

sediment_bedload

`SUBROUTINE sediment_bedload`

File
Sediment_Equations.F90

Type
Subroutine

Purpose
Apply one of the formulations for bed load transport

Reference
Section 7.4

Arguments
None

Called external procedures
bottom_stress_waves

Calling procedures
sediment_equation, sediment_totalload

sediment_equation

`SUBROUTINE sediment_equation`

File
Sediment_Equations.F90

Type
Subroutine

Purpose
Main sediment routine

Reference
Chapter 7

Arguments
None

Called external procedures
bed_slope_arrays, bottom_stress_sed, critical_shear_stress, kinematic_viscosity, median_particle_diameter, sediment_advdiff, sediment_bedload, sediment_totalload

Calling procedures
coherens_main, initialise_model

sediment_suspendedload

```
SUBROUTINE sediment_suspendedload
```

File
Sediment_Equations.F90

Type
Subroutine

Purpose
Apply one of the formulations for suspended load

Reference
Section 7.5

Arguments
None

Called external procedures
ackerswhite_params, bottom_flux_sed, bottom_flux_sed, thetastar_engelund_hansen

Calling procedures
equilibrium_concentration, sediment_equation

sediment_totalload

```
SUBROUTINE sediment_totalload
```

File
Sediment_Equations.F90

Type
Subroutine

Purpose
Apply one of the formulations for total load

Reference
Section 7.5

Arguments
None

Called external procedures
ackerswhite_params, bottom_flux_sed, sediment_bedload, settling_velocity, thetastar_engelund_hansen

Calling procedures
sediment_equation

settling_velocity

```
SUBROUTINE settling_velocity
```

File
Sediment_Equations.F90

Type
Subroutine

Purpose
Apply one of the formulations for the settling velocity

Reference
Section 7.3.3

Arguments
None

Calling procedures
sediment_advdiff, sediment_totalload

thetastar_engelund_hansen

```
SUBROUTINE thetastar_engelund_hansen(theta,mask,thetastar)
REAL, INTENT(IN), DIMENSION(ncloc,nrloc) :: theta
REAL, INTENT(OUT), DIMENSION(ncloc,nrloc) :: thetastar
LOGICAL, INTENT(IN), DIMENSION(ncloc,nrloc) :: mask
```

File
Sediment_Equations.F90

Type
Subroutine

Purpose
Computes θ_* in the Engelund & Hansen (1967) formula

Reference
Equation (7.87)

Arguments

theta	Shields parameter
thetastar	Modified Shields parameter for Engelund & Hansen (1967) formula
mask	Mask to exclude dry points

Calling procedures
sediment_suspendedload, sediment_totalload

34.1.5 *Sediment_Finalisation.f90*

write_sedics

```
SUBROUTINE write_sedics
```

File
Sediment_Finalisation.f90

Type
Subroutine

Purpose
Write the initial conditions for sediments to a file in standard COHERENS format.

Arguments
None

Calling procedures
coherens_main, initialise_model

write_sed_spec

```
SUBROUTINE write_sed_spec
```

File
Sediment_Finalisation.f90

Type
Subroutine

Purpose
Write the particle attributes to a file in standard COHERENS format.

Arguments
None

Calling procedures
initialise_model

34.1.6 *Sediment_Initialisation.f90*

exchange_sedics

```
SUBROUTINE exchange_sedics
```

File
Sediment_Initialisation.f90

Type
Subroutine

Purpose
Perform exchange communications for sediment arrays, storing the initial conditions into the respective halos.

Arguments
None

Calling procedures
initialise_model

initialise_sediment_arrays

`SUBROUTINE initialise_sediment_arrays`

File
Sediment_Initialisation.f90

Type
Subroutine

Purpose
Initialise a series of sediment arrays, not obtained from the initial conditions file.

Arguments
None

Calling procedures
initialise_model

read_sedics

`SUBROUTINE read_sedics`

File
Sediment_Initialisation.f90

Type
Subroutine

Purpose
Read the initial conditions for sediments from a file in standard COHERENS format.

Arguments
None

Calling procedures
initialise_model

read_sed_spec

```
SUBROUTINE read_sed_spec
```

File
Sediment_Initialisation.f90

Type
Subroutine

Purpose
Read the particle attributes from a file in standard COHERENS format.

Arguments
None

Calling procedures
initialise_model

34.1.7 *Sediment_Parameters.f90*

assign_cif_vars_sed

```
SUBROUTINE assign_cif_vars_sed(cname,cvals,numvars)
CHARACTER (LEN=lenname), INTENT(IN), OPTIONAL :: cname
CHARACTER (LEN=lencifvar), INTENT(IN), DIMENSION(MaxCIFvars) :: cvals
INTEGER, INTENT(IN) :: numvars
```

File
Sediment_Parameters.f90

Type
Subroutine

Purpose
Convert a data string from a CIF input line to the appropriate numerical, logical or character values of the corresponding sediment variables.

Reference
Section 9.4

Arguments

cname	Variable name(s)
cvals	Data values in string format

numvars Number of sediment variables read from the CIF input line

Calling procedures
read_cif_params

write_cif_vars_sed

`SUBROUTINE write_cif_vars_sed`

File
Sediment_Parameters.f90

Type
Subroutine

Purpose
Write the CIF for sediment model setup.

Reference
Section 9.4

Arguments
None

Calling procedures
initialise_model

34.2 Module routines

34.2.1 *check_sediments.f90*

check_sed_ics

`SUBROUTINE check_sedics`

File
check_sediments.f90

Type
Module subroutine

Purpose
Check initial conditions for sediments, as e.g. defined in usrdef_sedics.

Arguments
None

Calling procedures
initialise_model

check_sed_params

`SUBROUTINE check_sed_params`

File
check_sediments.f90

Type
Module subroutine

Purpose
Check setup parameters and switches for the sediment model setup, as e.g. defined in usrdef_sed_params.

Arguments
None

Calling procedures
initialise_model

34.2.1.1 *default_sediments.f90*

default_sedics

`SUBROUTINE default_sedics`

File
default_sediments.f90

Type
Module subroutine

Purpose
Set the default initial conditions for the sediment model.

Arguments
None

Calling procedures
initialise_model

default_sed_params

```
SUBROUTINE default_sed_params
```

File
default_sediments.f90

Type
Module subroutine

Purpose
Default settings of sediment parameters and switches

Arguments
None

Calling procedures
initialise_model

34.2.2 *reset_sediments.f90*

reset_sedics

```
SUBROUTINE reset_sedics
```

File
reset_sediments.f90

Type
Module subroutine

Purpose
Reset the initial conditions of the sediment model where needed.

Arguments
None

Calling procedures
initialise_model

reset_sed_params

```
SUBROUTINE reset_sed_params
```

File
reset_sediments.f90

Type
Module subroutine

Purpose
Reset the parameters and switches of the sediment model where needed.

Arguments
None

Calling procedures
initialise_model

34.2.3 *sediment_output.f90*

define_sed0d_int2d

```
SUBROUTINE define_sed0d_int2d(ivarid,f,out2d)
INTEGER, INTENT(IN) :: f, ivarid
REAL, INTENT(OUT), DIMENSION(ncloc,nrloc) :: out2d
```

File
sediment_output.f90

Type
Module subroutine

Purpose
Define 2-D sediment data for area integrated/averaged output.

Arguments

ivarid	Variable key id
f	Fraction number (if needed)
out2d	Data area as defined on the model grid

Calling procedures
define_out0d_vals

define_sed0d_int3d

```
SUBROUTINE define_sed0d_int3d(ivarid,f,out3d)
INTEGER, INTENT(IN) :: f, ivarid
REAL, INTENT(OUT), DIMENSION(ncloc,nrloc,nz) :: out3d
```

File
sediment_output.f90

Type
Module subroutine

Purpose
Define 3-D sediment data for area integrated/averaged output.

Arguments

ivarid	Variable key id
f	Fraction number (if needed)
out3d	Data area as defined on the model grid

Calling procedures
define_out0d_vals

define_sed0d_vals

```
SUBROUTINE define_sed0d_vals(ivarid,f,outdat)
INTEGER, INTENT(IN) :: f, ivarid
REAL, INTENT(OUT) :: outdat
```

File
sediment_output.f90

Type
Module subroutine

Purpose
Define 0-D sediment output data.

Arguments

ivarid	Variable key id
f	Fraction number (if needed)
outdat	Returned output data value

Calling procedures
define_out0d_vals

define_sed2d_int1d

```
SUBROUTINE define_sed2d_int1d(ivarid,i,j,f,nzdim,out1d)
INTEGER, INTENT(IN) :: f, i, ivarid, j, nzdim
REAL, INTENT(OUT), DIMENSION(nzdim) :: out1d
```

File
sediment_output.f90

Type
Module subroutine

Purpose
Define sediment data for vertically integrated/averaged output.

Arguments

ivarid	Variable key id
i	X-index of the output location
j	Y-index of the output location
f	Fraction number (if needed)
nzdim	(Vertical) size of the profile
out1d	Returned vertical profile data

Calling procedures
define_out2d_vals

define_sed2d_vals

```
SUBROUTINE define_out2d_vals(ivarid,i,j,f,outdat)
INTEGER, INTENT(IN) :: f, i, ivarid, j
REAL, INTENT(OUT) :: outdat
```

File
sediment_output.f90

Type
Module subroutine

Purpose
Define 2-D sediment output data at a given location.

Arguments

ivarid	Variable key id
i	X-index of the output location
j	Y-index of the output location
f	Fraction number (if needed)
outdat	Returned output data

Calling procedures
define_out2d_vals

define_sed3d_vals

```
SUBROUTINE define_sed3d_vals(ivarid,i,j,k,f,cnode,outdat)
CHARACTER (LEN=lennode), INTENT(IN) :: cnode
INTEGER, INTENT(IN) :: f, i, ivarid, j, k
REAL, INTENT(OUT) :: outdat
```

File
sediment_output.f90

Type
Module subroutine

Purpose
Define 3-D output data at a given location.

Arguments

ivarid	Variable key id
i	X-index of the output location
j	Y-index of the output location
k	vertical index of the output location
f	Fraction number (if needed)
cnode	Defines the vertical node, i.e. 'C' or 'W', where a W-node quantity is evaluated (see Section 20.1.1.1 for details)
outdat	Returned output data

Calling procedures
define_out2d_vals, define_out3d_vals

34.2.4 *sedvars_routines.f90*

inquire_sedvar

```
SUBROUTINE inquire_sedvar(varid,f90_name,long_name,units,node,vector_name,&
                          & data_type,nodim,shape,dom_dims,halo_size,varatts)
CHARACTER (LEN=lenname), INTENT(OUT) :: f90_name
CHARACTER (LEN=lendesc), INTENT(OUT) :: long_name, vector_name
CHARACTER (LEN=lenunit), INTENT(OUT) :: units
CHARACTER (LEN=lennode), INTENT(OUT) :: node
INTEGER, INTENT(IN) :: varid
INTEGER, INTENT(OUT) :: data_type, nodim
INTEGER, INTENT(OUT), DIMENSION(4) :: dom_dims, halo_size, shape
TYPE (VariableAtts), INTENT(OUT) :: varatts
```

File

sedvars_routines.f90

Type

Module subroutine

Purpose

Returns attributes of a sediment array variable given its variable key id.

Arguments

varid	Variable key id
f90_name	Returned FORTRAN 90 name attribute
long_name	Returned long name attribute
units	Returned units attribute
node	Returned grid node where the variable is defined on the model grid
vector_name	Returned vector name attribute
data_type	Returned data type attribute
nodim	Returned array rank
shape	Returned array shape
dom_dims	Returned array shape without halo
halo_size	Returned halo sizes
varatts	Variable attributes returned as a variable of type VariableAtts

Calling procedures
inquire_var

set_sedfiles_atts

```
SUBROUTINE set_sedfiles_atts(iddesc,ifil,iotype)
INTEGER, INTENT(IN) :: iddesc, ifil, iotype
```

File
sedvars_routines.f90

Type
Module subroutine

Purpose
Define the global attributes of a forcing file for the sediment model.

Arguments

iddesc	Forcing file key id
ifil	File number of the forcing file
iotype	I/O type of the forcing file
	1: Input file
	2: Output file

Calling procedures
set_modfiles_atts

set_sedvars_atts

```
SUBROUTINE set_sedvars_atts(iddesc,ifil,ivarid,nrank,nshape,numvarsid,novars)
INTEGER, INTENT(IN) :: iddesc, ifil, novars
INTEGER, INTENT(OUT), DIMENSION(novars) :: ivarid, nrank, numvarsid
INTEGER, INTENT(OUT), DIMENSION(novars,4) :: nshape
```

File
sedvars_routines.f90

Type
Module subroutine

Purpose
Define the variable attributes of a forcing file for the sediment model.

Arguments

iddesc	Forcing file key id
ifil	File number of the forcing file
ivarid	Returned variable key idids
nrank	Returned variable ranks
nshape	Returned variable shapes
numvarsid	Returned variable indices in case the last dimension is a variable dimension, undefined otherwise
novars	Number of variables (coordinate plus data) in the forcing file

Calling procedures
set_modvars_atts

34.3 User defined routines

34.3.1 *Usrdef_Sediment.f90*

usrdef_sed_params

```
SUBROUTINE usrdef_sed_params
```

File
Usrdef_Sediment.f90

Type
Subroutine

Purpose
Define switches and parameters for the sediment model.

Arguments
None

Calling procedures
initialise_model

usrdef_sedics

```
SUBROUTINE usrdef_sedics
```

File
Usrdef_Sediment.f90

Type
Subroutine

Purpose
Define initial conditions for the sediment model.

Arguments
None

Calling procedures
initialise_model

usrdef_sed_spec

```
SUBROUTINE usrdef_sed_spec
```

File
Usrdef_Sediment.f90

Type
Subroutine

Purpose
Define particle attributes for the sediment fractions.

Arguments
None

Calling procedures
initialise_model

34.4 Sediment model variables

34.4.1 Sediment arrays

```
REAL, ALLOCATABLE, DIMENSION(:) :: dp, rhos, tau_cr_cst, ws_cst
REAL, ALLOCATABLE, DIMENSION(:,:) :: bdragcoefatc_sed, bdragcoefatu_sed, &
```

```
    & bdragcoefatv_sed, bed_slope_x, bed_slope_x_atu, bed_slope_x_atv, &
    & bed_slope_y, bed_slope_y_atu, bed_slope_y_atv, bstresatc_sed, &
    & bstresatc_wav, bstresatu_wav, bstresatv_wav, d50_bed, ubstresatc_sed, &
    & ubstresatu_sed, ubstresatv_sed, vbstresatc_sed, vbstresatu_sed, &
    & vbstresatv_sed, zroughatc_sed, zroughatu_sed, zroughatv_sed
REAL, ALLOCATABLE, DIMENSION(:,:,:) :: bed_fraction, beta_sed, &
    & bottom_sed_flux, ceq, cref, ctot, height_c, qbedatu, qbedatv, &
    & qsusatc, qtotatu, qtotatv, tau_cr, t_equil
INTEGER, ALLOCATABLE, DIMENSION(:,:,:) :: cell_layer
REAL, ALLOCATABLE, DIMENSION(:,:,:,:) ::  beta_state_sed, cvol, sedsrcuser, &
                                          & obcsedatu, obcsedatv
                                          & vdiffcoef_sed, wfall
```

File

sedarrays.f90

Type

Module

Purpose

Sediment arrrays

Description

bdragcoefatc_sed	Bottom drag coefficient at the C-nodes using skin roughness
bdragcoefatu_sed	Bottom drag coefficient at the U-nodes using skin roughness
bdragcoefatv_sed	Bottom drag coefficient at the V-nodes using skin roughness
bed_fraction	Fractional amounts of sediments at the sea bed (between 0 and 1)
bed_slope_x	Bed slope in the X-direction at the C-nodes
bed_slope_x_atu	Bed slope in the X-direction at the U-nodes
bed_slope_x_atv	Bed slope in the X-direction at the V-nodes
bed_slope_y	Bed slope in the Y-direction at the C-nodes
bed_slope_y_atu	Bed slope in the Y-direction at the U-nodes
bed_slope_y_atv	Bed slope in the Y-direction at the V-nodes

beta_sed	Ratio between sediment eddy diffusivity and eddy viscosity β
beta_state_sed	Expansion factor β_c in equation of state (per sediment fraction) [m^3/m^3]
bottom_sed_flux	Upward net sediment flux at the sea bed $E - D$ per fraction [m/s]
bstresatc_sed	Bed shear stress using skin roughness at the C-nodes [m^2/s^2]
bstresatc_wav	Bed shear stress due to wave effects at the C-nodes [m^2/s^2]
bstresatu_wav	Bed shear stress due to wave effects at the U-nodes [m^2/s^2]
bstresatv_wav	Bed shear stress due to wave effects at the V-nodes [m^2/s^2]
cell_layer	The vertical layer (cell) for applying the bottom sediment flux
ceq	Equilibrium sediment concentration c_{eq} (per fraction) [m^3/m^3]
cref	Near bed reference sediment concentration c_a (at height_c) per fraction [m^3/m^3]
ctot	Volumetric sediment concentration (sum over all fractions) [m^3/m^3]
cvol	Volumetric sediment concentration c (per sediment fraction) [m^3/m^3]
dp	Particle diameter d [m]
d50_bed	Median grain size (by mass) d_{50} at the sea bed [m]
height_c	The elevation a at which the near bed boundary conditions is applied (per fraction) [m]
obcsedatu	Storage array for sediment concentrations in case the open boundary conditions at the U-nodes require the solution of a differential equation in time
obcsedatv	Storage array for sediment concentrations in case the open boundary conditions at the V-nodes require the solution of a differential equation in time
qbedatu	Bed load q_{b1} (per sediment fraction) in the X-direction at the U-nodes [m^2/s]

qbedatv	Bed load q_{b1} (per sediment fraction) in the Y-direction at the V-nodes	[m^2/s]
qsusatc	Suspended load q_s (per sediment fraction) at the C-nodes	[m^2/s]
qtotatu	Total load q_{t1} (per sediment fraction) in the X-direction at the U-nodes	[m^2/s]
qtotatv	Total load q_{t2} (per sediment fraction) in the Y-direction at the V-nodes	[m^2/s]
rhos	Particle density ρ_s	[kg/m^3]
sedsrcuser	Sediment source defined by the user (per sediment fraction)	[m^3/s]
tau_cr	Critical bed shear stress τ_{cr} (per sediment fraction)	[m^2/s^2]
tau_cr_cst	Spatially uniform critical shear stress τ_{cr} per fraction	[m^2/s^2]
t_equil	Non-dimensional equilibrium time scale T_e (per fraction)	
ubstresatc_sed	X-component of the skin bed shear stress at the C-nodes	[m^2/s^2]
ubstresatu_sed	X-component of the skin bed shear stress at the U-nodes	[m^2/s^2]
ubstresatv_sed	X-component of the skin bed shear stress at the V-nodes	[m^2/s^2]
vbstresatc_sed	Y-component of the skin bed shear stress at the C-nodes	[m^2/s^2]
vbstresatu_sed	Y-component of the skin bed shear stress at the U-nodes	[m^2/s^2]
vbstresatv_sed	Y-component of the skin bed shear stress at the V-nodes	[m^2/s^2]
vdiffcoef_sed	Vertical eddy diffusivity D_V for sediment (per fraction)	[m^2/s]
wfall	Settling velocity w_s (per sediment fraction)	[m/s]
ws_cs	Spatially uniform settling velocity w_s per fraction	[m/s]
zroughatc_sed	Skin roughness length at the C-nodes	[m]
zroughatu_sed	Skin roughness length at the U-nodes	[m]
zroughatv_sed	Skin roughness length at the V-nodes	[m]

34.4.2 Key ids of sediment variables

```
MODULE sedids
!---start key id number
INTEGER, PARAMETER, PRIVATE :: n0 = MaxModArids
!---sediment particle attributes
INTEGER, PARAMETER :: &
& iarr_dp = n0+1, iarr_rhos = n0+2, iarr_tau_cr_cst = n0+3,
....
```

File
sedids.f90

Type
Module

Purpose
Definitions of key ids for sediment variables. The key id name has the form iarr_⋆ where ⋆ is the FORTRAN name of the variable.

34.4.3 Sediment model parameters

```
REAL, PARAMETER :: beta_sed_max = 1.5, beta_sed_min = 1.0
INTEGER::  maxitbartnicki = 100, nf = 1, nrquad_sed = 7, nrquad_wav = 10
REAL ::  alpha_VR = 2.19, a_leussen = 0.02, beta_sed_cst = 1.0, &
       & b_leussen = 0.0024, cgel = 0.0, cmax = 0.65, coef_bed_grad = 1.3, &
       & floc_VR_max = 10.0, floc_VR_min = 1.0, height_c_cst = 0.01, &
       & maxRV = 0.1, minRV = 1.0E-05, n_RichZaki = 4.6, parth_coef = 1.0E-08, &
       & parth_exp = 1.0, wu_exp = -0.6, zrough_sed_cst, z0_coef = 30.0
```

File
sedarrays.f90

Type
Module

Purpose
Sediment arrrays

Description

alpha_VR — Exponent α in the flocculation equation (7.48) by Van Rijn (2007b)

a_leussen	Coefficient a in the (7.46) flocculation equation by Van Leussen (1994) [s]
beta_sed_cst	Constant value of the eddy diffusivity to viscosity ratio as used in equation (7.136) if iopt_sed_beta=2
beta_sed_max	Maximum value for the ratio β of sediment diffusivity to eddy viscosity
beta_sed_min	Minimum value for the ratio β of sediment diffusivity to eddy viscosity
b_leussen	Coefficient b in the (7.46) flocculation equation by Van Leussen (1994) [s^2]
cgel	Volumetric gelling concentration used for hindered settling of mud and flocculation [m^3/m^3]
cmax	Volumetric maximum concentration for sand at the sea bed used in equations (7.104), (7.114) for total load and for calulating the reference concentration in the Smith & McLean (1977) formula (7.124) [m^3/m^3]
coef_bed_grad	Coefficient β_s used in the bed slope formula (7.83) of Koch & Flokstra (1981)
floc_VR_max	Maximum value for the flocculation factor ϕ_{loc} in equation (7.48) by Van Rijn (2007b)
floc_VR_min	Minimum value for the flocculation factor ϕ_{loc} in equation (7.48) by Van Rijn (2007b)
height_c_cst	Constant reference height a (normalised by the water depth) if iopt_sed_bbc=0
maxitbartnicki	Maximum number of iterations used by the bartnicki filter
maxRV	Maximum value for the reference height a (normalised by the water depth)
minRV	Minimum value for the reference height a (normalised by the water depth)
nf	Number of sediment fractions
nrquad_sed	Number of vertical locations used by the Gauss-Legendre numerical integration scheme for depth averaging of sediment (equilibrium) sediment profiles
nrquad_wav	Number of time steps used by the Gauss-Legendre numerical integration scheme for phase-averaging over a wave period

n_RichZaki	Exponent n in equation (7.43) for hindered settling by Richardson & Zaki (1954)
parth_coef	Coefficient M in the formulation (7.126) for erosion of mud by Partheniades (1965) [m/s]
parth_exp	Exponent n_p in the formulation (7.126) for erosion of mud by Partheniades (1965)
wu_exp	Exponent m used to calculate the hiding factor (7.36) in the Wu *et al.* (2000) formulation
zrough_sed_cst	Uniform roughness length used to obtain the (skin) bed stress if iopt_sed_tau=2 [m]
z0_coef	Factor by which z_0 is multiplied to determine the minimum depth for averaging used in the boundary condition at the sea bed in the EFDC method (iopt_sed_bbc_type = 2 or 3)

34.4.4 Sediment switches

```
!---on-off
INTEGER :: iopt_sed_mode = 2, iopt_sed_nodim = 3, iopt_sed_type = 2
!---type of equation
INTEGER :: iopt_sed_bedeq = 1, iopt_sed_ceqeq = 1, &
         & iopt_sed_dens = 0, iopt_sed_toteq = 1
!---suspended sediment transport
INTEGER :: iopt_sed_bbc= 1, iopt_sed_beta = 1, iopt_sed_wave_diff = 0
!---settling velocity
INTEGER :: iopt_sed_floc = 0, iopt_sed_hindset = 0, iopt_sed_vadv = 3, &
         & iopt_sed_ws = 1
!---bed load
INTEGER ::  iopt_sed_eha = 1, iopt_sed_hiding = 0, iopt_sed_median = 1, &
          & iopt_sed_slope = 0, iopt_sed_tau = 1, iopt_sed_taucr = 1
!---numerical
INTEGER :: iopt_sed_bbc_type = 3, iopt_sed_filter = 0
```

File
sedarrays.f90

Type
Module

Purpose
Sediment arrrays

Description

iopt_sed_bbc Type of boundary condition at the sea bed.

- 0: no bed boundary conditions (no flux to and from the bed)
- 1: using reference concentration (7.124) from Smith & McLean (1977)
- 2: using reference concentration from (7.125) Van Rijn (1984a)
- 3: deposition taken as an advective flux at the bottom, erosion parameterised using equation (7.126) from Partheniades (1965)

iopt_sed_bbc_type Selects the method to transpose the near bed boundary condition to the computational grid (see Section 7.7.1.1). It is strongly recommended not to change the default value.

- 1: EFDC method applied to lowest cell (not recommended)
- 2: EFDC method applied to the first the cell above the bottom (not recommended)
- 3: using the Rouse profile

iopt_sed_bedeq Type of formulation for bed load transport.

- 1: Meyer-Peter & Müller (1948)
- 2: Engelund & Fredsøe (1976)
- 3: Van Rijn (1984b)
- 4: Wu *et al.* (2000)
- 5: Soulsby (1997). This equation includes wave effects.
- 6: Van Rijn (2003). This formula includes wave effects.
- 7: Van Rijn (2007a). This method includes wave effects.

iopt_sed_beta The type of equation used for β, the ratio between the eddy viscosity and eddy diffusivity.

- 1: $\beta = 1$
- 2: β is defined by the user (parameter beta_cst).

3: Van Rijn (1984b) formulation (7.136)

iopt_sed_ceqeq — The type of model for determining the equilibrium sediment concentration used to evaluate the sediment flux at the sea bed for 2-D sand transport.

1: numerical integration of the Rouse profile

2: using q_t/U determined with the equation of Engelund & Hansen (1967). The precise form is also determined by the switch iopt_sed_eha.

3: using q_t/U using the formulation by Ackers & White (1973)

4: using q_s/U using the formulation by Van Rijn (2003). This formulation is very similar to Van Rijn (1984b), but takes wave stresses into account.

5: Using q_s/U and the method of Wu *et al.* (2000).

iopt_sed_dens — Disables (0) or enables (1) effects of sediments in the equation of state and density stratification.

iopt_sed_eha — Switch to select the type of formulation in the Engelund & Hansen (1967) total load formula. (7.86).

1: original form

2: Chollet & Cunge (1979) form as function of θ_*

iopt_sed_filter — The type of filter used to prevent the occurrence of negative concentrations.

0: no filter

1: Bartnicki (1989) filter.

iopt_sed_floc — Type of flocculation factor for the settling velocity.

0: flocculation effect disabled

1: Van Leussen (1994) equation (7.46)

2: Van Rijn (2007b) equation (7.48)

3: combination of the two previous methods

iopt_sed_hiding — Type of formulation for the hiding factor.

0: hiding disabled

1: Wu *et al.* (2000) equation (7.36)

2: Ashida & Michiue (1972) equation (7.37)

iopt_sed_hindset	Type of formulation for hindered settling. 0: hindered settling disabled 1: Richardson & Zaki (1954) equation (7.43) 2: Winterwerp & van Kesteren (2004) formula (7.44)
iopt_sed_median	Method for calculating the median size d_{50} at the sea bed. 1: no interpolation 2: linear interpolation (not recommended, especially for a low number of fractions)
iopt_sed_mode	Type of mode for applying the sediment transport model. 1: bedload transport only computed by a formula, which is determined by iopt_sed_bedeq 2: suspended load transport only (computed with the advection-diffusion equation) 3: bedload and suspended transport (i.e. option 1 and 2 together) 4: total load transport computed with a formula, which is determined by iopt_sed_toteq
iopt_sed_nodim	Type of grid mode for the sediment transport. 2: depth averaged transport[1] 3: 3-D sediment transport
iopt_sed_slope	Bed slope effects for bed load. 0: bed slope effects disabled 1: bed slope effect effects enabled and using the Koch & Flokstra (1981) formulation for bed load
iopt_sed_tau	Type of roughness length formulation for sediments. 1: the same as for the hydrodynamics 2: used-defined constant roughness length zrough_sed_cst 3: user-defined spatially non-uniform value
iopt_sed_taucr	Selects type of method for the critical shear stress. 1: user-defined value for each fraction

[1]Note that iopt_sed_nodim is always set to 2 if iopt_grid_nodim = 2.

2: Brownlie (1981) equation (7.31)

3: Soulsby & Whitehouse (1997) equation (7.32)

4: Wu *et al.* (2000) equation (7.33)

iopt_sed_toteq Type of method for total load transport.

1: Engelund & Hansen (1967). The precise form is also determined by the switch iopt_sed_eha.

2: Ackers & White (1973)

3: Madsen & Grant (1976). This equation includes wave effects.

4: Wu *et al.* (2000). Total load is calculated as the sum of suspended and bed load.

5: Van Rijn (2003). This equation includes wave effects and total load is the sum of suspended and bed load.

6: Van Rijn (2007a). This equation includes wave effects and total load is the sum of suspended and bed load.

iopt_sed_type Type of sediment.

1: sand (non-cohesive)

2: mud (cohesive)

iopt_sed_vadv Disables (0), enables (>0) vertical settling of sediments and selects the type of numerical advection scheme if >0 and vertical advection for (non-sediment) scalars is disabled (iopt_adv_scal=0). If iopt_adv_scal>0, then either iopt_sed_vadv=0 or set equal to iopt_adv_scal.

iopt_sed_wave_diff Selects the turbulent diffusion coefficient due to waves.

0: no diffusion coefficient

1: according to Van Rijn (2007b)

iopt_sed_ws Type of method for the settling velocity.

1: user-defined value for each fraction

2: Camenen (2007) formulation (7.39) for sand

3: Camenen (2007) formulation (7.39) for mud

4: Stokes formula (7.40)

5: Soulsby (1997) formula (7.41)

6: Zhang & Xie (1993) equation (7.42)

34.5 Interfaces

A series of "generic" routines for sediment transport are called from the physical ("main") part of COHERENS. If the user wants to couple COHERENS with an alternative sediment module, different from the one in COHERENS, the user must provide these routines. Empty routines, with only a declaration part of arguments are allowed. A complete list is given below.

The routine declarations are given below without further explanation. Further details about the meaning of the routine and arguments are found in this chapter.

34.5.1 External routine interfaces

```
SUBROUTINE allocate_sed_arrays
! Allocate sediment arrays
...
END SUBROUTINE allocate_sed_arrays

SUBROUTINE assign_cif_vars_sed(iddesc,cname,cvals,numvars)
! convert data strings from the sediment CIF to the format (numeric or
! non-numeric) in COHERENS
CHARACTER (LEN=lenname), INTENT(IN), OPTIONAL :: cname
CHARACTER (LEN=lencifvar), INTENT(IN), DIMENSION(MaxCIFvars) :: cvals
INTEGER, INTENT(IN) :: iddesc, numvars
...
END SUBROUTINE assign_cif_vars_sed

SUBROUTINE baroclinic_gradient_sed_cubic(zcoord,dzx,dzy,dzz,cdir)
! sediment contribution for the baroclinic
! density gradient (cube-H method)
CHARACTER (LEN=1) :: cdir
REAL, DIMENSION(0:ncloc+1,0:nrloc+1,nz), INTENT(IN) ::  dzx, dzy, dzz
REAL,DIMENSION(1-nhalo:ncloc+nhalo,&
             & 1-nhalo:nrloc+nhalo,nz), INTENT(IN) :: zcoord
...
END SUBROUTINE baroclinic_gradient_sed_cubic

SUBROUTINE baroclinic_gradient_sed_sigma(zcoord,cdir)
! sediment contribution for the baroclinic
! density gradient (sigma second order method)
CHARACTER (LEN=1) :: cdir
```

```
REAL,DIMENSION(1-nhalo:ncloc+nhalo,&
             & 1-nhalo:nrloc+nhalo,nz), INTENT(IN) :: zcoord
...
END SUBROUTINE baroclinic_gradient_sed_sigma

SUBROUTINE baroclinic_gradient_sed_z(sigint,kint,cdir)
! sediment contribution for the baroclinic
! density gradient (z-level method)
CHARACTER (LEN=1) :: cdir
INTEGER, DIMENSION(ncloc,nrloc,2:nz+1,2), INTENT(IN) :: kint
REAL, DIMENSION(ncloc,nrloc,2:nz+1,2), INTENT(IN) :: sigint
...
END SUBROUTINE baroclinic_gradient_sed_z

SUBROUTINE buoyancy_frequency_sed(bgrad)
! sediment contribution to the squared buoyancy frequency
REAL, DIMENSION(0:ncloc+1,0:nrloc+1,2:nz), INTENT(INOUT) :: bgrad
...
END SUBROUTINE buoyancy_frequency_sed

SUBROUTINE deallocate_sed_arrays
! deallocate sediment arrays
...
END SUBROUTINE deallocate_sed_arrays

SUBROUTINE equation_of_state_sed
! sediment contribution to density, sediment expansion coefficient
...
END SUBROUTINE equation_of_state_sed

SUBROUTINE exchange_sedics
! exchange the sediment arrays defined by the sediment initial conditions
...
END SUBROUTINE exchange_sedics

SUBROUTINE initialise_sediment_arrays
! initialise sediment model
...
END SUBROUTINE initialise_sediment_arrays

SUBROUTINE read_sedics
```

```
! read the initial conditions for the sediment model from
! a file in standard COHERENS format
...
END SUBROUTINE read_sedics

SUBROUTINE read_sed_spec
! read sediment specifiers (particle attributes) from
! a file in standard COHERENS format
...
END SUBROUTINE read_sed_spec

SUBROUTINE sediment_equation
! main unit of the sediment transport model
...
END SUBROUTINE sediment_equation

SUBROUTINE write_cif_vars_sed
! write the sediment CIF
...
END SUBROUTINE write_cif_vars_sed

SUBROUTINE write_sedics
! write the initial conditions for the sediment model to
! a file in standard COHERENS format
...
END SUBROUTINE write_sedics

SUBROUTINE write_sed_spec
! write sediment specifiers (particle attributes) to
! a file in standard COHERENS format
...
END SUBROUTINE write_sed_spec
```

34.5.2 Module routine interfaces

```
MODULE check_sediments
! check sediment model parameters and arrays for errors
CONTAINS
SUBROUTINE check_sedics
! check initial conditions in the sediment model
```

```
...
END SUBROUTINE check_sedics
SUBROUTINE check_sed_params
! check sediment switches and parameters
...
END SUBROUTINE check_sed_params
END MODULE check_sediments

MODULE default_sediments
! default settings for the sediment model
CONTAINS
SUBROUTINE default_sedics
! default initial conditions in the sediment model
...
END SUBROUTINE default_sedics
SUBROUTINE default_sed_params
! default settings for sediment switches and parameters
...
END SUBROUTINE default_sed_params
END MODULE default_sediments

MODULE reset_sediments
! reset sediment parameters and arrays where needed
CONTAINS
SUBROUTINE reset_sedics
! reset initial conditions in the sediment model
...
END SUBROUTINE reset_sedics
SUBROUTINE reset_sed_params
! reset sediment switches and parameters
...
END SUBROUTINE reset_sed_params
END MODULE reset_sediments

MODULE sediment_output
! routines for sediment user output data
CONTAINS
SUBROUTINE define_sed0d_int2d(ivarid,f,out2d)
! define 2-D sediment data for area integrated/averaged output
INTEGER, INTENT(IN) :: f, ivarid
REAL, INTENT(OUT), DIMENSION(ncloc,nrloc) :: out2d
```

```
...
END SUBROUTINE define_sed0d_int2d
SUBROUTINE define_sed0d_int3d(ivarid,f,out3d)
! define 3-D sediment data for area integrated/averaged output
INTEGER, INTENT(IN) :: f, ivarid
REAL, INTENT(OUT), DIMENSION(ncloc,nrloc,nz) :: out3d
...
END SUBROUTINE define_sed0d_int3d
SUBROUTINE define_sed0d_vals(ivarid,f,outdat)
! define 0-D sediment output data
INTEGER, INTENT(IN) :: f, ivarid
REAL, INTENT(OUT) :: outdat
...
END SUBROUTINE define_sed0d_vals
SUBROUTINE define_sed2d_int1d(ivarid,i,j,f,nzdim,out1d)
! define sediment data for vertically integrated/averaged output
INTEGER, INTENT(IN) :: f, i, ivarid, j, nzdim
REAL, INTENT(OUT), DIMENSION(nzdim) :: out1d
...
END SUBROUTINE define_sed2d_int1d
SUBROUTINE define_sed2d_vals(ivarid,i,j,f,outdat)
! define 2-D sediment output data at a given location
INTEGER, INTENT(IN) :: f, i, ivarid, j
REAL, INTENT(OUT) :: outdat
...
END SUBROUTINE define_sed2d_vals
SUBROUTINE define_sed3d_vals(ivarid,i,j,k,f,cnode,outdat)
! define 3-D output data at a given location
CHARACTER (LEN=lennode) :: cnode
INTEGER, INTENT(IN) :: f, i, ivarid, j, k
REAL, INTENT(OUT) :: outdat
...
END SUBROUTINE define_sed3d_vals
END MODULE sediment_output

MODULE sedvars_routines
! attributes of sediment variables and forcing files
CONTAINS
SUBROUTINE inquire_sedvar(varid,f90_name,long_name,units,node,vector_name,&
                    & data_type,nodim,shape,dom_dims,halo_size,varatts)
! returns attributes of a sediment array variable given its
```

```
! variable key id
CHARACTER (LEN=lenname), INTENT(OUT), OPTIONAL :: f90_name
CHARACTER (LEN=lendesc), INTENT(OUT), OPTIONAL :: long_name, vector_name
CHARACTER (LEN=lenunit), INTENT(OUT), OPTIONAL :: units
CHARACTER (LEN=*), INTENT(OUT), OPTIONAL :: node
INTEGER, INTENT(IN) :: varid
INTEGER, INTENT(OUT), OPTIONAL :: data_type, nodim
INTEGER, INTENT(OUT), OPTIONAL, DIMENSION(4) :: dom_dims, halo_size, shape
TYPE (VariableAtts), INTENT(OUT), OPTIONAL :: varatts
...
END SUBROUTINE inquire_var
SUBROUTINE set_sedfiles_atts(iddesc,ifil,iotype)
! global attributes of a forcing file in the sediment model
INTEGER, INTENT(IN) :: iddesc, ifil, iotype
...
END SUBROUTINE set_sedfiles_atts
SUBROUTINE set_sedvars_atts(iddesc,ifil,ivarid,nrank,nshape,numvarsid,novars)
! variable attributes of a forcing file in the sediment model
INTEGER, INTENT(IN) :: iddesc, ifil, novars
INTEGER, INTENT(OUT), DIMENSION(novars) :: ivarid, nrank, numvarsid
INTEGER, INTENT(OUT), DIMENSION(novars,4) :: nshape
...
END SUBROUTINE set_sedvars_atts
END MODULE sedvars_routines
```

34.5.3 User defined routines

```
SUBROUTINE usrdef_sed_params
! define parameters and switches for the sediment model
...
END SUBROUTINE usrdef_sed_params
SUBROUTINE usrdef_sedics
! define intitial conditions for the sediment model
...
END SUBROUTINE usrdef_sedics
SUBROUTINE usrdef_sed_spec
! define particle properties in the sediment model
...
END SUBROUTINE usrdef_sed_spec
```

COHERENS

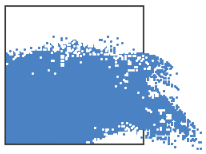
A Coupled Hydrodynamical-Ecological Model for Regional and Shelf Seas

Release Notes

Last update: August, 2013

Patrick Luyten

Royal Belgian Institute of Natural Sciences (RBINS-MUMM)
Gulledelle 100, 1200 Brussels, Belgium



museum

Version V2.0

Coherens Version	: V2.0
previous release	: V1 (release_8.4)
Revision	: 113
svn path	: http://svn.mumm.ac.be/svn/root/coherens/versions/V2.0
Date of release	: 2010-10-04
File (code)	: coherensV20_r109.tar.gz
File (manual)	: manualV20_r124.pdf

Description

This version represents a full update of the first version of coherens, released in April 2000 (more than ten years ago). Most important changes are:

1. The code is re-written in FORTRAN 90.
2. Implementation of parallelisation using the MPI message MPI library (available as an option).
3. Options for curvilinear grids in the horizontal and generalised σ-coordinates in the vertical.
4. One-way nesting
5. Standard formats (including netCDF) for forcing and user-defined output
6. "Usrdef" files where the user can define all model setup.
7. Improvement of numerical schemes:
   - baroclinic pressure gradients
   - additional types of open and surface boundary conditions
   - additional turbulence schemes

8. A drying/wetting algorithm.
9. Possibility to run the model in 1-D (water column), 2-D (depth-averaged mode) and 3-D mode
10. Forcing data can be read from several files with different time resolution.
11. Possibility to launch different simulation within one run.
12. The user can specify different times within the simulation period when initial conditions are written for eventual re-starts of the program.

Model code

See Coherens V2.0 User Manual

User instructions

See Coherens V2.0 User Manual

Version V2.1.0

Coherens Version	: V2.1.0
previous release	: V2.0
Revision	: 139
svn path	: http://svn.mumm.ac.be/svn/root/coherens/versions/V2.1.0
Date of release	: 2010-11-10
File (code)	: coherensV210_r136.tar.gz
File (manual)	: in preparation

Implementations

1. A new optional utility, called the CIF ("Central Input File") has been implemented, which can be considered as an extension of the *namelist* file utility. The aim is to define all model setup parameters by reading the CIF instead of calling a usrdef_ routine.

   - Parameters can be re-set by editing the CIF without re-programming the usrdef_ routines.
   - The following routines are no longer called if the CIF option is selected.

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

   - The CIF can be created by the user as an ASCII file or be generated by the program itself.
   - The syntax of the file is described below.

2. Forcing parameters and data, defined in a usrdef_ routine, can be written to a file in standard COHERENS format. Although the utility was

already available in the previous version, a series of program bugs had to be removed. The utility has been fully tested.

- The output file is created and written (eventually over-written) if the status file attribute is set to 'W', i.e. modfiles(idesc,ifil,2)%status='W'
- The following usrdef_ calls can be made redundant in this way

  `usrdef_partition, usrdef_phsics, usrdef_1dsur_spec, usrdef_2dobc_spec, usrdef_profobc_spec, usrdef_1dsur_data, usrdef_2dobc_data, usrdef_profobc_data, usrdef_rlxobc_spec, usrdef_surface_absgrd, usrdef_surface_relgrd, usrdef_surface_data, usrdef_nstgrd_spec, usrdef_surface_nstgrd_abs, usrdef_surface_nstgrd_rel`

3. The routine usrdef_parallel has been removed. The three parameters, which could previously be defined in this routines, are now defined as follows

   parallel_set — The parameter is automatically set to .TRUE. if the compiler option -DMPI is added in *options.cpp*, and to .FALSE. otherwise. In that case MPI will be initialised and finalised. For a parallel run the parameter nprocs must take a value greater than 1.

   shared_read — Shared reading by all processes is now taken as .TRUE. This means that the parameter (and its opposite noshared_read) are longer needed and they are therefore removed from the code. As a consequence all copy operations have been deleted in the source code. Details are given below.

   idmaster — The parameter is now defined in usrdef_mod_params (or in the CIF).

Instructions for users

The *defruns* file

Three parameters are read from each line of the *defruns*, separated by a ','. The general systax is

```
runtitle,status,filename
```

where

runtitle The title of the simulation which has the same meaning as before

status The status of the CIF

- '0' The CIF utility is switched off (both for reading and writing). This is the default condition.
- 'R' Model setup parameters are read from a CIF.
- 'W' Model setup parameters are written to a CIF.

filename Name of the CIF file. If not given, the default name TRIM(runtitle)//'.cifmodA' is taken. This parameter is obviously not used if status equals '0'

Defaults are taken (except for runtitle which must always be given) when the value is an empty string, one blank or several blanks. All blanks are ignored on the input line.

Consider the following example

```
conesA,,
conesA,R,
conesA,W,myciffile
```

The first line initiates the run conesA without CIF, the second one reads the setup from the file *conesA.cifmodA*, the third writes the CIF data to the file *myciffile*.

Lines can be commented if the first character is a '!'. This replaces, for compatibility with the CIF syntax below, the '#' character used in previous versions.

The CIF file

syntax of a CIF

As shown in the example 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 is 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 each component in the order as 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 is not given by a numeric value but by its file descriptor format 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.

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, a block 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 between seconds and milliseconds in a date/time string is now replaced by a ‘:’.

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.

Test cases

The test case results are exactly the same as in the previous version (V2.0)[2]. To illustrate the use of the CIF utility, the test case runs can be set up in two modes, depending on different choices for the *defruns* file. In the first case, the *defruns* file located in the test case directory is taken and the setup is as before.

In the second case, instructions for installation are the same as before except that the following copy has to be made in the working directory

```
cp cifruns defruns
```

The simulation of a test case now proceeds in two phases:

1. The test is run with the cold_start option set to .TRUE. and the CIF status in defruns set to ‘W’. The program creates a CIF file and a series of forcing files in COHERENS standard format.

[2]A small change is seen in output parameters of test cases rhone and bohai due to a small bug in the program. This will be repaired in the next version.

2. The test is run again with the CIF status set to 'R' and input forcing using the previously written standard files.

Model code

The following changes are made with respect to version V2.0:

1. *Model_Initialisation.F90*

   - The switch parallel_set is defined in coherens_start.
   - Routine read_cif_mod_params is called from initialise_model if the CIF status is set to 'R'.
   - Routine simulation_start which reads the *defruns* file, is completely re-written. Error coding is provided.

2. *Model_Parameters.f90*
   The following routines are added

assign_cif_vars	Converts the data string(s) in the CIF input line to the appropriate numerical, logical or string format.
read_cif_mod_params	General routines for reading a CIF file. The actual data conversion is made by calling read_cif_line and assign_cif_vars.
write_cif_mod_params	Writes the CIF for physical model setup.

   Since copy operations between processes have been removed from the program, the routine copy_mod_params has been deleted.

3. *cif_routines.f90*
   This is a new file with utility routines needed for the CIF implementation.

4. Since reading is now always performed in shared mode, all copy operations invoked when the now deleted parameter shared_read was set to .FALSE. have been removed. This affects the routines in the following files

Grid_Arrays.F90	define_global_grid, open_boundary_arrays
Open_Boundary_Data_2D.f90	define_2dobc_data, define_2dobc_spec
Open_Boundary_Data_Prof.f90	define_profobc_data, define_profobc_spec

Nested_Grids.F90	define_nstgrd_locs, define_nstgrd_spec
Parallel_Initialisation.f90	domain_decomposition
Relaxation.f90	define_rlxobc_spec
Surface_Boundary_Data_1D.f90	define_1dsur_data, define_1dsur_spec
Surface_Data_1D.f90	define_surface_data
Surface_Grids.f90	define_surface_input_grid, define_surface_output_grid

For the same reason the following routines are deleted from the code:

inout_paral.f90	read_copy_vars
paral_comms.f90	copy_filepars, copy_gridpars_2d, copy_hgrid_2d, copy_outgpars_1d, copy_statlocs_1d, copy_varatts_1d

5. The default value for the status attribute of all forcing files is '0' (undefined). This means that a usrdef_ routine (except usrdef_init_params) is called only if the status of the corresponding forcing file is reset to 'R'.

Compatibility with previous versions

An application, which has been set up with version V2.0, can be made compatible with V2.1.0, taking account of the following changes:

1. The status attribute is set to '0' by default.

2. For application without CIF, each line in defruns should end by two colons (',,').

3. Lines in defruns are commented by putting a '!' instead of '#' as the first character on the input line.

4. The routines usrdef_parallel no longer exists.

5. The switch parallel_set is set automatically.

6. The switch shared_read no longer exists.

7. An (eventual) non default value for idmaster is defined in usrdef_mod_params.

Recommendations for developers

When a developer wants to couple the model with a new compartment (e.g. biology, sediments, ...), a separate CIF can be created to read or write the setup parameters for the specific compartment. Taking the case of a new sediment module (activated when the switch iopt_sed>0) as an example, this can be implemented as follows:

1. Increase the value of the system parameter MaxCIFTypes in syspars.f90 by 1.

2. Create a new CIF key id, e.g. icif_sed in *iopars.f90*.

3. The new CIF has two attributes, status and filename, which, contrary to the model CIF, are not defined in *defruns*, but by the usual program procedures

   - Set the values by default in default_mode_params:

     ```
     ciffiles(icif_sed)%status = '0'
     ciffiles(icif_sed)%filename = ''
     ```

   - The user can be activate the file by resetting the status and (eventually) the filename atribute in usrdef_mod_params:

     ```
     ciffiles(icif_sed)%status = ?
     ciffiles(icif_sed)%filename = ?
     ```

   - Since the two attributes are considered as general model parameters, their values should be included in the model CIF. This means that the following code has to be added in assign_cif_vars

     ```
     CASE ('CIFFILES')
        CALL check_cif_lbound_novars(iddesc,numvars,3)
        CALL conv_from_chars(cvals(1),icif,iddesc,1)
        CALL conv_from_chars(cvals(2),ciffiles(icif)%status,iddesc,2)
        CALL conv_from_chars(cvals(3),ciffiles(icif)%filename,iddesc,3)
     ```

     and in write_cif_mod_params

     ```
     icif_*: DO icif=3,MaxCIFTypes
        CALL conv_to_chars(cvals(1),icif)
        cvals(2) = ciffiles(icif)%status
        cvals(3) = ciffiles(icif)%filename
        CALL write_cif_line(iddesc,cvals(1:3),'CIFFILES')
     ENDDO icif_*
     ```

4. Create a new subroutine assign_cif_vars_sed for converting the CIF data strings to the values of the sediment parameters. The routine can be constructed using the formats given in assign_cif_vars.

5. Insert the following routine call at the beginning of routine assign_cif_vars:

```
IF (iddesc.EQ.icif_sed) CALL assign_cif_vars_sed
```

6. Read the new CIF at the appropriate place in routine initialise_model:

```
IF (ciffiles(icif_sed)%status.EQ.'R') THEN
  CALL read_cif_mod_params(icif_sed,iopt_sed.GT.0.0)
ENDIF
```

7. Write the new CIF if requested by adding the following code in coherens_main:

```
IF (master.AND.iopt_sed.GT.0.AND.ciffiles(icif_sed)%status.EQ.'W') THEN
  CALL write_cif_mod_params(icif_sed)
  CALL close_filepars(ciffiles(icif_sed))
ENDIF
```

Parts of the CIF produced for test case plume.

```
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 = 2
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.
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,width,Plume width,km,
TSRVARS = 2,0,0,hfront,Plume length,km,
TSRVARS = 3,161,2,umvel,X-component of depth-mean current,m/s,
          Depth-mean current
TSRVARS = 4,170,2,vmvel,Y-component of depth-mean current,m/s,
          Depth-mean current
TSRVARS = 5,113,2,zeta,Surface elevation,m,
TSRVARS = 6,162,3,uvel,X-component of current,m/s,Current
TSRVARS = 7,171,3,vvel,Y-component of current,m/s,Current
TSRVARS = 8,175,3,wphys,Physical vertical velocity,m/s,Physical current
TSRVARS = 9,204,3,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 = 1,161,2,umvel,X-component of depth-mean current,m/s,
           Depth-mean current
ANALVARS = 2,170,2,vmvel,Y-component of depth-mean current,m/s,
           Depth-mean current
ANALVARS = 3,113,2,zeta,Surface elevation,m,
ANALVARS = 4,162,3,uvel,X-component of current,m/s,Current
ANALVARS = 5,171,3,vvel,Y-component of current,m/s,Current
ANALVARS = 6,175,3,wphys,Physical vertical velocity,m/s,Physical current
ANALVARS = 7,204,3,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
```

#

Version V2.1.1

Coherens Version	: V2.1.1
previous release	: V2.1.0
Revision	: 160
svn path	: http://svn.mumm.ac.be/svn/root/coherens/versions/V2.1.1
Date of release	: 2011-01-07
File (code)	: coherensV210_r149.tar.gz
File (manual)	: in preparation

Implementations

A verification procedure has been created through an external shell script for testing new developments or to compare different versions of the code. The script is documented elsewhere and is not considered as a new model development. However, since the script uses the results of all test cases, a few modifications of the model code and in the setup of the test cases were necessary.

1. A new CPP compiler switch VERIF together with a new model switch iopt_verif have been created. If the compiler option is inserted in *options.cpp*, the verification switch iopt_verif is set to 1. Otherwise, its value is 0. Its value cannot be re-defined by the user.

2. Once the verification switch is activated, the following parameters for time series output are reset (if necessary)

   - Only one output set is allowed, i.e. nosetstsr=1.
   - All output files are in netCDF format.
   - Names of output variables must be different from the names of the test parameters defined in Usrdef_Output.f90.

- User output files must keep their default attributes.
- The output grid must extend over the whole domain.

3. A few other changes and corrections have been made further discussed below.

Test cases

The setup of the test case parameters, defined in Usrdef_Output.f90, is changed as follows

cones The parameters xmin, xplus, ymin, yplus have the same meaning as before, but are calculated by interpolation giving a higher precision. Output at 12.5 minutes intervals.

csnsp Output at daily intervals.

river Output at 3 hours intervals.

plume Output at 3 hours intervals.

rhone Two new parameters etot and bdissip are added. Output at 6 hours intervals.

Model code

The following changes have been with repsect to version V2.1.0.

1. Implementations for the verification procedure.

   Model_Initialisation.F90 The switch iopt_verif is defined in coherens_start depending on the whether VERIF is set in *options.cpp*.

   reset_model.F90 Attributes of the user output files and output grid are reset in routines reset_out_files and reset_out_gpars if iopt_verif=1.

2. Routine default_out_gpars in *default_model.f90*: all output data grids are set, by default, to the full (physical) model grid

   ```
   xlims = (/1,nc-1,1/); ylims = (/1,nr-1,1/); zlims = (/1,nz,1/)
   ```

3. The "namelist" utility has become redundant by implementation of the CIF, and has been removed from the code.

- The parameter MaxNMLTypes is removed in *syspars.f90*.
- The variable nmlfiles and the key ids nml_* are removed from iopars.
- Routines read_mod_params and write_mod_params are deleted in *Model_parameters.f90*.
- Routine reset_nml_params is removed in *reset_model.F90*.
- Settings of default attributes for nmlfiles are deleted in default_init_params.
- The read and write calls for namelist files are removed in initialise_model.
- Namelist definitions are removed in modules gridpars, iopars, paralpars, physpars, switches, timepars, turbpars.

4. The definition of the parameter gacc_ref in the previous versions caused a conflict with the CIF utility. A new parameter gacc_mean has been defined which has the same meaning as gacc_ref as in the previous versions

   - If gacc_ref is defined, gacc_mean=gacc_ref.
   - If gacc_ref is undefined and the grid is Cartesian, gacc_mean is defined by the geodesic formula at the latitude given by dlat_ref.
   - If gacc_ref is undefined and the grid is spherical, gacc_mean is defined by the geodesic formula applied at each C-node point and averaged over the physical domain.

   In this way, the parameter gacc_ref can be written to the CIF while retaining its initial setup value.

5. A small discrepancy was discovered when test cases ***rhone*** and ***bohai*** are run without and with CIF. This has been removed by making a small correction in the definitions of the model grid (i.e. replacing fractions by numerical values).

6. A correction has to be made in the setup of all test cases so that the test cases can be run without and with CIF. The code line

   ```
   cold_start = .TRUE.
   ```

   is changed to

   ```
   IF (ciffiles(icif_model)%status.EQ.'W') cold_start = .TRUE.
   ```

in the usrdef_init_params routine for all test cases.

7. A bug was found in the nesting procedure for baroclinic currents. The correct the error in a robust way the shapes of vertical profiles at open boundaries has been modified so that the first dimension refers to the horizontal position and the second to the vertical dimension. This means in particular that

   - obcvel3d(nz,0:noprofs),profvel(nz,maxprofs,2:nofiles,2) becomes obcvel3d(0:noprofs,nz), profvel(maxprofs,nz,2:nofiles,2) in current_cor.
   - obcsal3d(nz,0:noprofs),profsal(nz,maxprofs,2:nofiles,2) becomes obcsal3d(0:noprofs,nz), profsal(maxprofs,nz,2:nofiles,2) in salinity_equation.
   - obctemp3d(nz,0:noprofs),proftmp(nz,maxprofs,2:nofiles,2) becomes obctemp3d(0:noprofs,nz), proftmp(maxprofs,nz,2:nofiles,2) in temperature_equation.
   - obcdata(nzdat,0:noprofs) becomes obcdata(0:noprofs,nzdat).
   - psiprofdat(nzdat,numprofs,numvars) becomes psiprofdat(numprofs,nzdat,numvars) in usrdef_profobc_data.
   - A similar switch of dimensions has been made in array arguments of routines defined in *Open_Boundary_Data_Prof.f90*.

8. The positions of the arguments representing the shape of vertical profile arrays at open boundary arrays have been switched in the calls of the following routines

   - *Open_Boundary_Conditions.f90*: open_boundary_conditions_3d, open_boundary_conds_prof
   - *Open_Boundary_Data_Prof.f90*: define_profobc_data, read_profobc_data, write _profobc_data
   - *Usrdef_Model.f90*: usrdef_profobc_data

   For example

```
SUBROUTINE usrdef_profobc_data(iddesc,ifil,ciodatetime,psiprofdat,nzdat,&
                               & numprofs,numvars)
```

   becomes

```
SUBROUTINE usrdef_profobc_data(iddesc,ifil,ciodatetime,psiprofdat,&
                               & numprofs,nzdat,numvars)
```

Compatibility with previous versions

To make an application, set up under version V2.1.0, compatible with V2.1.1 the following needs to be taken into account

- The *namelist* utility has been removed (and replaced by the CIF utility).
- The shape of the array psiprofdat and the positions of the arguments nzdat, numprofs have been modified in usrdef_profobc_data (see above).

Version V2.1.2

Coherens Version	: V2.1.2
previous release	: V2.1.1
Revision	: 237
svn path	: http://svn.mumm.ac.be/svn/root/coherens/versions/V2.1.2
Date of release	: 2011-05-20
File (code)	: coherensV212_r237.tar.gz
File (manual)	: in preparation

Implementations

user output

Data values for user output can be defined automatically by the program if the user defines non-zero values for the variable key ids either in the CIF or in the corresponding usrdef_*_params routines. New variable attributes are available which can be set by the user and allow to apply an operator on the output variable(s).

- If the key id of an output variable is set with a non-zero value, all metadata and output values are automatically generated by the program. Otherwise, both metadata and output values need to be defined by the user.

- If all key ids are positive, the following routines are no longer called

  `usrdef_tsr0d_vals, usrdef_tsr2d_vals, usrdef_tsr3d_vals, usrdef_avr0d_vals, usrdef_avr2d_vals, usrdef_avr3d_vals, usrdef_anal0d_vals, usrdef_anal2d_vals, usrdef_anal3d_vals`

  In the mixed case, i.e. if not all variables (with attributes stored in tsrvars, avrvars or alnalvars) are defined with a non-zero key id, the

values of the ones with zero id need to be defined in the appropriate usrdef_*_vals routines while the other ones are defined by the program itself.

- A series of additional variable attributes have been implemented to allow for different types of output

  - The operator attribute oopt. If the rank of the result is different from the one implemented by the variable's rank, the rank attribute must be set to the rank of the result. For example, the domain average of a 3-D variable has a rank of 0. The attribute has one of the following values

    oopt_null No operator is applied (default).

    oopt_mean Result depends on the rank of the model variable and the rank of the output data.

    - If the rank of the result is 0, the output value is the domain average in case of a 3-D or the surface average in case of a 2-D variable. Land areas are excluded in the averaging.
    - If the rank of the output value is 2, the result is the depth averaged value.

    oopt_max Result depends on the rank of the model variable and the rank of the output data.

    - If the rank of the result is 0, the output value is the domain maximum in case of a 3-D or the surface maximum in case of a 2-D variable. Land areas are excluded.
    - If the rank of the output value is 2, the result is the maximum over the water depth.

    oopt_min Result depends on the rank of the model variable and the rank of the output data.

    - If the rank of the result is 0, the output value is the domain minimum in case of a 3-D or the surface minimum in case of a 2-D variable. Land areas are excluded.
    - If the rank of the output value is 2, the result is the minimum over the water depth.

oopt_klev Produces the value of a 3-D variable at the vertical level given by the attribute klev. Rank of the result is 2.

oopt_dep Produces the value of a 3-D variable at a specified depth given by the attribute dep. Rank of the result is 2.

- The attribute klev defines the output vertical level in case oopt equals oopt_klev.
- The attribute dep defines the output water depth (measured positively from the surface) in case oopt equals oopt_dep. Result is 0, if dep is larger than the total water depth at the output location.
- The node attribute is only used for 3-D variables defined at W-nodes on the model grid. If node is set to 'C' (default), the vertical profile of the variable is first interpolated at the C-node before the operator is applied. It is remarked that quantities defined at U- or V-nodes are always interpolated at C-nodes first before applying the operator.

open boundary conditions

multi-variable arrays

In view of the implementation of biological and sediment models foreseen in future version of COHERENS, the setup of open boundary conditions and the input of open boundary data for 3-D scalars has been adapted so that open boundary conditions can be defined and open boundary data can be obtained for multi-variable arrays (such as biological state variables or different sediment fractions). Since the current version only contains a physical component and the open boundary conditions and data are defined separately for each variable (baroclonic current, temperature, salinity), the procedures have not fully tested. Instructions for users of the current version and compatibility with previous versions are discussed below.

baroclinic mode

Two additional open boundary conditions for the baroclinic current have been implemented

1. Second order gradient condition. In case of ragged open boundaries the (first order) zero gradient condition may yield spurious discontinuities

of the vertical current at the first interior node. The effect is reduced when using the second order condition

$$\frac{1}{h_1}\frac{\partial}{\partial \xi_1}\Big[\frac{1}{h_1 h_2}\frac{\partial}{\partial \xi_1}(h_2 h_3 \delta u)\Big] = 0\,, \quad \frac{1}{h_2}\frac{\partial}{\partial \xi_2}\Big[\frac{1}{h_1 h_2}\frac{\partial}{\partial \xi_2}(h_1 h_3 \delta v)\Big] = 0 \quad (1)$$

at respectively U- and V-node open boundaries. The discretised version of 1 at U-nodes becomes

$$\begin{aligned} \delta u_i &= \frac{h^u_{2;i+1:i-1}}{h^u_{2;i}}\frac{h^u_{3;i+1:i-1}}{h^u_{3;i}}\Big(1 + \frac{h^c_{1;i:i-1}}{h^c_{1;i+1:i-2}}\frac{h^c_{2;i:i-1}}{h^c_{2;i+1:i-2}}\Big)\delta u_{i+1:i-1} \\ &- \frac{h^u_{2;i+2:i-2}}{h^u_{2;i}}\frac{h^u_{3;i+2:i-2}}{h^u_{3;i}}\frac{h^c_{1;i:i-1}}{h^c_{1;i+1:i-2}}\frac{h^c_{2;i:i-1}}{h^c_{2;i+1:i-2}}\delta u_{i+2:i-2} \end{aligned} \quad (2)$$

at U-nodes. A similar expression applies at V-nodes.

2. A local solution for the baroclinic current is obtained by solving the equation, obtained from the 3-D and 2-D momentum equations without advection and horizontal diffusion.

$$\frac{\partial \delta u}{\partial t} - 2\Omega \sin\phi \delta v = F^b_1 - \frac{\overline{F^b_1}}{H} + \frac{1}{h_3}\frac{\partial}{\partial s}\Big(\frac{\nu_T}{h_3}\frac{\partial \delta u}{\partial s}\Big) + \frac{\tau_{b1} - \tau_{s1}}{H} \quad (3)$$

at U-nodes and

$$\frac{\partial \delta v}{\partial t} + 2\Omega \sin\phi \delta u = F^b_2 - \frac{\overline{F^b_2}}{H} + \frac{1}{h_3}\frac{\partial}{\partial s}\Big(\frac{\nu_T}{h_3}\frac{\partial \delta v}{\partial s}\Big) + \frac{\tau_{b2} - \tau_{s2}}{H} \quad (4)$$

at V-node open boundaries. At the surface and the bottom the diffusive fluxes are set to zero.

2-D open boundary conditions

A relaxation condition can (optionally) be applied for all exterior 2-D data (transports and elevation) in case the model is set up with the default initial conditions (zero transports and elevations). In that case the exterior data function $\psi^e(\xi_1, \xi_2, t)$ is multiplied by the factor $\alpha_r(t) = \min(t/T_r, 1)$, where T_r is the relaxation period. The method avoids the development of discontinuities during the initial propagation of (e.g.) a tidal wave into the domain.

Instructions for users

user output

Automatic generation of user output data is selected by defining an output variable with a zero key id attribute ivarid and, eventually, defining the additional variable attributes oopt, klev, dep. The only other attribute which must be defined always, is the output dimension of the variable nrank. In case of a user-defined variable with a zero ivarid, all attributes must be supplied by the user. Its value must then be defined in the approriate usrdef_*_vals routine with the appropriate index in the output data vector. For example, if X is a user-defined 2-D variable and defined as the second 2-D variable in tsrvars, its output value is defined in usrdef_tsr2d_vals using

```
out2ddat(2) = X(i,j)
```

open boundary conditions

The routine usrdef_profobc_spec is called with different arguments

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

The arguments have the same meaning as before except that

- Since novars (number of variables in case of multi-variable model arrays) equals 1 in the current implementation, the new second dimension for iprofobu and iprofobv can be omitted.
- The previous argument novarsd has been removed.
- Argument iobctype is replaced by indexvar and should not be defined in the current implementation.

- The type of open boundary condition is selected at each open boundary point by the value of itypobu or itypobv. When the routine is called for baroclinic currents (iddesc=io_3uvobc) their meaning is modified as follows

  0: (First order) zero gradient condition or specified profile

  1: Second order zero gradient condition

  2: Local solution

  3: Radiation condition using the baroclinic internal wave speed

  4: Orlanski condition

A more detailed explanation will be presented in future release notes.

Routine is usrdef_profobc_data is now declared as follows

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

The following changes have been made

- The arguments nzdat and numvars have been removed.
- The third dimension of psiprofdat has been removed.
- The code in the routine must contain the statement

  ```
  USE gridpars
  ```

The relaxation condition at open boundaries for the 2-D mode is defined by the new parameter ntobcrlx which equals T_r/Δ_{2D}. Default is zero in which case no relaxation is applied. The parameter is defined either in usrdef_mod_params or in the CIF.

Test cases

No changes are made to the definition of the test cases. The output test parameters are mostly the same as in the previous version. Most output data are automatically generated so that most usrdef_*_vals routines become empty.

Model code

user output

The generation of 0-D time series output is illustrated by the following code lines in routine time_series

```
IF (tsr0d(iset)%defined) THEN
   outvars(1:noutvars0d) = tsrvars(ivarstsr0d(iset,1:noutvars0d))
   CALL define_out0d_vals(out1dsub,noutvars0d,&
                          & outvars=outvars(1:noutvars0d))
   IF (ANY(outvars(1:noutvars0d)%ivarid.EQ.0)) THEN
      CALL usrdef_tsr0d_vals(outdat(1:novars0d),novars0d)
      WHERE (outvars(1:noutvars0d)%ivarid.EQ.0)
         out1dsub = outdat(ivarstsr0d(iset,1:noutvars0d))
      END WHERE
   ENDIF
ENDIF
```

The routine define_out0d_vals provides the output data for the variables with a specified variable key id. The routine usrdef_tsr0d_vals has the same meaning as in the previous versions. A similar procedure is taken for 2-D and 3-D output and for time averaging and harmonic analysis.

The define_ routines are declared as follows

```
SUBROUTINE define_out0d_vals(outdat,novars,outvars,ivarid,oopt)
INTEGER, INTENT(IN) :: novars
REAL, INTENT(OUT), DIMENSION(novars) :: outdat
INTEGER, INTENT(IN), OPTIONAL, DIMENSION(novars) :: ivarid, oopt
TYPE (VariableAtts), INTENT(IN), OPTIONAL, DIMENSION(novars) :: outvars
```

which defines 2-D output data. The arguments have the following meaning

outdat Returned output values

novars Number of (0-D) output data

outvars Attributes of the output variables. Must be present only when ivarid is not present.

ivarid Key ids of the output variables. Must be present only if outvars is not present.

oopt Operator attributes selecting the type of output if outvars is not present. Default is oopt_null.

```
SUBROUTINE define_out2d_vals(outdat,i,j,novars,outvars,ivarid,oopt,&
                           & klev,dep,node)
INTEGER, INTENT(IN) :: i, j, novars
REAL, INTENT(OUT), DIMENSION(novars) :: outdat
CHARACTER (LEN=lennode), OPTIONAL, DIMENSION(novars) :: node
INTEGER, INTENT(IN), OPTIONAL, DIMENSION(novars) :: ivarid, klev, oopt
REAL, INTENT(IN), OPTIONAL, DIMENSION(novars) :: dep
TYPE (VariableAtts), INTENT(IN), OPTIONAL, DIMENSION(novars) :: outvars
```

which defines 2-D output data. The arguments have the following meaning

outdat Returned output values

i (Local) grid index in the X-direction

j (Local) grid index in the Y-direction

novars Number of (2-D) output data

outvars Attributes of the output variables. Must be present only when ivarid is not present.

ivarid Key ids of the output variables. Must be present only if outvars is not present.

oopt Operator attributes selecting the type of output if outvars is not present. Default is oopt_null.

klev klev attributes if outvars is not present and the corresponding oopt value equals oopt_klev

dep dep attributes if outvars is not present and the corresponding oopt value equals oopt_dep

node node attributes if outvars is not present. Should be defined only for 3-D variables located at W-nodes on the model grid in which case output is taken at the node given by cnode. Allowed values are 'C' (default) and 'W'.

```
SUBROUTINE define_out3d_vals(outdat,i,j,k,novars,outvars,ivarid,node)
INTEGER, INTENT(IN) :: i, j, k, novars
REAL, INTENT(OUT), DIMENSION(novars) :: outdat
CHARACTER (LEN=lennode), OPTIONAL, DIMENSION(novars) :: node
INTEGER, INTENT(IN), OPTIONAL, DIMENSION(novars) :: ivarid
TYPE (VariableAtts), INTENT(IN), OPTIONAL, DIMENSION(novars) :: outvars
```

which defines 3-D output data. The arguments have the following meaning

outdat	Returned output values
i	(Local) grid index in the X-direction
j	(Local) grid index in the Y-direction
k	Vertical grid index
novars	Number of (3-D) output data
outvars	Attributes of the output variables. Must be present only when ivarid is not present.
ivarid	Key ids of the output variables. Must be present only if outvars is not present.
node	node attributes if outvars is not present. Should be defined only for 3-D variables located at W-nodes on the model grid in which case output is taken at the node given by cnode. Allowed values are 'C' (default) and 'W'.

other

- The interfaces of the routines in *Open_Boundary_Data_Prof.f90* have been modified. Details are found in the model code.
- The new (and already existing) algorithms for the baroclinic current are defined in open_boundary_conds_3d (and removed from current_corr).
- The relaxation condition for 2-D open boundary data are applied in update_2dobc_data.
- The new attributes for output variables are added to the CIF.
- The following routines have been added
  - intpol1d_model_to_dep in *grid.interp.F90*
  - tvd_limiter_0d in *utility_routines.F90*

bug corrections

- In the previous versions an error may occur when non-uniform averaging is applied for model grid arrays if either iopt_arrint_hreg or iopt_arrint_vreg are set to 1. The correction has been made in routine grid_spacings by defining all horizontal grid spacing arrays over a virtual extended computational domain.
- A correction is made in the calculation of the Richardson number in *turbulence_routines.F90*.

Compatibility with previous versions

Version V2.1.2 is compatible with tV2.1.1 except that

- Additional parameters appear in the CIF (new attributes of user output variables).
- The arrays itypobu and itypobv have a slightly different meaning if applied for defining the open boundary conditions for the baroclinic mode.

Version V2.2

Coherens Version	: V2.2
previous release	: V2.1.2
Revision	: 367
svn path	: http://svn.mumm.ac.be/svn/root/coherens/versions/V2.2
Date of release	: 2011-10-14
File (code)	: coherensV2.2.tar.gz
File (manual)	: manualV2.1.2.pdf

Implementations

3-D masks

This version has primarily been created to anticipate the implementation of structures at velocity nodes (e.g. thins dams, groines, current deflection walls, ...), foreseen in a future version. For this reason, the pointer arrays nodeatu and nodeatv have been re-defined with an extra vertical dimension. In this way, a dynamic 3-D mask can be implemented in the future to simulate (e.g.) the flow over a thin dam, which extends above/below the water surface at low/high tide.

momentum fluxes at corner points

In analogy with the pointer arrays at velocity nodes, a new 3-D array nodeatuv has been created at corner nodes (replacing the pointer arrays nodeate, nodeatx, nodeaty in the previous versions). These nodes are used in the program for the evaluation of the cross-stream advective and diffusive fluxes for horizontal momentum and are located at the intersection of two U- and two V-interfaces. Main difference, with the previous version(s), is that a corner node is declared as wet if at least one of the adjacent U-nodes and one of

the adjacent V-nodes is wet (see below). The new definition is of importance near ragged coastal and open sea boundaries.

The concept of X- and Y-nodes is no longer retained, except at open boundaries (see below).

open boundaries at corner nodes

The previous definitions are changed as follows:

- A corner node is defined as a X-node open boundary if both neighbouring U-nodes are open boundaries or one of the neighbouring U-nodes is an open boundary and the other a land boundary.
- A corner node is defined as a Y-node open boundary if both neighbouring V-nodes are open boundaries or one of the neighbouring V-nodes is an open boundary and the other a land boundary.

open boundary conditions at corner nodes

In several applications of COHERENS, especially those using ragged open boundaries, instabilities were observed near the open boundaries. These took the form of growing vortices, eventually leading to a crash of the program. Using the notations in the User documentation (see Chapter Numerical Methods) the following changes and new implementations have been made.

- Two schemes are now available to evaluate the cross-stream advective fluxes in the u-equation at Y-open boundary nodes (analogous expressions apply for the v-equation and 2-D momentum equations):
  1. The first one uses a zero gradient condition
  $$F^{uv}_{12;ij} = F^{uv}_{12;i,j+1:j-1} \tag{5}$$
  which is the same as before.
  2. The flux is determined using the upwind scheme (where possible). This means that
  $$F^{uv}_{12;ij} = \frac{1}{2} v^{uv}_{ij} \Big((1 + s_{ij}) u_{i,j-1:j} + (1 - s_{ij}) u_{i,j:j-1} \Big) \tag{6}$$
  where $s_{ij} = 1$ in case of an inflow condition and either
     - (i,j-1:j) is a U-open boundary
     - (i-1,j) is a closed (land or coastal) V-node

– (i,j) is a closed V-node.

In all other cases, $s_{ij} = -1$.

- The cross-stream diffusive fluxes in the u-equation are avaluated as follows (analogous expressions apply for the v-equation and 2-D mode equations)
  - If either (i,j-1:j) is a U-open boundary, or (i-1,j) is a closed (land or coastal) V-node, or (i,j) is a closed V-node, the flux is calculated in the same way as for an internal node.
  - Otherwise, if (i,j-1:j) is an interior U-node, then the zero gradient condition $D^{uv}_{12;ij} = D^{uv}_{12;i,j+1:j-1}$ is applied.
  - Otherwise, the flux is set to zero, i.e. $D^{uv}_{12;ij} = 0$

relaxation condition for advection

An optional relaxation scheme has been implemented which reduces the impact of advection within a user-defined distance from the open boundaries. In that case, the advective terms are multiplied by the relaxation factor

$$\alpha_{or} = \min(d/d_{max}, 1) \tag{7}$$

where d is the distance to the nearest open boundary. Experiments showed that, with an appropriate choice of the maximum relaxation distance d_{max}, the unstable vortex motions no longer propagate into the domain.

The scheme replaces the previous scheme, selected by iopt_obc_int which has been removed from the code.

interpolation routines

The routines in *array_interp.f90* for interpolating a model array from one grid node to another one have been modified to take account of 3-D masks and coastal boundaries at velocity nodes. This is further discussed below.

Model code

pointer arrays

The following changes have been made

- The pointer arrays nodeatu and nodeatv have an extra vertical dimension .

- The arrays nodeatx, nodeaty, nodeate have been removed.
- New 3-D pointer arrays nodeatuw, nodeatvw and nodeatuv are introduced.

The pointer array nodeatc at C-nodes has the same meaning as before.

The arrays are declared with the following shapes

```
REAL, DIMENSION(1-nhalo:ncloc+nhalo,1-nhalo:nrloc+nhalo,nz) :: &
                                              & nodeatu, nodeatv, nodeatuv
REAL, DIMENSION(1-nhalo:ncloc+nhalo,1-nhalo:nrloc+nhalo,nz+1) :: &
                                              & nodeatuw, nodeatvw
```

and have the following meaning

nodeatu Pointer at U-node cell faces

- 0: dry (land) cell face
- 1: coastal or structure velocity boundary
- 2: interior wet U-node
- 3: open sea boundary
- 4: river open boundary

nodeatv Pointer at V-node cell faces

- 0: dry (land) cell face
- 1: coastal or structure velocity boundary
- 2: interior wet V-node
- 3: open sea boundary
- 4: river open boundary

nodeatuw Pointer at UW-node cell faces

- 0: dry (land) cell face or bottom cell (1) or surface cell (nz+1)
- 1: coastal or structure velocity boundary
- 2: interior wet UW-node
- 3: open sea boundary
- 4: river open boundary

nodeatvw Pointer at VW-node cell faces

- 0: dry (land) cell face or bottom cell (1) or surface cell (nz+1)

1: coastal or structure velocity boundary

2: interior wet VW-node

3: open sea boundary

4: river open boundary

nodeatuv Pointer at corner nodes

0: at least two surrrounding U-nodes or at least two surrrounding V-nodes are dry

1: interior wet node, i.e. at most one surrounding U-node and at most one surrounding V-node is dry and none of the four surrounding velocity nodes are open boundaries

2: X-node open boundary, in which case at least one of the surrounding U-nodes is an open boundary while the other one is either a closed node or open boundary, but the node is not a Y-node open boundary

3: Y-node open boundary, in which case at least one of the surrounding V-nodes is an open boundary while the other one is either a closed node or open boundary, but the node is not an X-node open boundary

4: the node is both a X- and a Y-node open boundary

Important to note that structures can only be defined at interior sea nodes which excludes coastal and open boundaries. This means that if nodeatu, nodeatv, nodeatuv have different values along the vertical, these values must be either 1 or 2 for the first two arrays and 0 or 1 for the third one. The same applies for nodeatuw, nodeatvw, with exception of the surface (nz+1) and bottom (1) level where the value is always 0.

open boundaries

1. The parameters nobx, noby and the logical open boundary arrays westobx, soutoby have the same meaning as before, taking account that a X-open boundary is defined at a corner node where nodeatuv equals 2 or 4, and a Y-open boundary where nodeatuv equals 3 or 4.

2. The type of open boundary condition for cross-stream advective (2-D and 3-D) fluxes at corner nodes is selected by the new switch iopt_obc_advflux

   1: zero gradient condition (5.286) which is the default

2: quasi upwind scheme (5.287)

3. The switch iopt_obc_int and (consequently) the arrays itypintobu, itypintobv have been removed.

4. The relaxation condition for advection near open boundaries is selected by the new switch iopt_obc_relax:

   0: relaxation scheme disabled (default)

   1: relaxation scheme enabled. In that case the parameter distrlx_obc (representing the parameter d_{max}) must be defined by the user in usrdef_mod_params or in the CIF.

interpolation routines

The grid interpolation routines, defined in *array_interp.f90*, have been rewritten. The name and meaning of each routine is the same before. There are however some important changes:

- The argument intsrce selects which points at the source node are taken into account. The meaning depends on the type of node and is documented internally in the source code and externally in the Reference Manual.

- The argument intdest selects at which points on the destination node interpolation is performed. The meaning depends on the type of node and is documented internally in the source code and externally in the Reference Manual.

- Vertical interpolation is (obviously) not allowed on land. This means, in particular, that W-, UW- and VW-nodes are excluded as source or destination nodes on land.

- A clear distinction has now been made between velocity nodes at coastal boundaries and at closed land cell faces.

- The arguments lbounds and ubounds are respectively the lower and upper boundaries of the interpolating array at the source node. Contrary to the previous versions, the interpolating array is assumed to be 3-D so that the two vectors must have a size of 3. Interpolation of a 2-D (horizontal) array is performed by taking the same value for lbounds(3) and ubounds(3).

structures

When structures will be implemented in the program, the bottom/surface boundary conditions are not always applied at the bottom/surface level itself but at a level k greater than 1 or lower nz. The following parameters are defined

- The switch iopt_structs disables/enables the use of structures in the model domain. If enabled (1), the bottom/surface conditions (and formulation of the bottom stress) are applied at vertical levels determined by masksuratu, masksuratv or maskbotatu, maskbotatv (defined below). The switch is currently disabled (0).

- The bottom and surface levels are defined by the following arrays

  masksuratu 3-D array at the U-nodes, which is set to .TRUE. at the vertical level where the surface boundary condition is applied

  masksuratv 3-D array at the V-nodes, which is set to .TRUE. at the vertical level where the surface boundary condition is applied

  maskbotatu 3-D array at the U-nodes, which is set to .TRUE. at the vertical level where the bottom boundary condition is applied

  maskbotatv 3-D array at the V-nodes, which is set to .TRUE. at the vertical level where the bottom boundary condition is applied

source code

The source code has been changed at many places, due to the introduction of 3-D masks at velocity nodes. This applies in particular for the advection (*Advection_Terms.F90*) and diffusion routines (*Diffusion_Terms.F90*) and for routines performing array interpolation on the model grid (*array_interp.f90*).

- The vertical k-loops have been replaced at several places by WHERE statements using 3-D masks.

- The interpolation routines have been modified to take account of velocity nodes with a non-uniform dry/wet status along the vertical.

- Application of the bottom and surface boundary conditions at velocity nodes in case iopt_structs=1.

As a consequence of these changes, the CPU time increases by about 10–20%. This has been confirmed by comparing the simulation times of the test cases with those performed using version V2.1.2. The problem will be more fully investigated at a later stage.

Test cases

The setup of the test cases has not been changed[3]. Since the code has been modified at several places, the output parameters slightly differ from the ones obtained with V2.1.2.

Compatibility with previous versions

Version V2.2 is compatible with V2.1.2 except that the switch iopt_obc_int and the open boundary setup arrays itypintobu, itypintobv have been removed.

[3]Sole exception is that the switch iopt_obc_int has been removed and iopt_obc_advflux is set to 2 in test case ***optos_csm***.

Version V2.3

Coherens Version	: V2.3
previous release	: V2.2
Revision	: 438
svn path	: http://svn.mumm.ac.be/svn/root/coherens/versions/V2.3
Date of release	: 2012-03-12
File (code)	: coherensV2.3.tar.gz
File (manual)	: manualV2.3.pdf

Implementations

inundation schemes

The existing drying/wetting scheme has been extended by implementing so-called "mask functions". They are defined as criteria for "masking" grid cells according to their condition (dry or wet). When the criterion evaluates as .TRUE. at a particular grid cell, the mask function will "mask in" the cell. Hence, they will be considered for the solution of the hydrodynamic equations. On the other hand, if grid cells become dry, the mask criterion will "mask out" such grid cells and updates of quantitites defined at these cells will be suspended. The process is repeated at the start of each predictor time step.

Eleven mask functions are defined and can be used in combined form. They can be divided in four groups. The first group compares the water depths of a cell and its neighbours with a threshold value d_{th} and is composed of the following six criteria:

$$\max(H_{i,j}, H_{i-1,j}, H_{i+1,j}, H_{i,j-1}, H_{i,j+1}) < d_{th} \quad (8)$$

$$\min(H_{i,j}, H_{i-1,j}, H_{i+1,j}, H_{i,j-1}, H_{i,j+1}) < d_{th} \quad (9)$$

$$\begin{aligned}
\text{mean}(H_{i,j}, H_{i-1,j}, H_{i+1,j}, H_{i,j-1}, H_{i,j+1}) &< d_{th} && (10)\\
\max(H_{i-1,j}, H_{i+1,j}, H_{i,j-1}, H_{i,j+1}) &< d_{th} && (11)\\
\min(H_{i-1,j}, H_{i+1,j}, H_{i,j-1}, H_{i,j+1}) &< d_{thd} && (12)\\
\text{mean}(H_{i-1,j}, H_{i+1,j}, H_{i,j-1}, H_{i,j+1}) &< d_{td} && (13)
\end{aligned}$$

where "mean" denotes an averaged value (excluding land cells which are permanently dry).

A second group of criteria verifies the "status" of the current grid cell and/or its neighbours. The status is defined by the function $\mathcal{N}$ which evaluates to 0 at dry and 1 at sea cells. The following criteria, used to prevent the formation of isolated dry or wet cells, have been implemented:

$$\begin{aligned}
\max(\mathcal{N}_{i-1,j}, \mathcal{N}_{i+1,j}, \mathcal{N}_{i,j-1}, \mathcal{N}_{i,j+1}) &= 0 && (14)\\
\min(\mathcal{N}_{i-1,j}, \mathcal{N}_{i+1,j}, \mathcal{N}_{i,j-1}, \mathcal{N}_{i,j+1}) &= 0 && (15)
\end{aligned}$$

The third group is a variant of the previous one and checks, in addition, whether the total water depth of the grid cell is lower than the threshold value:

$$\begin{aligned}
\max(\mathcal{N}_{i-1,j}, \mathcal{N}_{i+1,j}, \mathcal{N}_{i,j-1}, \mathcal{N}_{i,j+1}) = 0 \quad \text{and} \quad H_{i,j} < d_{th} && (16)\\
\min(\mathcal{N}_{i-1,j}, \mathcal{N}_{i+1,j}, \mathcal{N}_{i,j-1}, \mathcal{N}_{i,j+1}) = 0 \quad \text{and} \quad H_{i,j} < d_{th} && (17)
\end{aligned}$$

The last scheme is intended for channel flows and overflowing dykes. The criterion uses the total and mean water depths at the grid cell and its neighbours

$$\min(h_{i-1} - H_{i-1}, h_{i+1} - H_{i+1}) > h_i \tag{18}$$

If one (or more) of the above criteria evaluates as .TRUE., the grid cell is (temporarily) set to dry. In that case the surrounding velocity nodes are blocked and the currents set to zero. The same criteria are verified at the next 3-D time step. The cell will be come wet again as soon as the (combined) criterion evaluates to .FALSE.

The above criteria can be in applied in combination. This means that, if several criteria have been activated by the user, the cell becomes dry if at least one of them turns .TRUE. The cell becomes wet again if all of them evaluate to .FALSE.

In analogy with the existing algorithm in COHERENS a number of terms in the momentum equations are multiplied by the "drying" factor α and a minimum water depth is applied at each (2-D) time step. In version V2.3, these minima are determined as follows:

$$H^c_{ij} \geq H_{min}\,, \quad \zeta_{ij} \geq H_{min} - h_{ij} \tag{19}$$

at C-nodes,

$$H^u_{ij} = \min(H^c_{i-1,j}, H^c_{ij}) \quad \text{if} \quad \min(H^c_{i-1,j}, H^c_{ij}) < H_{crit} \tag{20}$$

at U-nodes, and

$$H^v_{ij} = \min(H^c_{i,j-1}, H^c_{ij}) \quad \text{if} \quad \min(H^c_{i,j-1}, H^c_{ij}) < H_{crit} \tag{21}$$

at V-nodes.

other

1. By default, user output files are written as netCDF ('N') or unformatted binary ('U') files depending on whether -DCDF compiler option has been specified or not. These defaults can be reset by the user in the usrdef_tsr_params, ... routines in the appropriate *Usrdef_* files.

2. The status of open boundary forcing files is (re)set to 'N' (undefined) if the appropriate switch is not set. For example, modfiles(io_2uvobc,:,:)%status is set to '0' is iopt_obc_2D=0.

Model code

The following files have been created or modified

Inundation_Schemes.f90

The following routines, only called if iopt_fld is non-zero, are defined here

mask_function — The routine is called by coherens_main at the start of the predictor time step and

- evaluates one or more mask criteria
- set a cell to dry (at the C-node) if the criteria return .TRUE.
- block the surrounding U- and V-velocity nodes
- set the currents to zero at blocked velocity nodes

minimum_depths — Use (19)–(21) to set the total water depths and surface elevations to their minimum values where necessary. The routine is called from water_depths.

drying_factor — Evaluates the drying factor α at each 2-D time step. The routines is called from surface_elevation.

Grid_Arrays.F90

A new routine store_depths_old for storing the old water depths has been created. The routine is called before mask_function from coherens_main at the start of the predictor step.

depths.f90

When the total water depth is reset to its minimum value, artificial water is added to the water column. This means that mass conservation has been violated. The program stores this "depth deficit" at each time step into the depth error array deptotatc_err

```
WHERE (depmeanatc(1:ncloc,1:nrloc).GT.0.0.AND.&
     & deptotatc(1:ncloc,1:nrloc).LT.dmin_fld)
   deptotatc_err = deptotatc_err - deptotatc(1:ncloc,1:nrloc) + &
                 & dmin_fld
END WHERE
```

Note that the array is always positive and can only increase in time.

gridpars.f90

The parameters nowetatc, nowetatcloc, nowetu, nowetuloc, nowetv, nowetvloc, nowetuv, nowetuvloc have been removed. The following parameters are added

noseaatc	Number of sea (dry or wet, but excluding permanent land points) C-node points on the global domain
noseaatcloc	Number of sea (dry or wet, but excluding permanent land points) C-node points on the local domain
nowetatc	Number of currently active (wet) C-node points on the global domain
nowetatcloc	Number of currently active (wet) C-node points on the local domain

paralpars.f90

The arrays nowetcprocs, nowetuprocs, nowetvprocs, nowetuvprocs have been removed.

physpars.f90

The following parameters have been added

dthd_fld	User-defined treshold depth d_{th}. Default is 0.1 m
nofldmasks	Number of implemented mask functions.

fld_mask(nofldmasks) Enables (1) or disables (0) a specific mask function. Default is fld_mask(1)=1, fld_mask(2:)=0. This can be changed by the user.

switches.f90

iopt_CDF Enables/disables netCDF output (0/1). The switch is switched on automatically if the program is compiled with the -DCDF CPP option and cannot be set by the user.

iopt_fld Selects the type of drying/wetting algorithm

0: Drying/wetting disabled

1: Without using the dynamic mask function

2: Dynamic mask function enabled

inout_routines.f90

A few bugs (typing errors) have been corrected.

paral_utilities.f90

The sum2_vars generic routine has an additional (optional) argument

```
LOGICAL, INTENT(IN), OPTIONAL, DIMENSION(:,:[,:]) :: mask
```

If present, grid points where mask is .FALSE. are excluded in calculating the sum. Otherwise, the mask is defined internally. The principal reason for implementing the new argument is that cells which are temporarily set to dry, are excluded if the argument is not present. They can be put in again by providing the mask as an argument, e.g. mask=depmeanatc(1:ncloc,1:nrloc).GT.0.0.

reset_model.F90

The status attribute of the open boundary forcing files is set to '0' if the corresponding switch is zero.

Instructions for users

The procedures for setting up an application with the drying/wetting algorithms are as follows

usrdef_mod_params

1. Enable the inundation scheme by setting iopt_fld to 1 or 2.

2. If iopt_fld=2, select the mask criteria by setting the elements of the vector fld_mask to 0 or 1. In most cases, the default scheme is sufficient. Note that the array is not used if iopt_fld=1.
3. Define the depth parameters dmin_fld, dcrit_fld, dthd_fld. Defaults are (0.02,0.1,0.1). The threshold depth dthd_fld is only used when iopt_fld=2.

All these parameters can alternatively be defined in the CIF.

usrdef_grid

If it is the intention to apply COHERENS for the simulating the flooding of (intially dry) land arrays or obstacles, the following procedure is recommended

1. Define the topographic height (e.g. h_{max}) of the highest points on land which can potentially be flooded by a rising sea level.
2. Increase the reference mean sea level by adding h_{max} to the initial bathymetry (with respect to the standard mean sea level).
3. Note that grid points with a zero mean water depth are considered by the model as permanently dry land points and cannot be flooded. Negative depth values are not allowed.

usrdef_phsics

Reset the initial surface elevation to take account of the changed reference level

$$\zeta_{in}^{new} = \zeta_{in}^{old} - h_{max} \tag{22}$$

where ζ_{in}^{old} is the sea level with respect to the standard level. If, by this procedure, the total water depth becomes negative (more precisely lower than d_{min}) the total depth will be reset to d_{min}. In case a dynamic mask is applied, these grid cells may be (temporarily) set to dry (depending on the type of mask function) at the initial time.

usrdef_2dobc_data

If residual (non-harmonic) elevation data are used as open boundary forcing, the previous change in mean sea level must be taken into account.

Test cases

Two new inundation test case have been implemented

flood2d

Flooding and drying inside a channel. The following experiments are defined

'A' Flooding/drying of land masses over a sloping bottom by an oscillating (tidal) current. No dynamic mask is used.

'B' The same as experiment 'A', now using a dynamic mask.

'C' Flooding/drying over an obstacle located in the middle of the channel by an oscillating (tidal) current. No dynamic mask is used.

'D' The same as experiment 'C', now using a dynamic mask.

flood3d

Flooding and drying inside a rectangular basin. An oscillating current enters the basin on the western side. All other sides of the basin are closed. All experiments use a mask function.

'A' Flooding/drying over a spherical hill in the middle of the basin.

'A' Flooding/drying over a double-peaked hill in the middle of the basin.

'C' Flooding/drying over a spherical atoll. The inner side of the atoll is taken as dry at the initial time.

'D' The same as experiment 'C', now in depth-averaged mode.

Test case parameters, in particular for testing mass conservation, are defined for each experiment. Further details about the setup and output parameters of these test cases are described in the User Manual.

The three ***optos*** test cases are modified as follows:

1. Harmonically analysed values of surface elevation and currents (elliptic parameters) at selected stations are defined as test case parameters.
2. The drying/wetting algorithm with dynamic mask has been activated.
3. Each ***optos*** test is run for a one month period. Disadvantage is a significantly increased computing time.

Compatibility with previous versions

The setup of applications made with Version 2.2 can be used without modification with Version 2.3.

Version V2.4

Coherens Version	: V2.4
previous release	: V2.3
Revision	: 447
svn path	: http://svn.mumm.ac.be/svn/root/coherens/versions/V2.4
Date of release	: 2012-04-03
File (code)	: coherensV2.4.tar.gz
File (manual)	: manualV2.4.pdf

Implementations

implicit algorithm

A semi-implicit algorithm has been implemented for the free surface term in the momentum equations. With this method, there is no longer need to solve the depth-integrated momentum equations (unless a 2-D grid has been selected). The stringent CFL stability criterium is relaxed by treating the terms that provoke the barotropic mode in an implicit manner. Difference with the previous explicit version is that the surface slope term is taken at the new time level. Horizontal advection and diffusion are calculated, as before, at the old time level.

After an explicit "predictor" step, velocities are corrected with the implicit free surface correction in the "corrector" step. In this method, the free surface correction follows from the inversion of the elliptic free surface correction equation obtained from the 2-D continuity equation. Because of the non-linear dependency of the equations on the free surface height through the h_3-term, an iterative scheme has been implemented in addition.

For clarity, a new superscript is introduced indicating the iteration level. As such $\varphi^{n+1,it+1}$ denotes the value of φ at the new time level $n+1$, obtained after performing iteration it. The procedure consists of the following steps

1. At the first iteration $\zeta^{n+1,1} = \zeta^n$ and $h_3^{n+1,1} = (h + \zeta^n)\Delta\sigma$.

2. Using the notations, defined in Chapter 5 of the User Manual, the momentum equations are solved at the predictor step using the latest values for h_3 and ζ:

$$\begin{aligned}\frac{h_3^{n+1,it}\tilde{u}^p - h_3^n u^n}{h_3^n \Delta t} &= fv^n - \mathcal{A}_{h1}(u^n) - \mathcal{A}_{h2}(u^n) \\ &-\frac{v^{n;u}}{h_1^u h_2^u}(u^n \Delta_y^u h_1^{uv} - v^{n;u}\Delta_x^u h_2^c) - \theta_a \mathcal{A}_v(\tilde{u}^p) - (1-\theta_a)\mathcal{A}_v(u^n) \\ &+\theta_v \mathcal{D}_{mv}(\tilde{u}^p) + (1-\theta_v)\mathcal{D}_{mv}(u^n) - g\frac{h_3^{n+1,it}}{h_3^n}\frac{\Delta_x^u \zeta^{n+1,it}}{h_1^u} \\ &-\frac{\Delta_x^u P_a}{\rho_0 h_1^u} + F_1^{b;n} + F_1^{t;n+1} + \mathcal{D}_{mh1}(\tau_{11}^n) + \mathcal{D}_{mh2}(\tau_{12}^n) \end{aligned} \tag{23}$$

$$\begin{aligned}\frac{1}{h_3^n}\frac{h_3^{n+1,it}\tilde{v}^p - h_3^n v^n}{\Delta t} &= -fu^n - \mathcal{A}_{h1}(v^n) - \mathcal{A}_{h2}(v^n) \\ &-\frac{u^{n;v}}{h_1^v h_2^v}(v^n \Delta_x^v h_2^{uv} - u^{n;v}\Delta_y^v h_1^c) - \theta_a \mathcal{A}_v(\tilde{v}^p) - (1-\theta_a)\mathcal{A}_v(v^n) \\ &+\theta_v \mathcal{D}_{mv}(\tilde{v}^p) + (1-\theta_v)\mathcal{D}_{mv}(v^n) - g\frac{h_3^{n+1,it}}{h_3^n}\frac{\Delta_y^v \zeta^{n+1,it}}{h_2^v} \\ &-\frac{\Delta_y^v P_a}{\rho_0 h_2^v} + F_2^{b;n} + F_2^{t;n+1} + \mathcal{D}_{mh1}(\tau_{21}^n) + \mathcal{D}_{mh2}(\tau_{22}^n) \end{aligned} \tag{24}$$

   where the surface slope is taken at the previous iteration level. The equations are solved as in the previous versions of COHERENS. The predicted currents (u^p, v^p) are obtained from ($\tilde{u}^p, \tilde{v}^p$) after applying an implicit correction for the Coriolis terms.

3. The free surface correction ζ' is defined as

$$\zeta' = \zeta^{n+1,it+1} - \zeta^{n+1,it} \tag{25}$$

   The corrected depth-integrated current is then obtained by adding an implicit correction term

$$U^{n+1,it+1} = U^p - H^{n+1,it;u}\frac{\Delta t g}{h_1}\frac{\partial \zeta'}{\partial \xi_1} \tag{26}$$

$$V^{n+1,it+1} = V^p - H^{n+1,it;v}\frac{\Delta t g}{h_2}\frac{\partial \zeta'}{\partial \xi_2} \tag{27}$$

where (U^p,V^p) are the depth integrated values of (u^p,v^p).

The values for ζ' follow from inversion of the elliptic equation that arises by introducing (26)–(27) into the 2-D continuity equation

$$\frac{\zeta^{n+1,it}-\zeta^n}{\Delta t}+\frac{\zeta'}{\Delta t}=-\frac{1}{h_1h_2}\left(\Delta_x^c\left(h_2^uU^p\right)+\Delta_y^c\left(h_1^vV^p\right)\right)$$
$$+\frac{1}{h_1h_2}\left[\Delta_x^c\left(\frac{\Delta t h_2^u g^u H^{n+1,it;u}}{h_1^u}\Delta_x^u\zeta'\right)+\Delta_y^c\left(\frac{\Delta t h_1^v g^v H^{n+1,it;v}}{h_2^v}\Delta_y^v\zeta'\right)\right] \tag{28}$$

Equation (5.36) can be written as a linear system of equations with non-zero values only on the diagonal and five sub-diagonals

$$A_{ij}\zeta'_{i-1,j}+B_{i,j}\zeta'_{i,j-1}+C_{ij}\zeta'_{i,j}+D_{ij}\zeta'_{i,j+1}+E_{ij}\zeta'_{i+1,j}=F_{ij} \tag{29}$$

Since the decomposition (26)–(27) can no longer be used at open boundaries, U^{n+1} or V^{n+1} are firstly written as a sum of explicit and implicit (involving ζ') terms which are then substituted into the continuity equation. Details of this procedure are given in the User Documentation.

4. The free surface elevation is updated

$$\zeta^{n+1,it+1}=\zeta^{n+1,it}+\zeta' \tag{30}$$

5. The total water depth is updated

$$H^{n+1,it+1}=H^{n+1,it}+\zeta' \tag{31}$$

6. The depth-integrated velocity fields are corrected using (26)–(27).

7. The values of $U^{n+1,it+1}$ and $V^{n+1,it+1}$ are evaluated at the open boundaries by applying the open boundary conditions.

8. The predicted values u^p, v^p of the horizontal current are corrected to ensure that the depth-integrated currents obtained from equations (26)–(27) are identical to the depth-integrated values of the 3-D current. The corrected values are then given by

$$u^{n+1}=\frac{H^{n+1,it;u}u^p+U^{n+1,it+1}-U^p}{H^{n+1,it+1;u}} \tag{32}$$

$$v^{n+1}=\frac{H^{n+1,it;v}v^p+V^{n+1,it+1}-V^p}{H^{n+1,it+1;v}} \tag{33}$$

9. A convergence check is performed by comparing the norm of ζ' with a threshold value ϵ, i.e.

$$\|\zeta'\|_\infty = \max(\zeta') \leq \epsilon_{imp} \tag{34}$$

A new iteration is started when the criterion is not satisfied.

At present, no algorithm has been programmed within the COHERENS source code to solve the linear system, arising from the discretisation of the 2-D continuity equation. Routines have, however, been provided to solve (5.37) with the external PETSc library which is activated in the program by setting the -DPETSC compiler option. Different algorithms (linear solvers and preconditioners) are available, whose default values (Incomplete Cholesky preconditioner in combination with a GMRES solver) can be changed by the user. Since the solvers are iterative, a tolerance level has to be provided.

In summary, application of the implicit scheme involves two iteration loops. The inner loop solves the linear system for ζ' and is controlled by the routines of the PETSc library. The maximum number of iterations of the outer loop (needed for convergence of the h_3-factor) is set by the user with the parameter maxitsimp.

open boundary condition

For reasons of compatibility with the implicit scheme, the open boundary condition using the characteristic method with a zero normal gradient has been rewritten without the term on the right hand side arising from the continuity condition. This means that the previous formulation at U-open boundaries

$$\frac{\partial R_i^u}{\partial t} = \mp \frac{c}{h_1 h_2}\left(\frac{\partial}{\partial \xi_2}(h_1 V) + \frac{\partial h_2}{\partial \xi_1} U\right) + fV + HF_1^t + \tau_{s1} - \tau_{b1} \tag{35}$$

becomes

$$\frac{\partial R_i^u}{\partial t} = fV + HF_1^t + \tau_{s1} - \tau_{b1} \tag{36}$$

A similar change is made at V-nodes. A more appropriate implicit version of this condition will be implemented in a future model version.

Model code

routines

The following files have been created or modified:

Hydrodynamic_Equations.F90

hydrodynamic_equation	Main routines for solving the 2-D and/or 3-D momentum and continuity equations using either the explicit or implicit scheme
current_pred	Solve the 3-D momentum equations for the predicted currents using either the explicit or implicit (step 2 of the algorithm) method.
current_2d	Solve the 2-D momentum equations. In case of a 3-D grid (iopt_grid_nodim=3), the routine is called only as part of the mode splitting algorithm (explicit method). The routine is called at all time steps with the explicit and implicit scheme in case of a 2-D grid (iopt_grid_nodim).
correct_free_surf	Performs steps 3 to 7 of the implicit algorithm. The routine is not called in case of an explicit scheme.
current_corr	Applies the corrector step for both the explicit and implicit (step 8 of the algorithm) method.
surface_elevation	The same as before, but the routine is not called by the implicit scheme.

Open_Boundary_Conditions.f90

open_boundary_conds_impl	Insert the terms arising from the open boundary conditions at the appropriate places in the matrix system (5.37).

petsc_routines.f90

Series of routines for solving the system of linear equations using the PETSc library. The routines are called only if the compiler option -DPETSC has been defined.

Transport_Equations.F90

The transport equations for currents have been (slightly) modified so that they can be used both with the explicit and implicit method. The changes are purely technical and not documented.

arrays

The following new arrays are defined for internal purposes only:

currents.f90

umvel_old	Depth-mean current u at the old time level t^n
vmvel_old	Depth-mean current v at the old time level t^n

depths.f90

deptotatu_prev	Total water depth at the U-nodes and the previous (outer) iteration
deptotatv_prev	Total water depth at the V-nodes and the previous (outer) iteration
dzeta	Difference ζ' between the surface elevation at the next and previous iteration
zeta_old	Surface elevation at the old time level t^n

obconds.f90

obc2uvatu_old	Value of obc2uvatu at the old time level t^n
obc2uvatv_old	Value of obc2uvatv at the old time level t^n

switches

The following switches have been (re)defined in *switches.f90*:

iopt_mode_2D Status of the 2-D mode. Its value is set internally and cannot be changed by the user.

- 0: The 2-D mode is disabled. Transports U, V and surface elevations ζ are set to their (zero) default values and are not updated.
- 1: Transports and elevation are initialised, but not updated in time
- 2: Transports and elevations are initialised and updated in time

iopt_mode_3D — Status of the 3-D mode. Its value is set internally and cannot be changed by the user.

- 0: The 3-D current are set to their default (zero) values and are not updated.
- 1: The 3-D current is initialised, but not updated in time.
- 2: The 3-D current is initialised and updated in time.

iopt_petsc — Enables/disables the use of the PETSc library. Its value is set internally and switched on if -DPETSC is provided as compiler option.

- 0: PETSc is switched off
- 1: PETSc is switched on

iopt_hydro_impl — Disables/enables the implicit scheme.

- 0: The momentum equations are solved with the explicit scheme (default).
- 1: The momentum equations are solved using the implicit algorithm. The compiler option -DPETSC must be set.

iopt_curr — Type of current fields.

- 0: Currents and elevations are set to their default (zero) values and are not updated.
- 1: Currents and elevations are initialised but not updated in time.
- 2: Currents are initialised but not updated in time.

iopt_petsc_solver — Type of solver used by PETSc. For details, see the PETSc User Manual.

- 1: Richardson (KSPRICHARDSON)
- 2: Chebychev (KSPCHEBYCHEV)
- 3: Conjugate Gradient (KSPCG)
- 4: Biconjugate Gradient (KSPBICG)
- 5: Generalised Minimal Residual (KSPGMRES)
- 6: BiCGSTAB (KSPBCGS)
- 7: Conjugate Gradient Squared (KSPCGS)
- 8: Transpose-Free Quasi-Minimal Residual (1) (KSPTFQMR)

9: Transpose-Free Quasi-Minimal Residual (2) (KSPTCQMR)

10: Conjugate Residual (KSPCR)

11: Least Squares Method (KSPLSQR)

12: Shell for no KSP method (KSPPREONLY)

iopt_petsc_precond Type of preconditioner used by PETSc. For details, see the PETSc User Manual.

1: Jacobi (PCJACOBI)

2: Block Jacobi (PCBJACOBI)

3: SOR (and SSOR) (PCSOR)

4: SOR with Eisenstat trick (PCEISENSTAT)

5: Incomplete Cholesky (PCICC)

6: Incomplete LU (PCILU)

7: Additive Schwarz (PCASM)

8: Linear solver (PCKSP)

9: Combination of preconditioners (PCCOMPOSITE)

10: LU (PCLU)

11: Cholesky (PCCHOLESKY)

12: No preconditioning (PCNONE)

model parameters

physpars.f90

itsimp	Current iteration number for the outer loop of the implicit scheme
noitsimp	Last iteration number of the outer loop
maxitsimp	Largest allowed iteration number for the outer loop
dzetaresid	Value of $\|\zeta'\|_\infty$. Its value is saved at the last iteration until the next time step.
dzetaresid_conv	Threshold value ϵ_{imp} used in the convergence criterium for the outer loop
petsc_tol	Relative tolerance used by PETSc for solving the linear system. (The parameters atol, dtol, maxits used by PETSc in the solution procedure are set to the PETSc defaults).

delt2d In the explicit (mode splitting) case, the time step for the 2-D mode. In case an implicit scheme is taken, the time step used for all transport equations.

Instructions for users

model setup

The following new switches and model parameters can be set by the user in usrdef_mod_params or in the CIF. Default is given in parentheses.

iopt_curr	Type of current fields (2)
iopt_hydro_impl	Selects explicit or implicit scheme (0)
iopt_petsc_solver	Type of solver used by PETSc (5)
iopt_petsc_precond	Type of preconditioner used by PETSc (5)
maxitsimp	Maximum number of iterations allowed for the outer loop (1)
dzetaresid_conv	Threshold value ϵ_{imp} (10^{-14})
petsc_tol	Relative tolerance used by PETSc (10^{-7})

compilation

The procedures have been changed so that COHERENS can be compiled with or without the PETSc library. Note that the implicit scheme can only be used in the latter case. Difference with the previous version is that the file *options.cpp* has been removed and replaced by *coherensflags.cmp*. This file is read by the *Makefile* and contains definitions of machine-dependent macros. A default (empty) version, located in the **comps** directory is listed below.

```
1 :#
2 :# Version - @COHERENScoherensflags.cmp   V2.4
3 :#
4 :# $Date: 2013-04-23 10:59:00 +0200 (Tue, 23 Apr 2013) $
5 :#
6 :# $Revision: 556 $
```

```
7 :#
8:
9 :# options for compilation with CPP
10:## -DALLOC :allocates/deallocates local arrays
11:## -DMPI   :includes MPI library
12:## -DCDF   :includes netCDF library
13:## -DVERIF :enables output for verification procedure
14:## -DPETSC : includes PETSc library
15:
16:CPPDFLAGS =
17:
18:# netCDF directory path
19:#NETCDF_PATH = /usr/local
20:
21:# netCDF library file
22:#NETCDF_LIB_FILE = netcdf
23:
24:# netCDF include options
25:#FCIFLAGS_NETCDF = -I$(NETCDF_PATH)/include
26:
27:# netCDF library options
28:#FLIBS_NETCDF = -L$(NETCDF_PATH)/lib -l$(NETCDF_LIB_FILE)
29:
30:# PETSc directories
31:#PETSC_DIR = /home/patrick/petsc/petsc-3.1-p5
32:#PETSC_ARCH = linux-gfort
33:
34:# PETSc include options
35:#CPPIFLAGS = -I$(PETSC_DIR)/include -I$(PETSC_DIR)/include/mpiuni
               -I$(PETSC_DIR)/$(PETSC_ARCH)/include
36:#FCIFLAGS_PETSC = -I$(PETSC_DIR)/include -I$(PETSC_DIR)/include/mpiuni
                    -I$(PETSC_DIR)/$(PETSC_ARCH)/include
37:
38:# environment variables for PETSc
39:# include $(PETSC_DIR)/conf/variables
40:
41:# PETSc libary options
42:#FLIBS_PETSC = $PETSC_LIB
```

The macros, which can be defined by the user, are on the following lines

- Line 16: compiler options for the CPP (previously defined in *options.cpp*).

The following options are implemented

-DALLOC Enables allocation of local arrays

-DMPI Allows the use of MPI routine calls

-DCDF Allows the use of netCDF routine calls

-DVERIF Used to run the test cases with the verification procedure

-DPETSC Allows the use of PETSc routine calls.

- Line 19: installation path of the netCDF library. The compiler then expects that the library file and the compiled netCDF modules are found in respectively the directories $NETCDF_PATH/lib and $NETCDF_PATH/include
- Line 22: name of the netCDF library file
- Line 25: compiler include options for netCDF
- Line 28: options for compilation with the netCDF library
- Line 31: directory path where the PETSc library is installed
- Line 32: directory where the PETSc installation for a specific fortran compiler is located
- Line 35: CPP include options for PETSc
- Line 36: FORTRAN include options for PETSc
- Line 39: input file with definitions of PETSc variables
- Line 42: options for compilation with the PETSc library

The following changes are to be made by the user

- If -DCDF is defined on line 16, lines 19, 22, 25 and 28 must be uncommented and changed where necessary.
- If -DPETSC is defined on line 16 then:
  - The installation path names of PETSc must be defined on lines 31–32. The meaning of PETSC_DIR and PETSC_ARCH is explained in the PETSc manual.

- Either 35 or 36 must be uncommented (without further modification), depending on the compiler. In case of a gfortran compiler, only CPPIFLAGS needs to be defined, while for an intel compiler only line 36 needs to be uncommented
- Lines 39 and 42 must be uncommented without further modification.

The procedure for setting up the model for an implicit application is as follows

- Install the PETSc library using the instructions given in the PETSc installation manual.
- Add the -DPETSC compiler option and uncomment/change the lines in *coherensflags.cmp*, as explained above
- Set the switch iopt_hydro_impl to 1 and (where needed) other parameters, listed in the section model setup, in usrdef_mod_params.

installation

Test cases and user application can be installed on a working directory with the shell script `install_test`. An updated version is now available. The script is now used with optional arguments

`install_test [-t` *`test_name`*`] [-u` *`test_dir`*`] [-o` *`flag_file`*`]`

where

-t Installs the pre-defined test case *test_name*, e.g. ***cones***.

-u Installs a user defined application. The setup *Usrdef_** and *defruns* files are copied from directory *test_dir* to the directory where `install_test` is executed.

-o Copies the file *flag_file* with the user-specific compilation instructions (see above) to the file *coherensflags.cmp* in the working directory.

- The link **COHERENS** must be defined before using the script.
- The options -t and -u are mutually exclusive.
- If neither -t or -u are present, no application has been defined, but the program can still be compiled.

- If -o is not present, the file *coherensflags.cmp* in the **comps** directory is copied by default.

The script creates the following links

SOURCE	directory path of the "main" source code
BSOURCE	directory path of the biological source code
COMPS	directory path of the files for compilation of the "main" code
BCOMPS	directory path of the files for compilation of the biological source code
SCR	directory path of the scr directory
SETUP	path of the directory where the files for the application are located
DATA	directory path of the data directory

Test cases

- Test cases ***cones, front, pycno*** and ***csnsp*** are as before.
- All other test cases are run in either explicit or implicit mode, depending on whether -DPETSC has been specified.
- In the implicit case, the name of the test case file is as before but with a "2" added after the name of the experiment. For example, *fredyA.tst* and *fredyA2.tst* are the result files for experiment ***fredyA*** and respectively the explicit and implicit case.
- The switches iopt_mode_2D and iopt_mode_3D are no longer user-defined. The switch iopt_curr is defined where necessary.
- Except for ***cones, front, pycno*** and ***csnsp***, which are the same as before, the explicit results may be (slightly) different from version V2.3.
- The zero gradient open boundary condition in test case ***rhone*** has been replaced by the Orlanski condition.

Compatibility with previous versions

Installation and compilation have been changed with respect to the previous Version 2.3. The switches iopt_mode_2D and iopt_mode_3D are no longer user-defined. The type of current field is now defined with the new switch iopt_curr (see above). All other aspects of user setup are the same as previous. Note that the results may become slightly different.

Version V2.4.1

Coherens Version	: V2.4.1
previous release	: V2.4
Revision	: 469
svn path	: http://svn.mumm.ac.be/svn/root/coherens/versions/V2.4.1
Date of release	: 2012-05-30
File (code)	: coherensV2.4.1.tar.gz
File (manual)	: manualV2.4.pdf

Implementations

A CPU performance study was conducted showing that some of the model grid interpolations in *array_interp.f90* have a better CPU performance using 2-D masks and weight functions. The type of interpolation is defined with the switch iopt_structs which has now the following purpose:

0: The 3-D mask and weighting functions are replaced by 2-D versions in some grid interpolation routines. This is the default value.

1: Grid interpolation is performed as before.

Since the structure module has not yet been implemented, it is recommended to keep the default value.

Model code

The switch iopt_structs is introduced in the following interpolation routines, defined in *array_interp.f90*:

`Carr_at_U, Carr_at_UV, Carr_at_V, Uarr_at_C, Uarr_at_UV,`
`Uarr_at_UV, Varr_at_C, Varr_at_U, Varr_at_UV`

Test cases

There are no changes.

Compatibility with previous versions

Version V2.4.1 is fully compatible with the previous one (V2.4).

Version V2.4.2

Coherens Version	: V2.4.2
previous release	: V2.4.1
Revision	: 584
svn path	: http://svn.mumm.ac.be/svn/root/coherens/versions/V2.4.2
Date of release	: 2013-04-16
Code	: coherensV2.4.2.tar.gz
User documentation	: User_Documentation_V2.4.2.pdf
Reference manual	: Reference_Manual_V2.4.2.pdf

Implementations

Compared to the previous (V2.4.1) versions there are a few minor new implementations.

Data flag for bathymetry

In the previous versions of COHERENS mean water depths must be non-negative and grid points with a zero bathymetric value are considered as permanent land points. In order to allow flooding of land areas, the suggestion was made for the user to raise the mean water level by an amount (say) h_{ref}, while decreasing the sea surface elevation initally by the same amount. In this way, the (initial) total water depth remains the same whereas grid points on land with a height above the (real) mean water level below h_{ref} may become inundated.

Disadvantage of this method is that open boundary conditions, such as tidal harmonics, need to be adapted as well, while it creates problems for performing harmonic analysis. To overcome the problem, a data flag for mean water depths has now been implemented. Default is zero, but this value can be reset by the user. When a mean water depth takes the value

of this flag, the corresponding grid points are considered as permanent land cells, otherwise the point is taken as wet or temporarily dry. Land topography is then represented by negative mean water depths. Land flooding can then be simulated without changing the reference mean water level.

Drag coefficient

Two changes are made with respect to the calculation of the bottom drag coefficient. The expression used in the code is derived from the logarithmic profile of the current in the bottom boundary layer

$$|\mathbf{u}(z)| = \frac{u_{*b}}{\kappa} \ln(\frac{z}{z_0}) \tag{37}$$

where $u_{*b}^2 = \tau_b/\rho$ and z_0 the bottom roughness length. From (37) one obtains

$$\tau_b = \rho C_{db} |\mathbf{u}_b|^2 = \rho C_{db} \left(u_b^2 + v_b^2\right) \tag{38}$$

with

$$C_{db} = \left[\frac{\kappa}{\ln(z_b/z_0)}\right]^2 \tag{39}$$

and z_b is the height of the lowest C-node above the sea bed.

1. The log-layer approximation is only valid if $z_b \gg z_0$. This may create a problem in case the grid cell is drying and $z_b \to z_0$, $C_{db} \to \infty$. To prevent too large drag coefficients, a lower limit has been imposed of the form $z_b/z_0 > \xi_{min}$. In the previous versions this limit was set internally to a value of 1.5. In the current version ξ_{min} is user-defined. Default value is 2 yielding a maximum of 0.333 for C_{db}.

2. When COHERENS is applied in depth averaged mode (iopt_grid_nodim=2), the drag law was previously applied with $z_b = H/2$ where H is the total water depth. A more realistic method, implemented in the current version is to take the depth average of (37). Provided $z_0 \ll H$, equation (39) is recovered with $z_b = H/e = 0.736H/2$.

Model code

The following new parameters are defined

physpars.f90

depmeanflag Data flag marking land points in the bathymetry in m. Default is 0.

zbtoz0lim Value of the critical ratio r for z_b/z_0. Default is 2.0.

syspars.f90

enap Euler's number $e = 2.718282$.

The parameters depmean_flag and zbtoz0lim can be defined by the user in usrdef_mod_params or through the CIF.

Test cases

The following test cases have been modified:

- The 2-D experiments ***bohaiA, bohaiB, bohaiC*** are no longer defined with a constant drag coefficient, but with a constant roughness height (iopt_bstres_drag=3) using the same value as for the 3-D experiments ***D–F***.
- All inundation experiments ***flood2d, flood3d*** are now defined with a uniform roughness height of 0.001 m, instead of a constant bottom drag coefficient. This causes a stronger flow retardation during drying phases. Test case output parameters are therefore significantly different.

Compatibility with previous versions

This version is compatible with previous versions without changes.

Version V2.5

Coherens Version	: V2.5
previous release	: V2.4.2
Revision	: 576
svn path	: http://svn.mumm.ac.be/svn/root/coherens/versions/V2.5
Date of release	: 2013-05-31
Code	: coherensV2.5.tar.gz
User documentation	: User_Documentation_V2.5.pdf
Reference manual	: Reference_Manual_V2.5.pdf

Implementations

An extended sediment transport module has been implemented. This version can therefore be considered as a major update of the code. Details are described in Chapter 7 of the User Documentation. The main features are:

- a module for the advective-diffusive transport of suspended sediments
- different fractions for the simulation of graded sediments
- near-bed boundary conditions for sand as well as mud
- various methods for bed and total load transport
- different formulae for the settling velocity (including hindered settling, flocculation), critical shear stress, ...
- turbulence damping caused by vertical stratification due to sediment concentrations
- turbidity flows caused by horizontal sediment concentration gradients.

A first version of a surface wave module has been implemented. A full current-wave interaction module is foreseen for a future COHERENS version. The aim here is to provide wave parameters used in some of the formulations for bed and total load transport.

Installation

file directories

Some changes are made to the file directory tree, as shown in Figure 1.1.

- A new directory **physics** has been created containing all files releated to the "physical core" part of COHERENS. The **/code/physics/source** and **/code/physics/comps** directories are the analogues of **/code/source**, respectively **/code/comps** in the previous releases.
- The source code and compilation files for the new sediment model are in **/code/sediments/source** and **/code/sediments/comps**.
- The **/code/scr** directory has been moved to the root directory **/coherens/V2.5**.

compilation

The following new macros are defined in *coherensflags.cmp*:

```
# physics directory path
PHYSMOD = COHERENS/code/physics

# sediment directory path
#SEDMOD = $(PHYSMOD)
SEDMOD = COHERENS/code/sediment
```

where PHYSMOD and SEDMOD are the path names of the physical and sediment main directories. The first one should not be changed. For the sediments the following options are available:

1. If SEDMOD is set to $(PHYSMOD), the code is compiled without sediments. Since the main code of COHERENS now contains "generic" calls to a number of routines used for sediments, a number of dummy routines are provided in **/physics/source**. They are there only to prevent

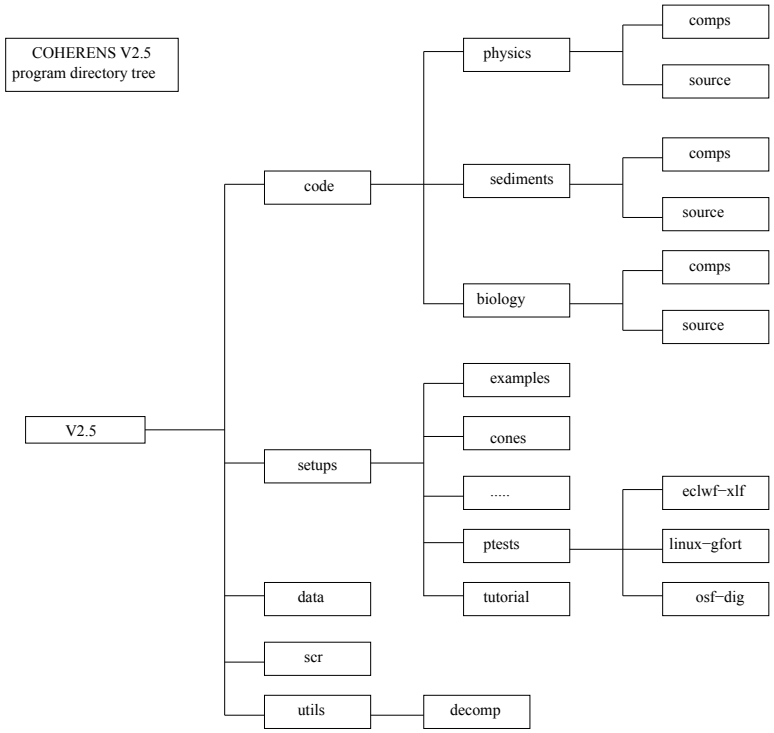


Figure 1.1: COHERENS directory structure

errors during compilation and do not contain actual code. A complete listing of these routines (and related files) is presented in Chapter 34 of the User Documentation.

2. If SEDMOD is set to the sediment directory **COHERENS/code/sediment**, the code is compiled with the COHERENS sediment module.

3. The user may provide his/hers own sediment module, in which case SEDMOD should be set to a path where the user has located the source code (i.e. in $(SEDMOD)/source) and the files for compilation (in $(SEDMOD)/comps). Note that, in that case, the previously mentioned generic routines need to be provided by the user.

Model code and setup

switches

The following switches have been added in *switches.f90*:

iopt_curr_wfall Type of formulation for the settling of particulate matter.

1: settling enabled without correction terms

2: settling enabled with the correction terms (7.117)–(7.118) included. This option is currently disabled.

iopt_kinvisc Formulation for kinematic viscosity.

0: user-defined uniform value kinvisc_cst (default)

1: ITTC (1978) relation (7.24)

iopt_obc_sed (General) type of open boundary conditions for sediments.

0: default conditions at all open boundaries (default option)

1: non-default conditions for at least one open boundary point

iopt_scal_depos Discretisation for the deposition (vertical advective flux at the sea bed) of particulate matter.

0: Deposition flux is set to zero.

1: first order (upwind) scheme (default)

2: second order scheme using extrapolation

iopt_sed Disables/enables (0/1) the activation of an external sediment module. Default is off.

iopt_turb_kinvisc Selects the type of background mixing mixing.

0: user-defined constant value vdifmom_cst (default)

1: kinematic viscosity as selected by iopt_kinvisc

iopt_waves Type of wave input.

0: wave input disabled (default)

1: wave height, period and wave direction

2: wave height, period, velocity, excursion and direction

A parabolic eddy viscosity profile can be selected by setting iopt_turb_alg=6.

key ids

forcing attributes

io_fincon Forcing id for final conditions. In the previous versions this key id was the same as the one for initial conditions (io_inicon).

io_sednst output file(s) for sediment nesting

io_sedobc definitions of open boundary conditions for sediment variables (file number equals 1) or input of open boundary data (file number larger than 1)

io_sedspc (time-independent) arrays used for the setup of a sediment model (particle attributes in the COHERENS sediment model)

io_wavgrd surface wave grid

io_wavsur surface wave data

other

igrd_waves key id of a surface wave grid

ics_sed initial condition file number for sediments

itm_sed timer key id for sediment routines

model parameters and arrays

The following parameter can be defined in usrdef_mod_params or the CIF

kinvisc_cst Constant value for the kinematic viscosity if iopt_kinvisc=1. Default is 1.E0-06 m/s^2.

The following arrays need to be defined in usrdef_nstgrd_spec in case sediment fractions are used for nesting

nosednst(nonestsets) number of fractions for each sub-grid

intsed(nf,nonestsets) fraction numbers for each sub-grid

where nf equals the number of sediment fractions (defined in the setup of the sediment model).

A series of wave arrays are introduced[4]

wavedir* Wave direction [rad]

wavexcurs* Near-bottom wave excursion amplitude [m]

wavefreq Peak wave frequency [rad/s]

waveheight* Significant wave height [m]

wavenum Wave number [1/m]

waveperiod* Peak wave period at [s]

[4]The arrays marked by a "*" can be used for input in usrdef_surface_data depending on the value of iopt_waves.

waveuvel	Near-bottom wave orbital velocity in the X-direction [m/s]
wavevel*	Near-bottom wave orbital velocity [m/s]
wavevvel	Near-bottom wave orbital velocity in the Y-direction [m/s]

model output

The derived type arrays tsrvars, avrvars, analvars have an additional (optional) attribute numvar representing the variable number in case of multi-variable arrays, such as sediment fractions. The number then represents the last index of the data variable (e.g. fraction number).

model routines

The following new routines have been implemented

- *Diffusion_Coefficients.F90*

kinematic_viscosity	calculates kinematic viscosity as function of temperature using equation (7.24)

- *math_library.F90*

gauss_squad	Returns the locations and weights for numerical integration using the Gauss-Legendre quadrature method.
poly_all_roots	Finds all roots of a polynomial using Laguerre's method.
poly_diff	Divides two polynomials.
vector_mag_arr_at*	Calculates the magnitude and/or phase of a vector array at the C-, U- or V-node.
vector_mag_var_at*	Calculates the magnitude and/or phase of a scalar vector at the C-, U- or V-node.

For technical reasons a number of routines have been moved from turbulence_routines.F90 to the following files:

- buoyancy_frequency to *Density_Equations.F90*
- shear_frequency to *Hydrodynamic_Equations.F90*
- dissip_lim_conds, dissip_to_zlmix, zlmix_lim_conds, zlmix_to_dissip to *Turbulence_Equations.F90*

Calls to "generic" routines for sediments are made from the following routines:

`baroclinic_gradient`, `buoyancy_frequency`, `coherens_main`, `define_out0d_vals`, `define_out2d_vals`, `define_out3d_vals`, `equation_of_sate`, `initialise_model`, `inquire_var`, `set_modfiles_atts`, `set_modvars_atts`, `simulation_end`

Test cases

Seven additional test cases for sediments are implemented. For details see Chapter 29 of the User Documentation.

Compatibility with previous versions

Except for the changes in *coherensflags.cmp* and the new test cases the setup and test cases of version 2.5 are backwards compatible.

Version V2.5.1

Coherens Version	: V2.5.1
previous release	: V2.5
Revision	: 613
svn path	: http://svn.mumm.ac.be/svn/root/coherens/versions/V2.5.1
Date of release	: 2013-08-08
Code	: coherensV2.5.1.tar.gz
User documentation	: User_Documentation_V2.5.1.pdf
Reference manual	: Reference_Manual_V2.5.1.pdf

Implementations

Novel features have been implemented for the generation of model grids in the horizontal and vertical directions.

horizontal In this release an option has been provided to align a rectangular model grid (i.e. grids for which $h_1(\xi_1)$, $h_2(\xi_2)$) with specific features such as coast lines, bathymetric contours or open boundaries by means of a simple grid rotation. In the spherical case, this is achieved by a coordinate transformation obtained by displacing the North Pole to a new location, in the Cartesian case by rotating the coordinate axes. For more details, see Section 4.1.3 of the User Documentation.

vertical A new switch iopt_vtype_transf is introduced which automatically generates several vertical grid transforms discussed in Section 4.1.4 of the Documentation.

Model code and setup

switches

iopt_grid_vtype_transf This new switch allows to select automatically different types of vertical grids transformation. Default is zero.

- 0 : uniform vertical grid (iopt_grid_vtype=1) or user-defined
- 11: log-transformation (4.23) at the bottom following Davies & Jones (1991) if iopt_grid_vtype=2
- 12: log-transformation (4.24) at the surface following Davies & Jones (1991) if iopt_grid_vtype=2
- 13: transformation with enhanced resolution near the bottom and/or the bottom as defined in Burchard & Bolding (2002)
- 21: Song & Haidvogel (1994) transformation given by (4.33) and (4.35) if iopt_grid_vtype=3

iopt_grid_node This switch allowed, in previous releases, to define bathymetry and vertical grid either at C- or at corner nodes. However, the latter option seems to provide no real advantage and becomes problematic in channels having a width of just one grid spacing. For this reason, the first option will always be selected in the code and the switch is no longer available in the current implementation.

model setup parameters

A rotated (Cartesian or rectangular spherical) grid is selected by defining the following new attributes for the user-defined derived type variable surfacegrids(igrd_model,1):

rotated must be set to .TRUE. in case of a rotated grid. Default is .FALSE..

gridangle grid rotation angle α (see Section 4.1.3) (decimal degrees). Must be between 0 and 180^0.

y0rot transformed latitude of the reference location in case of a rotated grid (decimal degrees). Only used for spherical (rotated) grids.

The following additional setup parameters are introduced for making vertical grid transforms (default in parentheses).

b_SH	Parameter b in the Song & Haidvogel (1994) vertical grid transformation (0.1)
dl_BB	Parameter d_l in the Burchard & Bolding (2002) vertical grid transformation (4.26) (1.5)
du_BB	Parameter d_u in the Burchard & Bolding (2002) vertical grid transformation (4.26) (1.5)
hcrit_SH	Parameter h_{crit} in the Song & Haidvogel (1994) vertical grid transformation (0.1)
sigstar_DJ	Parameter σ_* in the Davies & Jones (1991) vertical grid transformations (4.23) and (4.24) (0.0)
sig0_DJ	Parameter σ_0 in the Davies & Jones (1991) vertical grid transformations (4.23) and (4.24) (0.1)
theta_SH	Parameter θ in the Song & Haidvogel (1994) vertical grid transformation (8.0)

Test cases

No new test cases have been defined for this release.

Compatibility with previous versions

This version is compatible with previous versions without changes.

Version V2.6

Coherens Version	: V2.6
previous release	: V2.5.1
Revision	: 683
svn path	: http://svn.mumm.ac.be/svn/root/coherens/versions/V2.6
Date of release	: 2014-03-13
Code	: coherensV2.6.tar.gz
User documentation	: User_Documentation_V2.6.pdf
Reference manual	: Reference_Manual_V2.6.pdf

Implementations

Modules for hydraulic structures and discharges have been installed within this version of the code. A description can be found in Chapter 6 of the User Documentation. New features are:

- Dry cells which can be taken as permanently dry during the simulation.
- Thin dams, which are defined as infinitely thin vertical walls. They are located at velocity nodes and prohibit flow exchange and fluxes of scalars between the two adjacent computational grid cells without reducing the total wet surface and the volume of the model.
- Weirs which are similar to thin dams, except that a weir can be inundated, in which case an energy loss is generated. This structure will work as a thin dam in cases where the total water depth upstream of the structure is less than the crest level of the structure. In this case a blocking of flow exchange is imposed by the module. Groynes are typical examples of weirs.
- Barriers which have an opening close to the bottom, where users can define the width of the opening and the height of the sill.

- A discharge module has been implemented. Discharges are represented as sources or sinks in the continuity, momentum and scalar equations supplied at specified (fixed or moving) locations at the surface, bottom or within the water column (e.g. discharge structures, pumping stations, discharge from moving ships ...) by adding water to the water column with a specified salinity, temperature or contimant concentration. The discharge may or may not have a preferential direction.

Model setup

To apply COHERENS with structures and/or discharges the following switches and general parameters may need to be defined in *Usrdef_Model.f90*.

iopt_dischr	Disables/enables (0/1) the discharge module.
iopt_drycel	Disables/enables (0/1) the dry cell module.
iopt_thndam	Disables/enables (0/1) the thin dam module.
iopt_weibar	Disables/enables (0/1) the weirs/barriers module.
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

For details see Chapter 14.

The following additional key ids are available (where a "*" marks a forcing with time series data which can be spread over multiple files).

io_drycel	dry cell locations
io_thndam	thin dam locations
io_weibar	weirs/barries locations and parameters
io_disspc	discharge specifiers
io_disloc*	discharge locations
io_disvol*	volume discharge data
io_discur*	momentum discharge data
io_dissal*	salinity discharge data
io_distmp*	temperature discharge data

A new setup file *Usrdef_Structures.f90* has been created for setting up appplications with structures and/or discharges. The file contains the following routines

- usrdef_dry_cells: setup of the dry cells module
- usrdef_thin_dams: setup of the thin dams module
- usrdef_weirs: setup of the weirs/barriers module
- usrdef_dischr_spec: specifiers for the discharge module
- usrdef_dischr_data: defines discharge data

For more details see Chapter 18.

Model code

The following new source files have been created

1. *structures.f90*
   Declaration of paramameters for structures and discharges. See Section 33.18 in the Reference Manual.

2. *Structures_Model.f90*
   Program unit with all routines related to structures and discharges. See Section 30.22 in the Reference Manual.

A list of most relevant changes in existing source files is given below.

1. *Density_Equations.F90*

salinity_equation	Routines update_dischr_data and scalar_discharge (defined in *Structures_Model.f90*) are called in case the discharge module (for salinity) is activated.
temperature_equation	Routines update_dischr_data and scalar_discharge (defined in *Structures_Model.f90*) are called in case the discharge module (for salinity) is activated.

2. *Grid_Arrays.F90*
   A new routine update_pointer_arrays is created which sets currents to zero at blocking velocity interfaces. The routine is called by the weirs/barriers and inundation modules.

3. *Harmonic_Analysis.f90*
   The first dimensions of lstatprocs have been interchanges, i.e. lstatprocs(nprocs,maxstats,nosetsanal) becomes lstatprocs(maxstats,nprocs,nosetsanal)

4. *Hydrodynamic_Equations.F90*
   A number of new calls (defined in *Structures_Model.f90* are implemented in the following routines (depending on whether the weirs/barriers or discharge module has been activated for the specific routine).

   current_pred : momentum_discharge_3d, weirs_loss, weirs_sink

   current_2d : momentum_discharge_2d, weirs_loss, weir_sink

   surface_elevation: surface_discharge

5. *Inundation_Schemes.f90*
   Routine update_pointer_arrays is called in mask_function.

6. *Model_Finalisation.f90*
   The energy losses are written (in write_phsics) to the final condition file for weirs/barriers.

7. *Model_Initialisation.f90*
   If the weirs/barrier unit has been activated, two additional vector arrays are read (in read_phsics) from the initial condition file.

   wbarelossu: energy losses at U-nodes

   wbarelossv: energy losses at V-nodes

8. *Nested_Grids.F90*
   The first two dimensions of the arrays indexnstc, indexnstu, indexnstv, indexnstuv have been interchanged.

9. *array_interp.f90*
   Bugs have been corrected in Cvar_at_U and VWvar_at_U.

10. *grid.f90*

    - The arrays klevbotatu, klevbotatv, klevsuratu, klevsuratv have become redundant and removed at all places in the source code.
    - Parameters nobuloc1, nobvloc1, nobxloc1, nobyloc1 have be renamed to respectively nobuloc_ext, nobvloc_ext, nobxloc_ext, nobyloc_ext for transparancy.

- Parameters nobuloc2, nobvloc2, nobxloc2, nobyloc2 are not used in the code and have therefore been removed.

11. *inout_paral.f90*

    combine_write_stats_glb This generic routine replaces and extends the old routine combine_write_obc. The routine combines the elements of a real global array whose elements in the first dimension are defined at different domains on the parallel grid and writes the resulting array to the appropriate output file. For a syntax description see Section 31.11.

    combine_write_stats_loc This generic routine replaces the old routine combine_write_stats. The routine combines arrays defined locally on different sub-domains of the parallel grid to a global array which is written to the appropriate output file. For a syntax description see Section 31.11.

12. *paral_comms.f90*

    combine_stats_glb This generic routine replaces and extends the old routine combine_obc. The routine combines the elements of a real global array whose elements in the first dimension are defined at different domains on the parallel grid. For a syntax description see Section 31.17.

    combine_stats_loc This generic routine replaces the old routine combine_stats. The routine combines arrays defined locally on different sub-domains of the parallel grid to a global array. For a syntax description see Section 31.17.

13. *switches.f90*
    The switch iopt_structs has been replaced (for transparency of the code) by iopt_arrint_3D having the same purpose. The switch is automatically switched on if the weirs/barrizers module is activated.

Test cases

The following three new test cases have been implemented for testing the structures and discharge modules.

drythin Simulates the tidal flows around obstacles, either represented by a block of dry cells or a series of thin dams within an open channel.

weirbar A series of experiments are defined simulating the tidal flows over weirs and barriers within an open channel.

discharges The experiments are designed to test the various options of the discharge module.

Compatibility with previous versions

This version is compatible with previous versions without changes (taking account of the minor modifications mentioned above). The old test cases can be run as before.