Chapter 30

Description of external routines

30.1 *Advection_Terms.F90*

Advective terms in all transport equations: scalar equations at the C-node, 2-D and 3-D momentum equations at the U- and V-nodes, turbulence equations at the W-nodes.

Xadv_at_C

```
SUBROUTINE Xadv_at_C(psic,tridcfc,novars,iopt_adv,nh,klo,kup,psiobu,uadvvel)
INTEGER, INTENT(IN) :: iopt_adv, klo, kup, nh, novars
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                           & 1-nhalo:nrloc+nhalo,nz,novars) :: psic
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc,nz,4,novars) :: tridcfc
REAL, INTENT(IN), DIMENSION(nobu,nz,novars) :: psiobu
REAL, INTENT(IN), DIMENSION(2-nhalo:ncloc+nhalo,nrloc,nz,novars) :: uadvvel
```

File
: *Advection_Terms.F90*

Type
: Subroutine

Purpose
: Advective term in the X-direction for one or more scalar quantities at the C-nodes

Reference
: Section 5.5.4.1

Arguments

psic	C-node quantity or quantities to be advected	[psic]
tridcfc	Tridiagonal matrix for storing explicit and implicit (at open boundaries) parts of the advective term	[psic]
novars	Number of variables to be advected	
iopt_adv	Switch to select the advection scheme	
nh	Halo size of local arrays	
klo	1 for a Neumann, 2 for a Dirichlet condition at the bottom	
kup	nz for a Neumann, nz-1 for a Dirichlet condition at the surface	
psiobu	Data profiles at open boundaries. Flagged data indicate zero gradient conditions.	[psic]
uadvvel	Advective velocity	[m/s]

Calling procedures

transport_at_C_3d, transport_at_C_4d1, transport_at_C_4d2

Xadv_at_U_2d

```
SUBROUTINE Xadv_at_U_2d(psiu,xadvatu2d,nh,vintflag)
LOGICAL, INTENT(INOUT) :: vintflag
INTEGER, INTENT(IN) :: nh
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo) :: psiu
REAL, INTENT(OUT), DIMENSION(ncloc,nrloc) :: xadvatu2d
```

File

Advection_Terms.F90

Type

Subroutine

Purpose

Alongstream advective term for the X-component of the depth-integrated current U at the U-nodes

Reference

Section 5.3.5.1

Arguments

psiu	U-node quantity to be advected	[m^2/s]
xadvatu2d	Advective term (times $\Delta\tau$)	[m^2/s]
nh	Halo size of local arrays	
vintflag	Add the result to the third (2-D barotropic) part of equation (5.164) if set to .TRUE. at predictor time steps.	

Calling procedures
transport_at_U_2d

Xadv_at_U_3d

```
SUBROUTINE Xadv_at_U_3d(psiu,xadvatu3d,nh,vintflag)
LOGICAL, INTENT(IN) :: vintflag
INTEGER, INTENT(IN) :: nh
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz) :: psiu
REAL, INTENT(OUT), DIMENSION(ncloc,nrloc,nz) :: xadvatu3d
```

File
Advection_Terms.F90

Type
Subroutine

Purpose
Alongstream advection term for the 3-D current u at the U-nodes

Reference
Section 5.3.4.1

Arguments

psiu	U-node quantity to be advected	[m/s]
xadvatu3d	Advective term (times h_3^u)	[m^2/s^2]
nh	Halo size of local arrays	
vintflag	Add the result to the first (3-D) part of equation (5.164) if set to .TRUE. at predictor time steps.	

Calling procedures
transport_at_U_3d

Xadv_at_V_2d

```
SUBROUTINE Xadv_at_V_2d(psiv,xadvatv2d,nh,vintflag)
LOGICAL, INTENT(INOUT) :: vintflag
INTEGER, INTENT(IN) :: nh
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo) :: psiv
REAL, INTENT(OUT), DIMENSION(ncloc,nrloc) :: xadvatv2d
```

File

Advection_Terms.F90

Type

Subroutine

Purpose

Cross-stream advective term in the X-direction for the Y-component of the depth-integrated current V at the V-nodes

Reference

Section 5.3.5.3

Arguments

psiv	V-node quantity to be advected	$[m^2/s]$
xadvatv2d	Advective term (times $\Delta\tau$)	$[m^2/s]$
nh	Halo size of local arrays	
vintflag	Add the result to the third (2-D barotropic) part of equation (5.165) if set to .TRUE. at predictor time steps.	

Calling procedures

transport_at_V_2d

Xadv_at_V_3d

```
SUBROUTINE Xadv_at_V_3d(psiv,xadvatv3d,nh,vintflag)
LOGICAL, INTENT(IN) :: vintflag
INTEGER, INTENT(IN) :: nh
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz) :: psiv
REAL, INTENT(OUT), DIMENSION(ncloc,nrloc,nz) :: xadvatv3d
```

File

Advection_Terms.F90

Type
Subroutine

Purpose
Cross-stream advective term for the 3-D current v at the V-nodes

Reference
Section 5.3.4.3

Arguments

psiv	V-node quantity to be advected	[m/s]
xadvatv3d	Advective term (times h_3^v)	[m^2/s^2]
nh	Halo size of local arrays	
vintflag	Add the result to the second (3-D) part of equation (5.165) if set to .TRUE. at predictor time steps.	

Calling procedures
transport_at_V_3d

Xadv_at_W

```
SUBROUTINE Xadv_at_W(psiw,tridcfw,uadvatuw,nh,klo,kup)
INTEGER, INTENT(IN) :: klo, kup, nh
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz+1) :: psiw
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc,nz+1,4) :: tridcfw
REAL, INTENT(IN), DIMENSION(2-nhalo:ncloc+nhalo,&
                          & nrloc,2:nz) :: uadvatuw
```

File
Advection_Terms.F90

Type
Subroutine

Purpose
Advective term in the X-direction for a quantity at the W-nodes

Reference
Section 5.6.2.1

Arguments

psiw	W-node quantity to be advected	[psiw]
tridcfw	Tridiagonal matrix to store explicit and implicit (at open boundaries) parts of the advective term	[psiw]
uadvatuw	Advective velocity at the UW-nodes	[m/s]
nh	Halo size of local arrays	
klo	2 for a Neumann or a Dirichlet condition at the bottom, 3 for a Dirichlet condition at the first W-node above the bottom	
kup	nz for a Neumann condition or a Dirichlet condition at the surface, nz-1 for a Dirichlet condition at the first W-node below the surface	

Calling procedures

transport_at_W

Yadv_at_C

```
SUBROUTINE Yadv_at_C(psic,tridcfc,novars,iopt_adv,nh,klo,kup,psiobv,vadvvel)
INTEGER, INTENT(IN) :: iopt_adv, klo, kup, nh, novars
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz,novars) :: psic
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc,nz,4,novars) :: tridcfc
REAL, INTENT(IN), DIMENSION(nobv,nz,novars) :: psiobv
REAL, INTENT(IN),DIMENSION(ncloc,2-nhalo:nrloc+nhalo,nz,novars) ::  vadvvel
```

File

Advection_Terms.F90

Type

Subroutine

Purpose

Advective term in the Y-direction for one or more scalar quantities at the C-nodes

Reference

Section 5.5.4.2

Arguments

psic	C-node quantity or quantities to be advected	[psic]

tridcfc	Tridiagonal matrix for storing explicit and implicit (at open boundaries) parts of advective term	[psic]
novars	Number of variables to be advected.	
iopt_adv	Switch to select the advection scheme	
nh	Halo size of local arrays	
klo	1 for a Neumann, 2 for a Dirichlet condition at the bottom	
kup	nz for a Neumann, nz-1 for a Dirichlet condition at the surface	
psiobv	Data profiles at open boundaries. Flagged data indicate zero gradient conditions.	[psic]
uadvvel	Advective velocity	[m/s]

Calling procedures

transport_at_C_3d, transport_at_C_4d1, transport_at_C_4d2

Yadv_at_U_2d

```
SUBROUTINE Yadv_at_U_2d(psiu,yadvatu2d,nh,vintflag)
LOGICAL, INTENT(INOUT) :: vintflag
INTEGER, INTENT(IN) :: nh
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo) :: psiu
REAL, INTENT(OUT), DIMENSION(ncloc,nrloc) :: yadvatu2d
```

File

Advection_Terms.F90

Type

Subroutine

Purpose

Cross-stream advective term for the X-component of the depth-integrated current U at the U-nodes

Reference

Section 5.3.5.2

Arguments

psiu	U-node quantity to be advected	$[m^2/s]$
yadvatu2d	Advective term (times $\Delta\tau$)	$[m^2/s]$

nh	Halo size of local arrays
vintflag	Add the result to the fourth (2-D barotropic) part of equation (5.164) if set to .TRUE. at predictor time steps.

Calling procedures
transport_at_U_2d

Yadv_at_U_3d

```
SUBROUTINE Yadv_at_U_3d(psiu,yadvatu3d,nh,vintflag)
LOGICAL, INTENT(IN) :: vintflag
INTEGER, INTENT(IN) :: nh
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz) :: psiu
REAL, INTENT(OUT), DIMENSION(ncloc,nrloc,nz) :: yadvatu3d
```

File
Advection_Terms.F90

Type
Subroutine

Purpose
Cross-stream advective term for the 3-D current u at the U-nodes

Reference
Section 5.3.4.2

Arguments

psiu	U-node quantity to be advected	[m/s]
yadvatu3d	Advective term (times h_3^u)	[m^2/s^2]
nh	Halo size of local arrays	
vintflag	Add the result to the second (3-D) part of equation (5.164) if set to .TRUE. at predictor time steps.	

Calling procedures
transport_at_U_3d

Yadv_at_V_2d

```
SUBROUTINE Yadv_at_V_2d(psiv,yadvatv2d,nh,vintflag)
LOGICAL, INTENT(INOUT) :: vintflag
INTEGER, INTENT(IN) :: nh
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo) :: psiv
REAL, INTENT(OUT), DIMENSION(ncloc,nrloc) :: yadvatv2d
```

File

Advection_Terms.F90

Type

Subroutine

Purpose

Alongstream advective term for the Y-component of the depth-integrated current V at the V-nodes

Reference

Section 5.3.5.4

Arguments

psiv	V-node quantity to be advected	$[m^2/s]$
yadvatv2d	Advective term (times $\Delta\tau$)	$[m^2/s]$
nh	Halo size of local arrays	
vintflag	Add the result to the fourth (2-D barotropic) part of equation (5.165) if set to .TRUE. at predictor time steps.	

Calling procedures

transport_at_V_2d

Yadv_at_V_3d

```
SUBROUTINE Yadv_at_V_3d(psiv,yadvatv3d,nh,vintflag)
LOGICAL, INTENT(IN) :: vintflag
INTEGER, INTENT(IN) :: nh
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz) :: psiv
REAL, INTENT(OUT), DIMENSION(ncloc,nrloc,nz) :: yadvatv3d
```

File

Advection_Terms.F90

Type
Subroutine

Purpose
Alongstream advective term for the 3-D current v at the V-nodes

Reference
Section 5.3.4.4

Arguments

psiv	V-node quantity to be advected	[m/s]
yadvat3d	Advective term (times h_3^v)	[m^2/s^2]
nh	Halo size of local arrays	
vintflag	Add the result to the second (3-D) part of equation (5.165) if set to .TRUE. at predictor time steps.	

Calling procedures
transport_at_V_3d

Yadv_at_W

```
SUBROUTINE Yadv_at_W(psiw,tridcfw,vadvatvw,nh,klo,kup)
INTEGER, INTENT(IN) :: klo, kup, nh
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz+1) :: psiw
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc,nz+1,4) :: tridcfw
REAL, INTENT(IN), DIMENSION(ncloc,2-nhalo:nrloc+nhalo,&
                          & 2:nz) :: vadvatvw
```

File
Advection_Terms.F90

Type
Subroutine

Purpose
Advective term in the Y-direction for a quantity at the W-nodes

Reference
Section 5.6.2.2

Arguments

psiw	W-node quantity to be advected	[psiw]
tridcfw	Tridiagonal matrix for storing explicit and implicit (at open boundaries) parts of the advective term	[psiw]
vadvatvw	Advective velocity at the VW-nodes	[m/s]
nh	Halo size of local arrays	
klo	2 for a Neumann or a Dirichlet condition at the bottom, 3 for a Dirichlet condition at the first W-node above the bottom	
kup	nz for a Neumann condition or a Dirichlet condition at the surface, nz-1 for a Dirichlet condition at the first W-node below the surface	

Calling procedures
transport_at_W

Zadv_at_C

```
SUBROUTINE Zadv_at_C(psic,tridcfc,wadvatw,novars,iopt_adv,klo,kup)
INTEGER, INTENT(IN) :: iopt_adv, klo, kup, novars
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz,novars) :: psic
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc,nz,4,novars) :: tridcfc
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nz+1,novars) :: wadvatw
```

File
Advection_Terms.F90

Type
Subroutine

Purpose
Advective term in the vertical direction for one or more scalar quantities at the C-nodes

Reference
Section 5.5.4.3

Arguments

psic	C-node quantity or quantities be advected	[psic]
tridcfc	Tridiagonal matrix for storing implicit and explicit terms	[psic]

wadvatw	Advective velocity at the W-nodes	[m/s]
novars	Number of variables to be advected	
iopt_adv	Switch to select the vertical advection scheme	
klo	1 for a Neumann, 2 for a Dirichlet condition at the bottom	
kup	nz for a Neumann, nz-1 for a Dirichlet condition at the surface	

Calling procedures
transport_at_C_3d, transport_at_C_4d1, transport_at_C_4d2

Zadv_at_U

```
SUBROUTINE Zadv_at_U(psiu,tridcfu,wadvatuw)
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz) :: psiu
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc,nz,4) :: tridcfu
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nz+1) :: wadvatuw
```

File
Advection_Terms.F90

Type
Subroutine

Purpose
Vertical advective term for the 3-D current u at the the U-nodes

Reference
Section 5.3.7.1

Arguments

psiu	U-node quantity to be advected	[m/s]
tridcfu	Tridiagonal matrix for storing implicit and explicit terms	[m/s]
wadvatuw	Advective velocity at the UW-nodes	[m/s]

Calling procedures
transport_at_U_3d

Zadv_at_V

```
SUBROUTINE Zadv_at_V(psiv,tridcfv,wadvatvw)
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz) :: psiv
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc,nz,4) :: tridcfv
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nz+1) :: wadvatvw
```

File
Advection_Terms.F90

Type
Subroutine

Purpose
Vertical advective term for the 3-D current v at the V-nodes

Reference
Section 5.3.7.2

Arguments

psiv	V-node quantity to be advected	[m/s]
tridcfv	Tridiagonal matrix for storing implicit and explicit terms	[m/s]
wadvatvw	Advective velocity at the VW-nodes	[m/s]

Calling procedures
transport_at_V_3d

Zadv_at_W

```
SUBROUTINE Zadv_at_W(psiw,tridcfw,wadvatc,klo,kup)
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz+1) :: psiw
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc,nz+1,4) :: tridcfw
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nz) :: wadvatc
```

File
Advection_Terms.F90

Type
Subroutine

Purpose
Vertical advective term for a quantity at the W-nodes

Reference
Section 5.6.2.3

Arguments

psiw	W-node quantity to be advected	[psiw]
tridcfw	Tridiagonal matrix for storing implicit and explicit terms	[psiw]
wadvatw	Advective velocity at the W-nodes	[m/s]
klo	2 for a Neumann or a Dirichlet condition at the bottom, 3 for a Dirichlet condition at the first W-node above the bottom	
kup	nz for a Neumann condition or a Dirichlet condition at the surface, nz-1 for a Dirichlet condition at the first W-node below the surface	

Calling procedures
transport_at_W

30.2 *Allocate_Arrays.f90*

Allocation and deallocation of arrays which have a global scope in the program. The arrays are declared as allocatable arrays in the respective module files (see Table 12.1).

allocate_global_grid

`SUBROUTINE allocate_global_grid`

File
Allocate_Arrays.f90

Type
Subroutine

Purpose
Allocate grid arrays on the global (parallel) grid.

Calling procedures
initialise_model

allocate_mod_arrays

`SUBROUTINE allocate_mod_arrays`

File
Allocate_Arrays.f90

Type
Subroutine

Purpose
Allocate model arrays on the local sub-domain (mostly including an halo) with a "global" scope in the program, i.e. arrays used in different program routines. The arrays are declared as allocatable arrays in MODULE units of the program.

Reference
Sections 10.1.1

Arguments
None

Calling procedures
initialise_model
simulation_end

deallocate_mod_arrays

`SUBROUTINE deallocate_mod_arrays`

File
Allocate_Arrays.f90

Type
Subroutine

Purpose
Deallocate all model arrays with a "global" scope at the end of a simulation.

Calling procedures
simulation_end

30.3 *Bottom_Fluxes.f90*

Routines used for the calculation of the bottom stress.

bottom_stress

`SUBROUTINE bottom_stress`

File
Bottom_Fluxes.f90

Type
Subroutine

Purpose
Base program unit for the evaluation of the bottom stress.

Reference
Sections 4.9.2, 5.3.1.2 and 5.3.15.2

Called internal procedures
bottom_currents, bottom_drag_coef1, bottom_drag_coef2, bottom_stress_linear, bottom_stress_quadr

Calling procedures
current_corr, current_corr_1d, current_2d, initialise_model

bottom_currents

`SUBROUTINE bottom_currents`

File
Bottom_Fluxes.f90

Type
Internal subroutine

Purpose
Magnitude of the current at the bottom level of the model grid

Calling procedures
bottom_stress

bottom_drag_coef1

`SUBROUTINE bottom_drag_coef1`

File
Bottom_Fluxes.f90

Type
Internal subroutine

Purpose
Interpolate the externally imposed bottom drag coefficient at the U- and the V-nodes when iopt_bstres_drag≤2.

Calling procedures
bottom_stress

bottom_drag_coef2

`SUBROUTINE bottom_drag_coef2`

File
Bottom_Fluxes.f90

Type
Internal subroutine

Purpose
Update the bottom drag coefficients using imposed roughness lengths when iopt_bstres_drag>2.

Reference
Equation (5.18) or (5.19)

Calling procedures
bottom_stress

bottom_stress_linear

`SUBROUTINE bottom_stress_linear`

File
Bottom_Fluxes.f90

Type
Internal subroutine

Purpose
Evaluate the bottom stress using a linear friction law.

Reference
Equation (4.338) or (4.339)

Calling procedures
bottom_stress

bottom_stress_quadr

`SUBROUTINE bottom_stress_quadr`

File
Bottom_Fluxes.f90

Type
Internal subroutine

Purpose
Evaluate the bottom stress using a quadratic friction law.

Reference
Equation (4.340) or (4.341)

Calling procedures
bottom_stress

30.4 *Coherens_Program.f90*

coherens_main

`SUBROUTINE coherens_main`

File
Coherens_Program.f90

Type
Main program

Purpose
Main program unit of COHERENS

Called external procedures
astronomical_tides, baroclinic_gradient, coherens_end, coherens_start, define_phsics equation_of_state, harmonic_analysis, horizontal_diff_coefs, hydrodynamic_equations, initialise_model, mask_function, meteo_input, salinity_equation, sediment_equation, simulation_end, simulation_start, store_depths_old, temperature_equation, time_averages, time_series, usrdef_output, vertical_diff_coefs, wave_input, write_phsics, write_sedics

30.5 *Corrector_Terms.F90*

Evaluate the corrector terms in the scalar transport equations. For a definition see equations (5.385)–(5.387) for scalars at the C-nodes and (5.507)–(5.509) for scalars at the W-nodes.

Xcorr_at_C

```
SUBROUTINE Xcorr_at_C(psic,ucorratc,novars,klo,kup)
INTEGER, INTENT(IN) :: klo, kup, novars
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz,novars) :: psic
REAL, INTENT(OUT), DIMENSION(ncloc,nrloc,klo:kup,novars) :: ucorratc
```

File
Corrector_Terms.F90

Type
Subroutine

Purpose
Corrector term in the X-direction for a quantity at the C-nodes

Reference
Equation (5.425)

Arguments

psic	C-node quantity or quantities to be advected	[psic]
ucorratc	Corrector term (times Δt)	[psic]
novars	Number of variables to be advected	

klo	1 for a Neumann, 2 for a Dirichlet condition at the bottom
kup	nz for a Neumann, nz-1 for a Dirichlet condition at the surface

Calling procedures

transport_at_C_3d, transport_at_C_4d1, transport_at_C_4d2

Xcorr_at_W

```
SUBROUTINE Xcorr_at_W(psiw,ucorratw,uvelatuw,klo,kup)
INTEGER, INTENT(IN) :: klo, kup
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz+1) :: psiw
REAL, INTENT(OUT), DIMENSION(ncloc,nrloc,klo:kup) :: ucorratw
REAL, INTENT(IN), DIMENSION(2-nhalo:ncloc+nhalo,&
                          & nrloc,2:nz) :: uvelatuw
```

File

Corrector_Terms.F90

Type

Subroutine

Purpose

Corrector term in the X-direction for a quantity at the W-nodes

Reference

Equation (5.507)

Arguments

psiw	W-node quantity to be advected	[psiw]
ucorratw	Corrector term (times Δt)	[psiw]
uvelatuw	X-component of the current u at the UW-nodes	[m/s]
klo	2 for a Neumann or a Dirichlet condition at the bottom, 3 for a Dirichlet condition at the first W-node above the bottom	
kup	nz for a Neumann condition or a Dirichlet condition at the surface, nz-1 for a Dirichlet condition at the first W-node below the surface	

Calling procedures

transport_at_W

Ycorr_at_C

```
SUBROUTINE Ycorr_at_C(psic,vcorratc,novars,klo,kup)
INTEGER, INTENT(IN) :: klo, kup, novars
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz,novars) :: psic
REAL, INTENT(OUT), DIMENSION(ncloc,nrloc,klo:kup,novars) :: vcorratc
```

File
Corrector_Terms.F90

Type
Subroutine

Purpose
Corrector term in the Y-direction for a quantity at the C-nodes

Reference
Equation (5.426)

Arguments

psic	C-node quantity or quantities to be advected	[psic]
vcorratc	Corrector term (times Δt)	[psic]
novars	Number of variables to be advected	
klo	1 for a Neumann, 2 for a Dirichlet condition at the bottom	
kup	nz for a Neumann, nz-1 for a Dirichlet condition at the surface	

Calling procedures
transport_at_C_3d, transport_at_C_4d1, transport_at_C_4d2

Ycorr_at_W

```
SUBROUTINE Ycorr_at_W(psiw,vcorratw,vvelatvw,klo,kup)
INTEGER, INTENT(IN) :: klo, kup
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz+1) :: psiw
REAL, INTENT(OUT), DIMENSION(ncloc,nrloc,klo:kup) :: vcorratw
REAL, INTENT(IN), DIMENSION(ncloc,&
                          & 2-nhalo:nrloc+nhalo,2:nz) :: vvelatvw
```

File
Corrector_Terms.F90

Type
Subroutine

Purpose
Corrector term in the Y-direction for a quantity at the W-nodes

Reference
Equation (5.508)

Arguments

psiw	W-node quantity to be advected	[psiw]
vcorratw	Corrector term (times Δt)	[psiw]
vvelatvw	Y-component of the current v at the VW-nodes	[m/s]
klo	2 for a Neumann or a Dirichlet condition at the bottom, 3 for a Dirichlet condition at the first W-node above the bottom	
kup	nz for a Neumann condition or a Dirichlet condition at the surface, nz-1 for a Dirichlet condition at the first W-node below the surface	

Calling procedures
transport_at_W

Zcorr_at_C

```
SUBROUTINE Zcorr_at_C(psic,wcorratc,novars,klo,kup)
INTEGER, INTENT(IN) :: klo, kup, novars
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz,novars) :: psic
REAL, INTENT(OUT), DIMENSION(ncloc,nrloc,klo:kup,novars) :: wcorratc
```

File
Corrector_Terms.F90

Type
Subroutine

Purpose
Corrector term in the Z-direction for a quantity at the C-nodes

Reference
Equation (5.427)

Arguments

psic	C-node quantity or quantities to be advected	[psic]
wcorratc	Corrector term (times Δt)	[psic]
novars	Number of variables to be advected	
klo	1 for a Neumann, 2 for a Dirichlet condition at the bottom	
kup	nz for a Neumann, nz-1 for a Dirichlet condition at the surface	

Calling procedures
transport_at_C_3d, transport_at_C_4d1, transport_at_C_4d2

Zcorr_at_W

```
SUBROUTINE Zcorr_at_W(psiw,wcorratw,wvelatc,klo,kup)
INTEGER, INTENT(IN) :: klo, kup
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz+1) :: psiw
REAL, INTENT(OUT), DIMENSION(ncloc,nrloc,klo:kup) :: wcorratw
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nz) :: wvelatc
```

File
Corrector_Terms.F90

Type
Subroutine

Purpose
Corrector term in the Z-direction for a quantity at the W-nodes

Reference
Equation (5.509)

Arguments

psiw	W-node quantity to be advected	[psiw]
wcorratw	Corrector term (times Δt)	[psiw]
wvelatc	Transformed vertical current ω at the C-nodes	[m/s]

klo 2 for a Neumann or a Dirichlet condition at the bottom, 3 for a Dirichlet condition at the first W-node above the bottom

kup nz for a Neumann condition or a Dirichlet condition at the surface, nz-1 for a Dirichlet condition at the first W-node below the surface

Calling procedures
transport_at_W

30.6 *Density_Equations.F90*

Routines for all kinds of calculations related to density: salinity and temperature equations, equation of state, optical formulation, baroclinic pressure gradient.

baroclinic_gradient

```
SUBROUTINE baroclinic_gradient
```

File
Density_Equations.F90

Type
Subroutine

Purpose
Evaluate the baroclinic pressure gradient using different formulations selected by iopt_dens_grad.

Reference
Section 5.3.13

Called external procedures
baroclinic_gradient_sed_cubic, baroclinic_gradient_sed_sigma, baroclinic_gradient_sed_z, hcube_deriv, hcube_fluxes

Calling procedures
coherens_main

buoyancy_frequency

`SUBROUTINE buoyancy_frequency`

File
Density_Equations.F90

Type
Subroutine

Purpose
Calculate the squared buoancy frequency N^2 on the model grid using the algorithm given in Section 5.3.12.2.

Reference
Section 5.3.12.2

Called external procedures
buoyancy_frequency_sed

Calling procedures
vertical_diff_coefs

equation_of_state

`SUBROUTINE equation_of_state`

File
Density_Equations.F90

Type
Subroutine

Purpose
Evaluate the density and temperature/salinity expansion/contraction coefficients using different versions of the equation of state.

Reference
Section 4.2.3

Calling procedures
initialise_model

hcube_deriv

```
SUBROUTINE hcube_deriv(psi,dpsi,cdir)
CHARACTER (LEN=1), INTENT(IN) :: cdir
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz) :: psi
REAL, INTENT(OUT), DIMENSION(0:ncloc+1,0:nrloc+1,nz) :: dpsi
```

File
Density_Equations.F90

Type
Subroutine

Purpose
Evaluate the horizontal or vertical derivative terms in the cube-H method for the baroclininic pressure gradient using the discretisations (5.233)–(5.236).

Reference
Section 5.3.13.3

Arguments

psi	Input array
dpsi	Array of derivatives
cdir	Direction of derivative (‘X’,‘Y’,‘Z’)

Calling procedures
baroclinic_gradient

hcube_fluxes

```
SUBROUTINE hcube_fluxes(psi,zcoord,dpsi,dzcoord,fcube,cdir)
CHARACTER (LEN=1), INTENT(IN) :: cdir
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz) :: psi, zcoord
REAL, INTENT(IN), DIMENSION(0:ncloc+1,0:nrloc+1,nz) :: dpsi, dzcoord
REAL, INTENT(OUT), DIMENSION(0:ncloc,0:nrloc,nz) :: fcube
```

File
Density_Equations.F90

Type
Subroutine

Purpose
Evaluate the integrals FU, FV, FW in the cube-H method for the baroclinic pressure gradient using the discretisations (5.228)–(5.232).

Reference
Section 5.3.13.3

Arguments

psi	Density (temperature or salinity) array
zcoord	Array of Z-coordinates
dpsi	Array of psi-derivatives
dzcoord	Array of Z-derivatives
fcube	Pseudo fluxes
cdir	Direction of derivative ('X','Y','Z')

Calling procedures
baroclinic_gradient

heat_optics

```
SUBROUTINE heat_optics
```

File
Density_Equations.F90

Type
Subroutine

Purpose
Evaluate the solar radiance within the water column using the optical formulation (4.59).

Reference
Section 4.2.1.1

Calling procedures
temperature_equation

salinity_equation

`SUBROUTINE salinity_equation`

File
Density_Equations.F90

Type
Subroutine

Purpose
Solve the salinity equation (4.83).

Reference
Section 4.2.1.2

Called external procedures
define_profobc_spec, open_boundary_conds_prof, salinity_flux, transport_at_C_3d, update_nest_data_prof, update_profobc_data

Calling procedures
initialise_model

temperature_equation

`SUBROUTINE temperature_equation`

File
Density_Equations.F90

Type
Subroutine

Purpose
Solve the temperature equation (4.82).

Reference
Section 4.2.1.2

Called external procedures
define_profobc_spec, define_surface_input_grid, heat_flux, heat_optics, open_boundary_conds_prof, solar_irradiance, transport_at_C_3d, update_nest_data_prof, update_profobc_data, update_surface_data

Calling procedures
initialise_model

30.7 *Diffusion_Coefficients.F90*

Calculation of the horizontal and vertical diffusion coefficients.

horizontal_diff_coefs

`SUBROUTINE horizontal_diff_coefs`

File
Diffusion_Coefficients.F90

Type
Subroutine

Purpose
Set the horizontal diffusion coefficients to a constant value on first call if iopt_hdif_coef=1. If iopt_hdif_coef=2, the coefficients are determined using the Smagorinsky formulation.

Reference
Sections 5.3.12.1 and 5.5.6.1

Calling procedures
coherens_main

kinematic_viscosity

`SUBROUTINE kinematic_viscosity`

File
Diffusion_Coefficients.F90

Type
Subroutine

Purpose
Calculate kinematic viscosity as function of temperature.

Reference
Equation (7.24)

Calling procedures
vertical_diff_coefs

vertical_diff_coefs

```
SUBROUTINE vertical_diff_coefs
```

File
Diffusion_Coefficients.F90

Type
Subroutine

Purpose
Calculate the vertical diffusion coefficients ν_T and λ_T using the turbulence model selected by the user. The actual calculation is performed by the routines in *Turbulence_Equations.F90*.

Reference
Section 4.4

Called external procedures
dissipation_equation, eddy_coefs_alg, eddy_coefs_tc, init_turbulence, kl_equation, mixing_length, tke_equation, tke_equilibrium

Calling procedures
coherens_main

30.8 *Diffusion_Terms.F90*

Calculate the diffusion terms in all transport equations: scalar equations at the C-node, 2-D and 3-D momentum equations at the U- and V-nodes, turbulence equations at the W-nodes.

Xdif_at_C

```
SUBROUTINE Xdif_at_C(psic,tridcfc,hdifcoefatu,novars,klo,kup)
INTEGER, INTENT(IN) :: klo, kup
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz,novars) :: psic
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc,nz,4,novars) :: tridcfc
REAL, INTENT(IN), DIMENSION(ncloc+1,nrloc,nz) :: hdifcoefatu
```

File
Diffusion_Terms.F90

Type
Subroutine

Purpose
Diffusion term in the X-direction for one or more scalar quantities at the C-nodes

Reference
Section 5.5.5.1

Arguments

psic	C-node quantity or quantitites to be diffused	[psic]
tridcfc	Tridiagonal solution matrix. The result is stored as an explicit term.	[psic]
hdifcoefatu	Horizontal diffusion coefficient at the U-nodes	[m^2/s]
novars	Number of variables to be diffused	
klo	1 for a Neumann, 2 for a Dirichlet condition at the bottom	
kup	nz for a Neumann, nz-1 for a Dirichlet condition at the surface	

Calling procedures
transport_at_C_3d, transport_at_C_4d1, transport_at_C_4d2

Xdif_at_U_2d

```
SUBROUTINE Xdif_at_U_2d(psiu,xdifatu2d,hdifcoefatc,vintflag)
LOGICAL, INTENT(INOUT) :: vintflag
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo) :: psiu
REAL, INTENT(OUT), DIMENSION(ncloc,nrloc) :: xdifatu2d
REAL, INTENT(IN), DIMENSION(0:ncloc,0:nrloc) :: hdifcoefatc
```

File
Diffusion_Terms.F90

Type
Subroutine

Purpose
Alongstream diffusion term for the X-component of the depth current-integrated current U at the U-nodes

Reference
Section 5.3.9

Arguments

psiu	U-node quantity to be diffused	[m^2/s]
xdifatu2d	Diffusion term times $\Delta\tau$	[m^2/s]
hdifcoefatc	Horizontal diffusion coefficient at the C-nodes	[m^2/s]
vintflag	Add the result to the third (2-D barotropic) part of equation (5.204) if set to .TRUE. at predictor time steps.	

Calling procedures
transport_at_U_2d

Xdif_at_U_3d

```
SUBROUTINE Xdif_at_U_3d(psiu,xdifatu3d,hdifcoefatc,vintflag)
LOGICAL, INTENT(IN) :: vintflag
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz) :: psiu
REAL, INTENT(OUT), DIMENSION(ncloc,nrloc,nz) :: xdifatu3d
REAL, INTENT(IN), DIMENSION(0:ncloc,0:nrloc,nz) :: hdifcoefatc
```

File
Diffusion_Terms.F90

Type
Subroutine

Purpose
Along-stream diffusion term in the X-direction for the X-component of the current u at the U-nodes

Reference
Section 5.3.8

Arguments

psiu	U-node quantity to be diffused	[m/s]
xdifatu3d	Diffusion term (times h_3^u)	[m^2/s^2]
hdifcoefatc	Horizontal diffusion coefficient at the C-nodes	[m^2/s]
vintflag	Add the result to the first (3-D) part of equation (5.204) if set to .TRUE. at predictor time steps.	

Calling procedures
transport_at_U_3d

Xdif_at_V_2d

```
SUBROUTINE Xdif_at_V_2d(psiv,xdifatv2d,hdifcoefatuv,vintflag)
LOGICAL, INTENT(INOUT) :: vintflag
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo) :: psiv
REAL, INTENT(OUT), DIMENSION(ncloc,nrloc) :: xdifatv2d
REAL, INTENT(IN), DIMENSION(ncloc+1,nrloc+1) :: hdifcoefatuv
```

File
Diffusion_Terms.F90

Type
Subroutine

Purpose
Diffusion term in the X-direction for the 2-D current V at the V-nodes

Reference
Section 5.3.9

Arguments

psiv	V-node quantity to be diffused	$[m^2/s]$
xdifatv2d	Diffusion term (times $\Delta\tau$)	$[m^2/s]$
hdifcoefatuv	Horizontal diffusion coefficient at the UV-nodes	$[m^2/s]$
vintflag	Add the result to the third (2-D barotropic) part of equation (5.165) if set to .TRUE. at predictor time steps.	

Calling procedures
transport_at_V_2d

Xdif_at_V_3d

```
SUBROUTINE Xdif_at_V_3d(psiv,xdifatv3d,hdifcoefatuv,vintflag)
LOGICAL, INTENT(IN) :: vintflag
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz) :: psiv
REAL, INTENT(OUT), DIMENSION(ncloc,nrloc,nz) :: xdifatv3d
REAL, INTENT(IN), DIMENSION(ncloc+1,nrloc+1,nz) :: hdifcoefatuv
```

File
Diffusion_Terms.F90

Type
Subroutine

Purpose
Cross-stream advection term for the 3-D current v at the V-nodes

Reference
Section 5.3.8

Arguments

psiv	V-node quantity to be diffused	[m/s]
xdifatv3d	Diffusion term (times h_3^v)	[m^2/s^2]
hdifcoefatuv	Horizontal diffusion coefficient at the UV-nodes	[m^2/s]
vintflag	Add the result to the first (3-D) part of equation (5.205) if set to .TRUE. at predictor time steps.	

Calling procedures
transport_at_V_3d

Xdif_at_W

```
SUBROUTINE Xdif_at_W(psiw,tridcfw,hdifcoefatuw,klo,kup)
INTEGER, INTENT(IN) :: klo, kup
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz+1):: psiw
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc,nz+1,4) :: tridcfw
REAL, INTENT(IN), DIMENSION(ncloc+1,nrloc,2:nz) :: hdifcoefatuw
```

File
Diffusion_Terms.F90

Type
Subroutine

Purpose
Diffusion term in the X-direction for a quantity at the W-nodes

Reference
Section 5.6.3.1

Arguments

psiw	W-node quantity to be diffused	[psiw]
tridcfw	Tridiagonal solution matrix. The result is stored as an explicit term.	[psiw]
hdifcoefatuw	Horizontal diffusion coefficient at the UW-nodes	
klo	2 for a Neumann or a Dirichlet condition at the bottom, 3 for a Dirichlet condition at the first W-node above the bottom	
kup	nz for a Neumann condition or a Dirichlet condition at the surface, nz-1 for a Dirichlet condition at the first W-node below the surface	

Calling procedures

transport_at_W

Ydif_at_C

```
SUBROUTINE Ydif_at_C(psic,tridcfc,hdifcoefatv,novars,klo,kup)
INTEGER, INTENT(IN) :: klo, kup, novars
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz,novars) :: psic
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc,nz,4,novars) :: tridcfc
REAL, INTENT(IN), DIMENSION(ncloc,nrloc+1,nz) :: hdifcoefatv
```

File

Diffusion_Terms.F90

Type

Subroutine

Purpose

Diffusion term in the Y-direction for one or more scalar quantities at the C-nodes

Reference

Section 5.5.5.2

Arguments

psic	C-node quantity or quantities to be diffused	[psic]
tridcfc	Tridiagonal solution matrix. The result is stored as an explicit term.	[psic]

hdifcoefatv	Diffusion coefficient at the V-nodes	[m^2/s]
novars	Number of variables to be diffused	
klo	1 for a Neumann, 2 for a Dirichlet condition at the bottom	
kup	nz for a Neumann, nz-1 for a Dirichlet condition at the surface	

Calling procedures

transport_at_C_3d, transport_at_C_4d1, transport_at_C_4d2

Ydif_at_U_2d

```
SUBROUTINE Ydif_at_U_2d(psiu,ydifatu2d,hdifcoefatuv,vintflag)
LOGICAL, INTENT(INOUT) :: vintflag
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo) :: psiu
REAL, INTENT(OUT), DIMENSION(ncloc,nrloc) :: ydifatu2d
REAL, INTENT(IN), DIMENSION(ncloc+1,nrloc+1) :: hdifcoefatuv
```

File

Diffusion_Terms.F90

Type

Subroutine

Purpose

Cross-stream diffusion term for the X-component of the depth current-integrated current U at the U-nodes

Reference

Section 5.3.9

Arguments

psiu	U-node quantity to be diffused	[m^2/s]
ydifatu2d	Diffusion term (times $\Delta\tau$)	[m^2/s]
hdifcoefatuv	Horizontal diffusion coefficient at the UV-nodes	[m^2/s]
vintflag	Add the result to the fourth (2-D barotropic) part of equation (5.204) if set to .TRUE. at predictor time steps.	

Calling procedures

transport_at_U_2d

Ydif_at_U_3d

```
SUBROUTINE Ydif_at_U_3d(psiu,ydifatu3d,hdifcoefatuv,vintflag)
LOGICAL, INTENT(IN) :: vintflag
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz) :: psiu
REAL, INTENT(OUT), DIMENSION(ncloc,nrloc,nz) :: ydifatu3d
REAL, INTENT(IN), DIMENSION(ncloc+1,nrloc+1,nz) :: hdifcoefatuv
```

File

Diffusion_Terms.F90

Type

Subroutine

Purpose

Cross-stream diffusion term in the X-direction for the X-component of the current u at the U-nodes

Reference

Section 5.3.8

Arguments

psiu	U-node quantity to be diffused	[m/s]
ydifatu3d	Diffusion term (times h_3^u)	[m^2/s^2]
hdifcoefatuv	Horizontal diffusion coefficient at the UV-nodes	[m^2/s]
vintflag	Add the result to the second (3-D) part of equation (5.204) if set to .TRUE. at predictor time steps.	

Calling procedures

transport_at_U_3d

Ydif_at_V_2d

```
SUBROUTINE Ydif_at_V_2d(psiv,ydifatv2d,hdifcoefatc,vintflag)
LOGICAL, INTENT(INOUT) :: vintflag
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo) :: psiv
REAL, INTENT(OUT), DIMENSION(ncloc,nrloc) :: ydifatv2d
REAL, INTENT(IN), DIMENSION(0:ncloc,0:nrloc) :: hdifcoefatc
```

File

Diffusion_Terms.F90

Type
Subroutine

Purpose
Diffusion term in the Y-direction for the 2-D current V at the V-nodes

Reference
Section 5.3.9

Arguments

psiv	V-node quantity to be diffused	$[m^2/s]$
ydifatv2d	Diffusion term	$[m^2/s^2]$
hdifcoefatc	Horizontal diffusion coefficient at the C-nodes	$[m^2/s]$
vintflag	Add the result to the fourth (2-D barotropic) part of equation (5.205) if set to .TRUE. at predictor time steps.	

Calling procedures
transport_at_V_2d

Ydif_at_V_3d

```
SUBROUTINE Ydif_at_V_3d(psiv,ydifatv3d,hdifcoefatc,vintflag)
LOGICAL, INTENT(IN) :: vintflag
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz) :: psiv
REAL, INTENT(OUT), DIMENSION(ncloc,nrloc,nz) :: ydifatv3d
REAL, INTENT(IN), DIMENSION(0:ncloc,0:nrloc,nz) :: hdifcoefatc
```

File
Diffusion_Terms.F90

Type
Subroutine

Purpose
Along-stream diffusion term in the Y-direction for the Y-component of the current v at the V-nodes

Reference
Section 5.3.8

Arguments

psiv	V-node quantity to be diffused	[m/s]
ydifatv3d	Diffusion term (times h_3^v)	[m^2/s^2]
hdifcoefatc	Horizontal diffusion coefficient at the C-nodes	[m^2/s]
vintflag	Add the result to the second (3-D) part of equation (5.205) if set to .TRUE. at predictor time steps.	

Calling procedures
transport_at_V_3d

Ydif_at_W

```
SUBROUTINE Ydif_at_W(psiw,tridcfw,hdifcoefatvw,klo,kup)
INTEGER, INTENT(IN) :: klo, kup
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz+1)::psiw
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc,nz+1,4) :: tridcfw
REAL, INTENT(IN), DIMENSION(ncloc,nrloc+1,2:nz) :: hdifcoefatvw
```

File
Diffusion_Terms.F90

Type
Subroutine

Purpose
Diffusion term in the Y-direction for a quantity at the W-nodes

Reference
Section 5.6.3.2

Arguments

psiw	W-node quantity to be diffused	[psiw]
tridcfw	Tridiagonal solution matrix. The result is stored as an explicit term.	[psiw]
hdifcoefatvw	Horizontal diffusion coefficient at the VW-nodes	[m^2/s]
klo	2 for a Neumann or a Dirichlet condition at the bottom, 3 for a Dirichlet condition at the first W-node above the bottom	
kup	nz for a Neumann condition or a Dirichlet condition at the surface, nz-1 for a Dirichlet condition at the first W-node below the surface	

Calling procedures
transport_at_W

Zdif_at_C

```
SUBROUTINE Zdif_at_C(psic,tridcfc,vdifcoefatw,novars,ibcsur,&
                     & ibcbot,nbcs,nbcb,bcsur,bcbot,klo,kup)
INTEGER, INTENT(IN) :: ibcbot, ibcsur, klo, kup, nbcb, nbcs, novars
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz,novars) :: psic
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc,nz,4,novars) :: tridcfc
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nz+1) :: vdifcoefatw
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nbcs,novars) :: bcsur
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nbcb,novars) :: bcbot
```

File
Diffusion_Terms.F90

Type
Subroutine

Purpose
Vertical diffusion term for one or more scalar quantities at the C-nodes

Reference
Sections 5.5.5.3 and 5.5.7

Arguments

psic	C-node quantity or quantities to be diffused	[psic]
tridcfc	Tridiagonal matrix to store implicit and explicit terms	[psic]
vdifcoefatw	Vertical diffusion coefficient at the W-nodes	[m^2/s]
novars	Number of variables to be diffused	
ibcsur	Type of surface boundary condition	
	0: zero flux (Neumann) condition	
	1: prescibed flux (Neumann) condition (5.439)	
	2: Neumann condition using a transfer velocity (5.440)	
	3: Dirichlet condition at the first C-node below the surface (5.441)	

4: Dirichlet condition (5.442) at the surface

ibcbot — Type of bottom boundary condition

0: zero flux (Neumann) condition

1: prescibed flux (Neumann) condition (5.444)

2: Neumann condition using a transfer velocity (5.445)

3: Dirichlet condition (5.446) at the first C-node above the bottom

4: Dirichlet condition (5.447) at the bottom

nbcs — Last dimension of the array bcsur

nbcb — Last dimension of the array bcbot

bcsur — Data for the surface boundary condition. The third index determines the type of the data

1: prescribed surface flux or surface value [psic] or [psic m/s]

2: transfer velocity [m/s]

bcbot — Data for the bottom boundary condition. The third index determines the type of the data

1: prescribed bottom flux or bottom value [psic] or [psic m/s]

2: transfer velocity [m/s]

klo — 1 for a Neumann, 2 for a Dirichlet condition at the bottom

kup — nz for a Neumann, nz-1 for a Dirichlet condition at the surface

Calling procedures

transport_at_C_3d, transport_at_C_4d1, transport_at_C_4d2

Zdif_at_U

```
SUBROUTINE Zdif_at_U(psiu,tridcfu,vdifcoefatuw,ibcsur,ibcbot,&
                  & nbcs,nbcb,bcsur,bcbot)
INTEGER, INTENT(IN) :: ibcbot, ibcsur, nbcb, nbcs
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                         & 1-nhalo:nrloc+nhalo,nz) :: psiu
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc,nz,4) :: tridcfu
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nz+1) :: vdifcoefatuw
```

```
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nbcs) :: bcsur
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nbcb) :: bcbot
```

File

Diffusion_Terms.F90

Type

Subroutine

Purpose

Vertical diffusion term for the 3-D current u at the U-nodes

Reference

Sections 5.3.11 and 5.3.15

Arguments

psiu	U-node quantity to be diffused	[m/s]
tridcfu	Tridiagonal matrix to store implicit and explicit terms	[m/s]
vdifcoefatuw	Diffusion coefficient at the UW-nodes	[m^2/s]
ibcsur	Type of surface boundary condition	
	0: zero flux (Neumann) condition	
	1: prescibed flux (Neumann) condition (5.245)	
ibcbot	Type of bottom boundary condition	
	0: zero flux (Neumann) condition	
	1: prescibed flux (Neumann) condition	
	2: Neumann condition using a transfer velocity (see equations (5.249), (5.251), (5.252))	
nbcs	Last dimension of the array bcsur	
nbcb	Last dimension of the array bcbot	
bcsur	Data for the surface boundary condition. The third index determines the type of the data	
	1: prescribed surface flux or surface value	[m^2/s^2] or [m/s]
	2: transfer velocity	[m/s]
bcbot	Data for the bottom boundary condition. The third index defines the type of the data	

1: prescribed bottom flux or bottom value [m^2/s^2] or [m/s]
2: transfer velocity [m/s]

Calling procedures
transport_at_U_3d

Zdif_at_V

```
SUBROUTINE Zdif_at_V(psiv,tridcfv,vdifcoefatvw,ibcsur,ibcbot,&
                  & nbcs,nbcb,bcsur,bcbot)
INTEGER, INTENT(IN) :: ibcbot, ibcbot, nbcb, nbcs
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz) :: psiv
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc,nz,4) :: tridcfv
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nz+1) :: vdifcoefatvw
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nbcs) :: bcsur
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nbcb) :: bcbot
```

File
Diffusion_Terms.F90

Type
Subroutine

Purpose
Vertical diffusion term for the 3-D current v at the V-nodes

Reference
Sections 5.3.11 and 5.3.15

Arguments

psiv	V-node quantity to be diffused	[m/s]
tridcfv	Tridiagonal matrix to store implicit and explicit terms	[m/s]
vdifcoefatvw	Diffusion coefficient at the VW-nodes	[m^2/s]
ibcsur	Type of surface boundary condition	
	0: zero flux (Neumann) condition	
	1: prescibed flux (Neumann) condition (5.245)	
ibcbot	Type of bottom boundary condition	

0: zero flux (Neumann) condition

1: prescibed flux (Neumann) condition

2: Neumann condition using a transfer velocity (see equations (5.250), (5.251), (5.253))

nbcs — Last dimension of array bcsur

nbcb — Last dimension of array bcbot

bcsur — Data for the surface boundary condition. The third index determines the type of the data

1: prescribed surface flux or surface value [m^2/s^2] or [m/s]

2: transfer velocity [m/s]

bcbot — Data for the bottom boundary condition. The third index determines the type of the data

1: prescribed bottom flux or bottom value [m^2/s^2] or [m/s]

2: transfer velocity [m/s]

Calling procedures

transport_at_V_3d

Zdif_at_W

```
SUBROUTINE Zdif_at_W(psiw,tridcfw,vdifcoefatc,ibcsur,ibcbot,&
                  & bcsur,bcbot)
INTEGER, INTENT(IN) :: ibcbot, ibcsur
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                           & 1-nhalo:nrloc+nhalo,nz+1)::psiw
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc,nz+1,4) :: tridcfw
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nz) :: vdifcoefatc
REAL, INTENT(IN), DIMENSION(ncloc,nrloc) :: bcbot, bcsur
```

File

Diffusion_Terms.F90

Type

Subroutine

Purpose

Vertical diffusion term for a quantity at the W-nodes

Reference
Sections 5.6.3.3 and 5.6.6

Arguments

psiw	W-node quantity to be diffused	[psiw]
tridcfw	Tridiagonal matrix to store implicit and explicit terms	[psiw]
vdifcoefatc	Diffusion coefficient at the C-nodes	[m^2/s]
ibcsur	Type of surface boundary condition	
	0: zero flux (Neumann) condition	
	1: prescibed (Neumann) flux at the surface (see equations (5.520)–(5.522))	
	2: prescribed Neumann flux at the first C-node below the surface (5.523)	
	3: Dirichlet condition (5.524) at the surface	
	4: Dirichlet condition (5.525) at the first W-node below the surface	
ibcbot	Type of bottom boundary condition	
	0: zero flux (Neumann) condition	
	1: prescibed (Neumann) flux at the bottom (see equations (5.526)–(5.528))	
	2: prescribed Neumann flux at the first C-node above the bottom (5.529)	
	3: Dirichlet condition at the bottom (5.530)	
	4: Dirichlet condition at the first W-node above the bottom (5.531)	
bcsur	Imposed surface flux or surface value	
bcbot	Imposed bottom flux or bottom value	

Calling procedures
transport_at_W

30.9 *Grid_Arrays.F90*

Ensemble of routines for setting up the model grid, definition of pointer arrays and arrays for applying the open boundary conditions, update of the water depths.

define_global_grid

`SUBROUTINE define_global_grid`

File
Grid_Arrays.F90

Type
Subroutine

Purpose
Define the global (horizontal and vertical) domain grid

Calling procedures
initialise_model

define_local_grid

`SUBROUTINE define_local_grid`

File
Grid_Arrays.F90

Type
Subroutine

Purpose
Define local grids and bathymetry for each local sub-domain.

Calling procedures
initialise_model

depth_at_nodes

`SUBROUTINE depth_at_nodes`

File
Grid_Arrays.F90

Type
Subroutine

Purpose
Interpolate mean water depths at different nodes

Calling procedures
initialise_model

grid_arrays

`SUBROUTINE grid_arrays`

File
Grid_Arrays.F90

Type
Subroutine

Purpose
Define the arrays related to the model grid at different nodes: grid spacings in the horizontal and vertical, transformed vertical coordinates, Coriolis frequency, acceleration of gravity.

Reference
Section 10.1.2.3

Calling procedures
initialise_model

open_boundary_arrays

`SUBROUTINE open_boundary_arrays`

File
Grid_Arrays.F90

Type
Subroutine

Purpose
Allocate and define the arrays at open boundaries: locations, index mapping arrays, number of open sea and river boundaries on each local sub-domain, arrays to store information for open boundary conditions.

Calling procedures
initialise_model

pointer_arrays

`SUBROUTINE pointer_arrays`

File
Grid_Arrays.F90

Type
Subroutine

Purpose
Cell pointers arrays at the different nodes, orientation of the cell faces at open boundary locations.

Reference
Section 10.1.2.4

Calling procedures
initialise_model

read_grid

`SUBROUTINE read_grid`

File
Grid_Arrays.F90

Type
Subroutine

Purpose
Read the model grid arrays, bathymetry and open boundary locations from a file in standard COHERENS format.

Calling procedures
initialise_model

store_depths_old

`SUBROUTINE store_depths_old`

File
Grid_Arrays.F90

Type
Subroutine

Purpose
Store total dephts at the old (3-D) time level.

Calling procedures
coherens_main, initialise_physical_arrays

update_pointer_arrays

`SUBROUTINE update_pointer_arrays`

File
Grid_Arrays.F90

Type
Subroutine

Purpose
Update pointer arrays at velocity and corner nodes and set currents to zero at blocking velocity nodes.

Calling procedures
mask_function, pointer_arrays, weirs_mask

water_depths

`SUBROUTINE water_depths`

File
Grid_Arrays.F90

Type
Subroutine

Purpose
Update the total water depths at different nodes.

Called external procedures
minimum_depths

Calling procedures
default_phsics, initialise_model, initialise_model, surface_elevation, update_1dsur_data

write_grid

`SUBROUTINE write_grid`

File
Grid_Arrays.F90

Type
Subroutine

Purpose
Write the model grid arrays, bathymetry and open boundary locations to a file in standard COHERENS format.

Calling procedures
initialise_model

30.10 *Harmonic_Analysis.f90*

Routines for performing harmonic analysis and writing harmonic data.

ellips_params

```
FUNCTION ellips_params(uacoef,vacoef,ubcoef,vbcoef)
REAL, INTENT(IN) :: uacoef, ubcoef, vacoef, vbcoef
REAL, DIMENSION(7) :: ellips_params
```

File
Harmonic_Analysis.f90

Type
Function

Purpose
Parameters of tidal ellipses

Reference
Section 4.12.2

Arguments

uacoef	Coefficient of the cosine-term in the harmonic expansion for the X-component of the current vector
vacoef	Coefficient of the cosine-term in the harmonic expansion for the Y-component of the current vector
ubcoef	Coefficient of the sine-term in the harmonic expansion for the X-component of the current vector
vbcoef	Coefficient of the sine-term in the harmonic expansion for the Y-component of the current vector

Calling procedures
harmonic_analysis_data

harmonic_analysis

```
SUBROUTINE harmonic_analysis
```

File
Harmonic_Analysis.f90

Type
Subroutine

Purpose
Base program unit for performing harmonic analysis

Reference
Section 4.12.1

Called internal procedures
harmonic_analysis_data, harmonic_analysis_grid, harmonic_analysis_init, harmonic_analysis_reset, harmonic_analysis_update

Called external procedures
read_cif_params

Calling procedures
coherens_main

harmonic_analysis_data

```
SUBROUTINE harmonic_analysis_data
```

File
Harmonic_Analysis.f90

Type
Internal subroutine

Purpose
Solve the linear system of equations (4.404), evaluate residuals, amplitudes, phases and elliptic parameters and write the results to the output files where requested

Reference
: Section 4.12.1

Called internal procedures
: ellips_params

Called external procedures
: astro_params

Calling procedures
: harmonic_analysis

harmonic_analysis_grid

`SUBROUTINE harmonic_analysis_grid`

File
: *Harmonic_Analysis.f90*

Type
: Internal subroutine

Purpose
: Write the coordinates of the output grid to the data file

Calling procedures
: harmonic_analysis

harmonic_analysis_init

`SUBROUTINE harmonic_analysis_init`

File
: *Harmonic_Analysis.f90*

Type
: Internal subroutine

Purpose
: Define the arrays for the setup of harmonic analysis and data output. Write the metadata to the data file. Set up the matrices for solving the linear systems (4.404)–(4.405)

Reference
: Sections 20.3.1 and 20.3.2

Called external procedures
read_cif_params, usrdef_anal_freqs, usrdef_anal_params

Calling procedures
harmonic_analysis

harmonic_analysis_reset

`SUBROUTINE harmonic_analysis_reset`

File
Harmonic_Analysis.f90

Type
Internal subroutine

Purpose
Initialise buffers for storing temporary data

Calling procedures
harmonic_analysis

harmonic_analysis_update

`SUBROUTINE harmonic_analysis_update`

File
Harmonic_Analysis.f90

Type
Internal subroutine

Purpose
Update the sums R_m and S_m defined by equations(4.409) and (4.410).

Reference
Section 4.12.1

Called external procedures
usrdef_anal0d_vals, usrdef_anal2d_vals, usrdef_anal3d_vals

Calling procedures
harmonic_analysis

30.11 *Hydrodynamic_Equations.F90*

Update all hydrodynamic parameters (2-D transports, water elevations, 3-D currents) by solving the 2-D and 3-D momentum and continuity equations.

correct_free_surf

`SUBROUTINE correct_free_surface`

File
: *Hydrodynamic_Equations.F90*

Type
: Subroutine

Purpose
: Solve the elliptic equation for the free surface correction ζ', apply the corrector step and open boundary conditions for the 2-D current.

Reference
: Section 5.3.2

Called external procedures
: open_boundary_conds_impl, open_boundary_conds_2d, update_2dobc_data, water_depths

Calling procedures
: hydrodynamic_equations

current_corr

`SUBROUTINE current_corr`

File
: *Hydrodynamic_Equations.F90*

Type
: Subroutine

Purpose
: Solve the 3-D momentum equations at the (explicit or implicit) corrector step.

Reference
: Sections 5.3.1.3 and 5.3.2.

Called external procedures
current_corr_1d, define_profobc_spec, open_boundary_conds_3d, relaxation_at_U, relaxation_at_V, update_nest_data_3d, update_profobc_data

Calling procedures
hydrodynamic_equations, initialise_model

current_corr_1d

```
SUBROUTINE current_corr_1d
```

File
Hydrodynamic_Equations.F90

Type
Subroutine

Purpose
Evaluate the depth-mean and depth-integrated currents in case of a 1-D water column application.

Calling procedures
current_corr

current_pred

```
SUBROUTINE current_pred
```

File
Hydrodynamic_Equations.F90

Type
Subroutine

Purpose
Solve the 3-D momentum equations (4.61) and (4.62) for u and v at the (explicit or implicit) predictor step.

Reference
Sections 4.2.1.2, 5.3.1.1 and 5.3.2.

Called external procedures
transport_at_U_3d, transport_at_V_3d, update_1dsur_data

Calling procedures
hydrodynamic_equations

current_2d

`SUBROUTINE current_2d`

File
Hydrodynamic_Equations.F90

Type
Subroutine

Purpose
Solve the 2-D momentum and continuity equations (4.85)–(4.87) for ζ, U, V.

Reference
Sections 4.2.2, 5.3.1.2 and 5.3.3.2

Called external procedures
astronomical_tides, open_boundary_conds_2d, surface_elevation, transport_at_U_2d, transport_at_V_2d, update_2dobc_data

Calling procedures
hydrodynamic_equations

hydrodynamic_equations

`SUBROUTINE hydrodynamic_equations`

File
Hydrodynamic_Equations.F90

Type
Subroutine

Purpose
Base program unit for solving the hydrodynamic equations

Called external procedures
bottom_stress, correct_free_surf, current_corr, current_pred, current_2d, drying_factor, physical_vertical_current, surface_stress, transf_vertical_current, update_nest_data_2d, update_nest_data_3d

Calling procedures
coherens_main

physical_vertical_current

`SUBROUTINE physical_vertical_current`

File
Hydrodynamic_Equations.F90

Type
Subroutine

Purpose
Update the physical vertical current w using the discretised equation (5.29).

Reference
Sections 4.2.1.2 and 5.3.1.4

Calling procedures
hydrodynamic_equations

shear_frequency

`SUBROUTINE shear_frequency`

File
Hydrodynamic_Equations.F90

Type
Subroutine

Purpose
Calculate the squared shear frequency M^2 on the model grid using the algorithm given in Section 5.3.12.2.

Reference
Section 5.3.12.2

Calling procedures
vertical_diff_coefs

surface_elevation

`SUBROUTINE surface_elevation`

File
Hydrodynamic_Equations.F90

Type
Subroutine

Purpose
Solve the 2-D continuity equation (4.85) for the surface elevation ζ using either the discretised form (5.12) for the explicit or (5.36) for the implicit case.

Reference
Sections 4.2.2 and 5.3.1.2

Called external procedures
water_depths

Calling procedures
current_2d

transf_vertical_current

`SUBROUTINE transf_vertical_current`

File
Hydrodynamic_Equations.F90

Type
Subroutine

Purpose
Solve the baroclinic continuity equation (4.102) for the (transformed) vertical current ω using the discretised form (5.28).

Reference
Sections 4.2.2 and 5.3.1.4

Calling procedures
hydrodynamic_equations

transf_vertical_fall

`SUBROUTINE transf_vertical_fall`

File
Hydrodynamic_Equations.F90

Type
Subroutine

Purpose
Add settling correction to the advective current for scalars.

Reference
Equations (7.117)–(7.118)

Calling procedures
transport_at_C_4d1, transport_at_C_4d2

30.12 *Inundation_Schemes.f90*

Routines used by the drying/wetting and flooding algorithms.

drying_factor

`SUBROUTINE drying_factor`

File
Inundation_Schemes.f90

Type
Subroutine

Purpose
Update the α-factor used by the drying/wetting algorithm.

Reference
Section 5.4.1 and equation (5.367)

Calling procedures
surface_elevation

mask_function

`SUBROUTINE mask_function`

File
Inundation_Schemes.f90

Type
Subroutine

Purpose
Evaluate the mask criteria for the flooding scheme, reset the pointer arrays and set the currents to zero at blocked velocity nodes.

Reference
Section 5.4.2 and equations (5.372)–(5.383)

Calling procedures
coherens_main

minimum_depths

`SUBROUTINE minimum_depths`

File
Inundation_Schemes.f90

Type
Subroutine

Purpose
Evaluate the minimum depth criteria (5.368)–(5.370).

Reference
Section 5.4.1

Calling procedures
water_depths

30.13 *Model_Finalisation.f90*

Perform all actions used for finalisation of a simulation (writing of initial conditions, timer report, resetting of all model parameters to their defaults, deallocation of all allocated arrays, closing of all file connections, closing the program after the last simulation).

coherens_end

`SUBROUTINE coherens_end`

File
Model_Finalisation.f90

Type
Subroutine

Purpose
Finalise the program after execution of all simulations. This is the last routine called by the program after closing the *defruns* file.

Reference
Section 12.2.5

Calling procedures
coherens_main

simulation_end

`SUBROUTINE simulation_end`

File
Model_Finalisation.f90

Type
Subroutine

Purpose
Finalise the current simulation before starting an eventual new one.

Reference
Section 12.2.5

Called external procedures
deallocate_mod_arrays, deallocate_sed_arrays, timer_report

Calling procedures
coherens_main

timer_report

`SUBROUTINE timer_report`

File
Model_Finalisation.f90

Type
Subroutine

Purpose
Write the timer report file.

Reference
Section 9.3.4

Calling procedures
simulation_end

write_phsics

`SUBROUTINE write_phsics`

File
Model_Finalisation.f90

Type
Subroutine

Reference
Table 9.3

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

Calling procedures
coherens_main

30.14 *Model_Initialisation.F90*

Start-up of the program and initialisation of a simulation.

coherens_start

`SUBROUTINE coherens_start`

File
Model_Initialisation.F90

Type
Subroutine

Purpose
Start-up procedure of the program beforing execution of the first simulation. This is the first routine called by the main program before opening the *defruns* file.

Calling procedures
coherens_main

default_grid

`SUBROUTINE default_grid`

File
Model_Initialisation.F90

Type
Subroutine

Purpose
Set the default bathymetry using uniform water depths.

Calling procedures
initialise_model

default_phsics

`SUBROUTINE default_phsics`

File
Model_Initialisation.F90

Type
Subroutine

Reference
Section 4.11

Purpose
Set the default physical initial conditions.

Called external procedures
init_turbulence, mixing_length, mixing_length, zlmix_to_dissip

Calling procedures
initialise_model

define_phsics

`SUBROUTINE define_phsics`

File
Model_Initialisation.F90

Type
Subroutine

Reference
Section 4.11

Purpose
(Re)define initial conditions.

Called external procedures
exchange_phsics, read_phsics, usrdef_phsics

Calling procedures
coherens_main, initialise_model

exchange_phsics

`SUBROUTINE exchange_phsics`

File
Model_Initialisation.F90

Type
Subroutine

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

Calling procedures
initialise_model

initialise_model

```
SUBROUTINE initialise_model
```

File
Model_Initialisation.F90

Type
Subroutine

Purpose
Ensemble of routine calls for initialisation of a simulation

Reference
Section 12.2.2

Called external procedures
allocate_global_grid, allocate_mod_arrays, allocate_sed_arrays, astronomical_tides, baroclinic_gradient, bottom_stress, current_corr, default_grid, default_phsics, define_global_grid, define_local_grid, define_phsics, define_rlxobc_spec, depth_at_nodes, domain_decomposition, equation_of_state, exchange_phsics, grid_arrays, harmonical_analysis, heat_flux, horizontal_diff_coefs, initialise_physical_arrays, initialise_sediment_arrays, meteo_input, nest_locations, open_boundary_arrays, pointer_arrays, read_cif_params, read_grid, read_sedics, read_sed_spec, relaxation_weights, salinity_equation, sediment_equation, set_procnums, set_workers_comm, surface_stress, temperature_equation, time_averages, time_series, vertical_diff_coefs, update_nest_data_2d, update_1dsur_data, update_2dobc_data, usrdef_grid, usrdef_mod_params, usrdef_output, usrdef_sedics, usrdef_sed_params, usrdef_sed_spec, water_depths, wave_input, write_cif_vars_mod, write_cif_vars_sed, write_grid, write_phsics, write_sedics, write_sed_spec

Calling procedures
coherens_main

initialise_physical_arrays

`SUBROUTINE initialise_physical_arrays`

File
Model_Initialisation.F90

Type
Subroutine

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

Called external procedures
physical_vertical_current, store_depths_old, transf_vertical_current

Calling procedures
initialise_model

read_phsics

`SUBROUTINE read_phsics`

File
Model_Initialisation.F90

Type
Subroutine

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

Reference
Table 9.3

Calling procedures
initialise_model

simulation_start

```
SUBROUTINE simulation_start(next_simul)
LOGICAL, INTENT(OUT) :: next_simul
```

File
Model_Initialisation.F90

Type
Subroutine

Purpose
Read the title of the next simulation from the *defruns* file and define the parameters for monitoring.

Arguments

next_simul	.TRUE. to start the next simulation, .FALSE. to exit the program

Called external procedures
read_cif_params, usrdef_init_params

Calling procedures
coherens_main

30.15 *Model_Parameters.f90*

Read/write model setup parameters from/to a central input file (CIF).

assign_cif_vars_mod

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

File
Model_Parameters.F90

Type
Subroutine

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

Reference
: Section 9.4

Arguments

cname	Variable name(s)
cvals	Data values in string format
numvars	Number of model variables

Calling procedures
: read_cif_params

read_cif_params

```
SUBROUTINE read_cif_params(iddesc,fass)
LOGICAL, INTENT(IN) :: fass
INTEGER, INTENT(IN) :: iddesc
```

File
: *Model_Parameters.F90*

Type
: Subroutine

Purpose
: General routines for reading a CIF. The actual data conversion is made by calling read_cif_line and assign_cif_vars.

Reference
: Section 9.4

Arguments

fass	File will be read with/without data assignment if .TRUE./.FALSE.
iddesc	Key id of the CIF

Called external procedures
: assign_cif_vars_mod, assign_cif_vars_sed

Calling procedures
harmonic_analysis, harmonic_analysis_init, initialise_model, simulation_start, time_averages, time_averages_init, time_series, time_series_init

write_cif_vars_mod

```
SUBROUTINE write_cif_vars_mod
INTEGER, INTENT(IN) :: iddesc
```

File
Model_Parameters.f90

Type
Subroutine

Purpose
Write the CIF for physical model setup.

Reference
Section 9.4

Arguments
None

Calling procedures
initialise_model

30.16 *Nested_Grids.F90*

Ensemble of routines for the definition of nested sub-grids, interpolation to sub-grid locations and writing to the appropriate output files.

define_nstgrd_locs

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

File
Nested_Grids.F90

Type
Subroutine

Purpose
Define the open boundary locations of all nested sub-grids. Absolute coordinates are converted into relative ones if necessary.

Reference
Sections 10.5 and 17.3

Arguments

iset	Index of the nested sub-grid
nhdat	Number of sub-grid open boundary locations
nzdat	Number of vertical levels
nonodes	The number of nodes for which relative coordinates need to be defined. Value is either 1 or 2.
hnests	Relative coordinates of the sub-grid open boundary locations
zcoord	Z-coordinates of the sub-grid open boundary locations
cnode	Nodal type of the open boundary locations ('C', 'U', 'V')

Called external procedures
read_nstgrd_abs, read_nstgrd_rel, usrdef_nstgrd_abs, usrdef_nstgrd_rel, write_nstgrd_abs, write_nstgrd_rel

Calling procedures
nest_locations

define_nstgrd_spec

```
SUBROUTINE define_nstgrd_spec
```

File
Nested_Grids.F90

Type
Subroutine

Purpose
General specifications of all sub-grid nests (number of sub-grid open boundary locations, number of vertical levels, type of required data)

Reference
Sections 10.5 and 17.3.1

Called external procedures
read_nstgrd_spec, usrdef_nstgrd_spec, write_nstgrd_spec

Calling procedures
nest_locations

nest_locations

```
SUBROUTINE nest_locations
```

File
Nested_Grids.F90

Type
Subroutine

Purpose
Define the positions (in relative coordinates) of all open boundary locations on each sub-grid nest and a series of associated arrays.

Reference
Sections 10.5

Called external procedures
define_nstgrd_locs, define_nstgrd_spec

Calling procedures
initialise_model

read_nstgrd_abs

```
SUBROUTINE read_nstgrd_abs(iset,nhdat,nzdat,xcoord,ycoord,zcoord)
INTEGER, INTENT(IN) :: iset, nhdat, nzdat
REAL, INTENT(OUT), DIMENSION(nhdat) :: xcoord, ycoord
REAL, INTENT(OUT), DIMENSION(nhdat,nzdat) :: zcoord
```

File
Nested_Grids.F90

Type
Subroutine

Purpose
Read the absolute coordinates of the open boundary locations on a sub-grid from a file in standard COHERENS format.

Reference
Section 17.3.2

Arguments

iset	Index of the nested sub-grid
nhdat	Number of open boundary locations on the sub-grid
nzdat	Number of vertical levels
xcoord	X-coordinates of the sub-grid data points [meters or longitude]
ycoord	Y-coordinates of the sub-grid data points [meters or latitude]
zcoord	Z-coordinates of the sub-grid locations, defined as the positive distance to the mean water level (only when nzdat>0) [m]

Calling procedures
define_nstgrd_locs

read_nstgrd_rel

```
SUBROUTINE read_nstgrd_rel(iset,nhdat,nzdat,nonodes,hnests,zcoord)
INTEGER, INTENT(IN) :: iset, nhdat, nonodes, nzdat
REAL, INTENT(OUT), DIMENSION(nhdat,nzdat) :: zcoord
TYPE (HRelativeCoords), INTENT(OUT),&
                     & DIMENSION(nhdat,nonodes) :: hnests
```

File
Nested_Grids.F90

Type
Subroutine

Purpose
Read the relative coordinates of the open boundary locations on a sub-grid from a file in standard COHERENS format.

Reference
Section 17.3.3

Arguments

iset	Index of the nested sub-grid
nhdat	Number of open boundary locations on the sub-grid
nzdat	Number of vertical levels
nonodes	Number of nodes for which relative coordinates need to be provided. Its value equals 1 or 2 depending on the value of cnode.
hnests	Relative coordinates of the sub-grid data points
zcoord	Z-coordinates of the sub-grid locations, defined as the positive distance to the mean water level (only when nzdat>0) [m]

Calling procedures
define_nstgrd_locs

read_nstgrd_spec

```
SUBROUTINE read_nstgrd_spec
```

File
Nested_Grids.F90

Type
Subroutine

Purpose
Read the general specifications of all sub-grid nests (number of sub-grid open boundary locations, number of vertical levels, type of required data) from a file in standard COHERENS format.

Reference
Section 17.3.1

Calling procedures
define_nstgrd_spec

update_nest_data_prof

```
SUBROUTINE update_nest_data_prof(psic,novars,novarsnst,instvars,iddesc)
INTEGER, INTENT(IN) :: iddesc, novars
INTEGER, INTENT(IN), DIMENSION(nonestsets) :: novarsnst
INTEGER, INTENT(IN), DIMENSION(novars,nonestsets) :: instvars
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                          & 1-nhalo:nrloc+nhalo,nz,novars) :: psic
```

File
Nested_Grids.F90

Type
Subroutine

Purpose
Interpolate 3-D scalar profile data from the model grid to the appropriate open boundaries of each nested sub-grid and write the interpolated data to the appropriate file(s).

Arguments

psic	C-node scalar quantity used for the nesting
novars	Total number of "state" variables (currently 1)
novarsnst	Total number of "state" variables to be nested per sub-grid (currentlty 1)
instvars	Indices of the variables to be nested (currently 1)
iddesc	Key id of the corresponding output file(s)

Called external procedures
write_nest_data_prof

Calling procedures
salinity_equation, temperature_equation

update_nest_data_2d

```
SUBROUTINE update_nest_data_2d
```

File
Nested_Grids.F90

Type
Subroutine

Purpose

Interpolate 2-D model data (transports and/or elevations) from the model grid to the appropriate open boundaries of each nested sub-grid and write the interpolated data to the appropriate file(s).

Calling procedures

current_2d, initialise_model

update_nest_data_3d

```
SUBROUTINE update_nest_data_3d
```

File

Nested_Grids.F90

Type

Subroutine

Purpose

Interpolate 3-D baroclinic current data from the model grid to the appropriate open boundaries of each nested sub-grid and write the interpolated data to the appropriate file(s).

Called external procedures

write_nest_data_prof

Calling procedures

current_corr

write_nest_data_prof

```
SUBROUTINE write_nest_data_prof(iddesc,iset,outvals,maxstats,nzdat,&
                              & nonst,nhdatloc,nostatsglb,nhdatprocs,&
                              & indexprocs)
INTEGER, INTENT(IN) :: iddesc, iset, maxstats, nhdatloc, nonst, nostatsglb, &
                     & nzdat
INTEGER, INTENT(IN), DIMENSION(nprocs) :: nhdatprocs
INTEGER, INTENT(IN), DIMENSION(nprocs,maxstats) :: indexprocs
REAL, INTENT(IN), DIMENSION(nhdatloc,nzdat,nonst) :: outvals
```

File

Nested_Grids.F90

Type
Subroutine

Purpose
Write 3-D scalar data interpolated at a specific sub-grid to a file in standard COHERENS format.

Arguments

iddesc	Key id of the corresponding output file(s)
iset	Index (file number) of the nested sub-grid
outvals	Output data
maxstats	Second dimension of the array indexprocs
nzdat	Number of data points in the vertical
nonst	Number of "state" variables to be nested (currently 1)
nhdatloc	Number of output locations in the horizontal on the local domain grid
nostatsglb	Number of output locations on the global grid
nhdatprocs	Number of output locations per process
indexprocs	Index mapping array (see Section 11.5)

Calling procedures
update_nest_data_prof

write_nest_data_2d

```
SUBROUTINE write_nest_data_2d(iset,ivar,outvals,maxstats,nhdatloc,&
                            & nostatsglb,nhdatprocs,indexprocs)
INTEGER, INTENT(IN) :: iset, ivar, maxstats, nhdatloc, nostatsglb
INTEGER, INTENT(IN), DIMENSION(nprocs) :: nhdatprocs
INTEGER, INTENT(IN), DIMENSION(nprocs,maxstats) :: indexprocs
REAL, INTENT(IN), DIMENSION(nhdatloc) :: outvals
```

File
Nested_Grids.F90

Type
Subroutine

Purpose
Write the 2-D data interpolated at a specific sub-grid to a file in standard COHERENS format.

Arguments

iset	Index (file number) of the nested sub-grid
ivar	(netCDF) Id of the variable within the file
outvals	Output data
maxstats	Second dimension of the array indexprocs
nhdatloc	Number of output locations in the horizontal on the local domain grid
nostatsglb	Number of output locations on the global grid
nhdatprocs	Number of output locations per process
indexprocs	Index mapping array (see Section 11.5)

Calling procedures

update_nest_data_2d

write_nest_data_3d

```
SUBROUTINE
write_nest_data_prof(iset,outvals,maxstats,nzdat,nhdatloc,&
                  & nostatsglb,nhdatprocs,indexprocs)
INTEGER, INTENT(IN) :: iset, maxstats, nhdatloc, nostatsglb, nzdat
INTEGER, INTENT(IN), DIMENSION(nprocs) :: nhdatprocs
INTEGER, INTENT(IN), DIMENSION(nprocs,maxstats) :: indexprocs
REAL, INTENT(IN), DIMENSION(nhdatloc,nzdat,nonst) :: outvals
```

File

Nested_Grids.F90

Type

Subroutine

Purpose

Write 3-D baroclinic current data interpolated at a specific sub-grid to a file in standard COHERENS format.

Arguments

iset	Index (file number) of the nested sub-grid
outvals	Output data
maxstats	Second dimension of the array indexprocs
nzdat	Number of data points in the vertical

nhdatloc	Number of output locations in the horizontal on the local domain grid
nostatsglb	Number of output locations on the global grid
nhdatprocs	Number of output locations per process
indexprocs	Index mapping array (see Section 11.5)

Calling procedures
update_nest_data_3d

write_nstgrd_abs

```
SUBROUTINE write_nstgrd_abs(iset,nhdat,nzdat,xcoord,ycoord,zcoord)
INTEGER, INTENT(IN) :: iset, nhdat, nzdat
REAL, INTENT(IN), DIMENSION(nhdat) :: xcoord, ycoord
REAL, INTENT(IN), DIMENSION(nhdat,nzdat) :: zcoord
```

File
Nested_Grids.F90

Type
Subroutine

Purpose
Write the absolute coordinates of the open boundary locations on a sub-grid to a file in standard COHERENS format.

Reference
Section 17.3.2

Arguments

iset	Index of the nested sub-grid
nhdat	Number of open boundary locations on the sub-grid
nzdat	Number of vertical levels
xcoord	X-coordinates of the sub-grid data points [meters or longitude]
ycoord	Y-coordinates of the sub-grid data points [meters or latitude]
zcoord	Z-coordinates of the sub-grid locations, defined as the positive distance to the mean water level (only when nzdat>0) [m]

Calling procedures
define_nstgrd_locs

write_nstgrd_rel

```
SUBROUTINE write_nstgrd_rel(iset,nhdat,nzdat,nonodes,hnests,zcoord)
INTEGER, INTENT(IN) :: iset, nhdat, nonodes, nzdat
REAL, INTENT(IN), DIMENSION(nhdat,nzdat) :: zcoord
TYPE (HRelativeCoords), INTENT(OUT),&
                      & DIMENSION(nhdat,nonodes) :: hnests
```

File
Nested_Grids.F90

Type
Subroutine

Purpose
Write the relative coordinates of the open boundary locations on a sub-grid to a file in standard COHERENS format.

Reference
Section 17.3.3

Arguments

iset	Index of the nested sub-grid
nhdat	Number of open boundary locations on the sub-grid
nzdat	Number of vertical levels
nonodes	Number of nodes for which relative coordinates need to be provided. Its value equals 1 or 2 depending on the value of cnode.
hnests	Relative coordinates of the sub-grid data points
zcoord	Z-coordinates of the sub-grid locations, defined as the positive distance to the mean water level (only when nzdat>0) [m]

Calling procedures
define_nstgrd_locs

write_nstgrd_spec

`SUBROUTINE write_nstgrd_spec`

File
Nested_Grids.F90

Type
Subroutine

Purpose
Write the general specifications of all sub-grid nests (number of sub-grid open boundary locations, number of vertical levels, type of required data) to a file in standard COHERENS format.

Reference
Section 17.3.1

Calling procedures
define_nstgrd_spec

30.17 *Open_Boundary_Conditions.f90*

Apply open boundary conditions for the 2-D and 3-D case.

open_boundary_conds_impl

`SUBROUTINE open_boundary_conds_impl`

File
Open_Boundary_Conditions.f90

Type
Subroutine

Purpose
Add the open boundary contributions to the coefficients of the elliptic equation (5.37) for the free surface correction. Apply the open boundary conditions for the 2-D mode.

Reference
Section 5.3.2

Calling procedures
correct_free_surf

open_boundary_conds_prof

```
SUBROUTINE open_boundary_conds_prof(psi,psiobu,psiobv,obcdata,&
                                    & obcpsiatu,obcpsiatv,itypobu,&
                                    & itypobv,iprofobu,iprofobv,&
                                    & noprofs,novars)
INTEGER, INTENT(IN) :: noprofs, novars
INTEGER, INTENT(IN), DIMENSION(nobu) :: itypobu
INTEGER, INTENT(IN), DIMENSION(nobv) :: itypobv
INTEGER, INTENT(INOUT), DIMENSION(nobu,novars) :: iprofobu
INTEGER, INTENT(INOUT), DIMENSION(nobv,novars) :: iprofobv
REAL, INTENT(IN), DIMENSION(1-nhalo:ncloc+nhalo,&
                           & 1-nhalo:nrloc+nhalo,nz,novars) :: psi
REAL, INTENT(INOUT), DIMENSION(nobu,nz,novars) :: psiobu
REAL, INTENT(INOUT), DIMENSION(nobv,nz,novars) :: psiobv
REAL, INTENT(IN), DIMENSION(0:noprofs,nz,novars) :: obcdata
REAL, INTENT(INOUT), DIMENSION(nobu,nz,novars,0:2) :: obcpsiatu
REAL, INTENT(INOUT), DIMENSION(nobv,nz,novars,0:2) :: obcpsiatv
```

File

Open_Boundary_Conditions.f90

Type

Subroutine

Purpose

Apply the open open boundary conditions for a 3-D scalar C-node quantity.

Reference

Section 4.10.2.1, 5.5.7.3 and 16.2.1

Arguments

psi	C-node scalar quantity defined on the 3-D model grid
psiobu	Returned values of the scalar quantity at West/East open boundaries. The data are flagged if a zero gradient condition is applied at the open boundary location.
psiobv	Returned values of the scalar quantity at South/North open boundaries. The data are flagged if a zero gradient condition is applied at the open boundary location.
obcdata	Time-interpolated external data profiles. A zero gradient condition is substituted for data which are flagged.

obcpsiatu	Storage array in case the open boundary condition at the U-nodes uses values from previous time steps
obcpsiatv	Storage array in case the open boundary condition at the V-nodes uses values from previous time steps
itypobu	Type of open boundary condition at the U-nodes 0: External data profile or zero gradient condition 1: Radiation condition using internal wave speed 2: Orlanski type of radiation condition
itypobv	Type of open boundary condition at the V-nodes 0: External data profile or zero gradient condition 1: Radiation condition using internal wave speed 2: Orlanski type of radiation condition
iprofobu	Profile number at the U-nodes for each data variable. If set to zero, the type of condition is determined by the value of itypobu.
iprofobv	Profile number at the V-nodes for each data variable. If set to zero, the type of condition is determined by the value of itypobv.
noprofs	Number of data profiles
novars	Number of C-node variables (currently 1)

Calling procedures
salinity_equation, temperature_equation

open_boundary_conds_2d

`SUBROUTINE open_boundary_conds_2d`

File
Open_Boundary_Conditions.f90

Type
Subroutine

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

Reference
Sections 4.10.1, 5.3.16.1 and 16.1

Calling procedures
correct_free_surf, current_2d

open_boundary_conds_3d

```
SUBROUTINE open_boundary_conds_3d(psiobu,psiobv,obcdata,itypobu,itypobv,&
                                & iprofobu,iprofobv,noprofs)
INTEGER, INTENT(IN) :: noprofs
INTEGER, INTENT(IN), DIMENSION(nobu) :: itypobu, iprofobu
INTEGER, INTENT(IN), DIMENSION(nobv) :: itypobv, iprofobv
REAL, INTENT(INOUT), DIMENSION(nobu,nz) :: psiobu
REAL, INTENT(INOUT), DIMENSION(nobv,nz) :: psiobv
REAL, INTENT(IN), DIMENSION(0:noprofs,nz) :: obcdata
```

File
Open_Boundary_Conditions.f90

Type
Subroutine

Purpose
Apply the open open boundary conditions for the 3-D baroclinic current.

Reference
Sections 4.10.2.1, 5.3.17.1 and 16.2.1

Arguments

psiobu	Returned values of the baroclinic current at West/East open boundaries. The data are flagged if a zero gradient condition is applied without external data.
psiobv	Returned values of the baroclinic current at South/North open boundaries. The data are flagged if a zero gradient condition is applied without external data.
obcdata	Time-interpolated external data profiles. A zero gradient condition is substituted for data which are flagged.
itypobu	Type of open boundary condition at the U-nodes 0: External data profile or first order zero gradient condition 1: Second order zero gradient condition

2: Local solution

3: Radiation condition using internal wave speed

4: Orlanski type of radiation condition

itypobv — Type of open boundary condition at the V-nodes

0: External data profile or first order zero gradient condition

1: Second order zero gradient condition

2: Local solution

3: Radiation condition using internal wave speed

4: Orlanski type of radiation condition

iprofobu — Profile number at the U-nodes. If set to zero, the type of condition is determined by the value of itypobu.

iprofobv — Profile number at the V-nodes. If set to zero, the type of condition is determined by the value of itypobv.

noprofs — Number of data profiles

Calling procedures

current_corr

30.18 *Open_Boundary_Data_Prof.f90*

Open boundary profile data for baroclinic currents and 3-D scalars.

define_profobc_data

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

File

Open_Boundary_Data_Prof.f90

Type

Subroutine

Purpose

Define open boundary profiles.

Reference
Section 16.2.2

Arguments

iddesc	Key id of the corresponding forcing data file
ifil	File index of the data file
ciodatetime	Returned date and time of the data
psiprofdat	Returned profile data
numprofs	Number of profiles in the data file

Called external procedures
read_profobc_data, usrdef_profobc_data, write_profobc_data

Calling procedures
update_profobc_data

define_profobc_spec

```
SUBROUTINE define_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(INOUT), DIMENSION(nobu) :: itypobu
INTEGER, INTENT(INOUT), DIMENSION(nobv) :: itypobv
INTEGER, INTENT(INOUT), DIMENSION(nobu,novars) :: iprofobu
INTEGER, INTENT(INOUT), DIMENSION(nobv,novars) :: iprofobv
INTEGER, INTENT(INOUT), DIMENSION(novars*(nobu+nobv),2:nofiles) :: indexprof
INTEGER, INTENT(INOUT), DIMENSION(novars*(nobu+nobv),2:nofiles) :: indexvar
INTEGER, INTENT(INOUT), DIMENSION(norlxzones) :: iprofrlx
```

File
Open_Boundary_Data_Prof.f90

Type
Subroutine

Purpose
Provide the information needed for applying the open boundary conditions for a 3-D quantity (baroclinic current, C-node scalar): type of condition, profile number of each open boundary point, number of

profiles in each data file and index mapping arrays used to locate the profiles and variables within each data file.

Reference

Sections 12.2.4 and 16.2.1

Arguments

iddesc	Key id of the corresponding forcing data file
itypobu	Type of open boundary condition at the U-nodes (see Section 16.2.1.1)
itypobv	Type of open boundary condition at the V-nodes (see Section 16.2.1.1)
iprofobu	Profile number at the U-nodes for each data variable. If set to zero, the type of condition is determined by the value of itypobu.
iprofobv	Profile number at the V-nodes for each data variable. If set to zero, the type of condition is determined by the value of itypobv.
iprofrlx	Disables/enables relaxation in "relaxation" areas.
noprofsd	Number of profiles per data file
indexprof	Mapping array of the profile numbers in the data files to the profile numbers assigned to the open boundaries. The physical size of the first dimension equals the number of profiles in a data file.
indexvar	Defines the variable number of the profiles in a data file. The physical size of the first dimension equals the number of profiles in a data file.
novars	Total number of (scalar) variables. Value in the current implementation is 1.
nofiles	Number of data files (minus 1)

Called external procedures

read_profobc_spec, usrdef_profobc_spec, write_profobc_spec

Calling procedures

current_corr, salinity_equation, temperature_equation

read_profobc_data

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

File
Open_Boundary_Data_Prof.f90

Type
Subroutine

Purpose
Read the open boundary profile data in standard COHERENS format.

Reference Section 16.2.2

Arguments

iddesc	Key id of the corresponding forcing data file
ifil	File index of the data file
ciodatetime	Returned date and time of the data
psiprofdat	Returned profile data
numprofs	Number of profiles in the data file

Calling procedures
define_profobc_data

read_profobc_spec

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

File

Open_Boundary_Data_Prof.f90

Type

Subroutine

Purpose

Read the information needed for applying the open boundary conditions for a 3-D quantity (baroclinic current, C-node scalar) from a file in standard COHERENS format: type of condition, profile number of each open boundary point, number of profiles in each data file and index mapping array used to locate the profiles and variables within each data file.

Reference

Sections 12.2.4 and 16.2.1

Arguments

iddesc	Key id of the corresponding forcing data file
itypobu	Type of open boundary condition at the U-nodes (see Section 16.2.1.1)
itypobv	Type of open boundary condition at the V-nodes (see Section 16.2.1.1)
iprofobu	Profile number at the U-nodes for each data variable. If set to zero, the type of condition is determined by the value of itypobu.
iprofobv	Profile number at the V-nodes for each data variable. If set to zero, the type of condition is determined by the value of itypobv.
iprofrlx	Disables/enables relaxation in "relaxation" areas.
noprofsd	Number of profiles per data file
indexprof	Mapping array of the profile numbers in the data files to the profile numbers assigned to the open boundaries. The physical size of the first dimension equals the number of profiles in a data file.
indexvar	Defines the variable number of the profiles in a data file. The physical size of the first dimension equals the number of profiles in a data file.
novars	Total number of (scalar) variables. Value is 1 in the current implementation.

nofiles Number of data files (minus 1)

Calling procedures
define_profobc_spec

update_profobc_data

```
SUBROUTINE update_profobc_data(profdata,obcdata,noprofsd,indexprof,&
                            & indexvar,maxprofs,noprofs,novars,&
                            & nofiles,nosecsdat,iddesc)
INTEGER, INTENT(IN) :: iddesc, maxprofs, nofiles, noprofs, novars
INTEGER, INTENT(IN) , DIMENSION(2:nofiles) :: noprofsd
INTEGER, INTENT(IN), DIMENSION(maxprofs,2:nofiles) :: indexprof
INTEGER, INTENT(IN), DIMENSION(maxprofs,2:nofiles) :: indexvar
INTEGER (KIND=kndilong), INTENT(INOUT),&
                       & DIMENSION(2:nofiles,2) :: nosecsdat
REAL, INTENT(INOUT), DIMENSION(maxprofs,nz,2:nofiles,2) :: profdata
REAL, INTENT(INOUT), DIMENSION(0:noprofs,nz,novars) :: obcdata
```

File
Open_Boundary_Data_Prof.f90

Type
Subroutine

Purpose
Update the open boundary profile data by reading new data (if needed). Interpolate at the current simulation time (if requested).

Reference
Section 12.2.4

Arguments

profdata	Profile data at the latest time earlier than the current date/time and at the earliest time later than the current date/time
obcdata	Returned profile arrays interpolated at the current simulation time
noprofsd	Number of profiles per data file
indexprof	Mapping array of the profile numbers in the data files to the profile numbers assigned to the open boundaries. The

	physical size of the first dimension equals the number of profiles in a data file.
indexvar	Defines the variable number of the profiles in a data file. The physical size of the first dimension equals the number of profiles in a data file.
maxprofs	Maximum number of profiles in any data file
noprofs	Total number of profiles
novars	Total number of variables. Value is 1 in the current implementation.
nofiles	Number of data files (minus 1)
nosecsdat	Number of seconds since the start of the simulation and the latest data time earlier than the current time and the earliest data time later than the current time
iddesc	Key id of the corresponding forcing data file

Called external procedures
define_profobc_data

Calling procedures
current_corr, salinity_equation, temperature_equation

write_profobc_data

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

File
Open_Boundary_Data_Prof.f90

Type
Subroutine

Purpose
Write the open boundary profile data in standard COHERENS format.

Reference Section 16.2.2

Arguments

iddesc	Key id of the corresponding forcing data file

ifil	File index of the data file
ciodatetime	Date and time of the data
psiprofdat	Profile data
numprofs	Number of profiles in the data file

Calling procedures
define_profobc_data

write_profobc_spec

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

File
Open_Boundary_Data_Prof.f90

Type
Subroutine

Purpose
Write the information needed for applying the open boundary conditions for a 3-D quantity (baroclinic current, C-node scalar) to a file in standard COHERENS format: type of condition, profile number of each open boundary point, number of profiles in each data file and index mapping array used to locate the profiles and variables within each data file.

Reference
Sections 12.2.4 and 16.2.1

Arguments

iddesc	Key id of the corresponding forcing data file

itypobu	Type of open boundary condition at the U-nodes (see Section 16.2.1.1)
itypobv	Type of open boundary condition at the V-nodes (see Section 16.2.1.1)
iprofobu	Profile number at the U-nodes for each data variable. If set to zero, the type of condition is determined by the value of itypobu.
iprofobv	Profile number at the V-nodes for each data variable. If set to zero, the type of condition is determined by the value of itypobv.
iprofrlx	Disables/enables relaxation in "relaxation" areas.
noprofsd	Number of profiles per data file
indexprof	Mapping array of the profile numbers in the data files to the profile numbers assigned to the open boundaries. The physical size of the first dimension equals the number of profiles in a data file.
indexvar	Defines the variable number of the profiles in a data file. The physical size of the first dimension equals the number of profiles in a data file.
novars	Total number of (scalar) variables. Value is 1 in the current implementation.
nofiles	Number of data files (minus 1)

Calling procedures

define_profobc_spec

30.19 *Open_Boundary_Data_2D.f90*

Open boundary data for the 2-D mode.

define_2dobc_data

```
SUBROUTINE define_2dobc_data(ifil,ciodatetime,data2d,nodat,novars)
CHARACTER (LEN=lentime), INTENT(INOUT) :: ciodatetime
INTEGER, INTENT(IN) :: ifil, nodat, novars
REAL, INTENT(INOUT), DIMENSION(nodat,novars) :: data2d
```

File

Open_Boundary_Data_2D.f90

Type
Subroutine

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

Reference
Section 16.1.2

Arguments

ifil	File index of the data file
ciodatetime	Returned date and time of the data
data2d	Returned open boundary data
nodat	Number of open boundary locations in the data file
novars	Number of data variables in the data file

Called external procedures
read_2dobc_data, usrdef_2dobc_data, write_2dobc_data

Calling procedures
update_2dobc_data

define_2dobc_spec

`SUBROUTINE define_2dobc_spec`

File
Open_Boundary_Data_2D.f90

Type
Subroutine

Purpose
Provide information for applying 2-D open boundary conditions: type of conditions, location of the 2-D data in the data files, amplitudes and phases in case an harmonic expansion is used.

Reference
Section 16.1.1

Called external procedures
read_2dobc_spec, usrdef_2dobc_spec, write_2dobc_spec

Calling procedures
update_2dobc_data

read_2dobc_data

```
SUBROUTINE read_2dobc_data(ifil,ciodatetime,data2d,nodat,novars)
CHARACTER (LEN=lentime), INTENT(OUT) :: ciodatetime
INTEGER, INTENT(IN) :: ifil, nodat, novars
REAL, INTENT(INOUT), DIMENSION(nodat,novars) :: data2d
```

File
Open_Boundary_Data_2D.f90

Type
Subroutine

Purpose
Read the 2-D open boundary data from a file in COHERENS standard format.

Reference
Section 16.1.2

Arguments

ifil	File index of the data file
ciodatetime	Returned date and time of the data
data2d	Returned input data
nodat	Number of open boundary locations in the data file
novars	Number of data variables in the data file (1 or 2)

Calling procedures
define_2dobc_data

read_2dobc_spec

```
SUBROUTINE read_2dobc_spec
```

File
Open_Boundary_Data_2D.f90

Type
Subroutine

Purpose
: Read the arrays which specify the type of condition and data, the location of the 2-D data within the data files, the amplitudes and phases in case of an harmonic expansion is used. The file is read in standard COHERENS format.

Reference
: Section 16.1.1

Calling procedures
: define_2dobc_spec

update_2dobc_data

```
SUBROUTINE update_2dobc_data
```

File
: *Open_Boundary_Data_2D.f90*

Type
: Subroutine

Purpose
: Update the 2-D open boundary data by reading new data (if needed). Interpolate at the current simulation time. Update the harmonic expansions (where needed) for U, V, ζ.

Reference
: Section 12.2.4

Called external procedures
: astro_params, define_2dobc_data, define_2dobc_spec

Calling procedures
: current_2d, initialise_model

write_2dobc_data

```
SUBROUTINE write_2dobc_data(ifil,ciodatetime,data2d,nodat,novars)
CHARACTER (LEN=lentime), INTENT(IN) :: ciodatetime
INTEGER, INTENT(IN) :: ifil, nodat, novars
REAL, INTENT(IN), DIMENSION(nodat,novars) :: data2d
```

File
Open_Boundary_Data_2D.f90

Type
Subroutine

Purpose
Write the 2-D open boundary data to a file in COHERENS standard format.

Arguments

ifil	File index of the data file
ciodatetime	Date and time of the data
data2d	Output data
nodat	Number of open boundary locations in the data file
novars	Number of data variables in the data file (1 or 2)

Calling procedures
define_2dobc_data

write_2dobc_spec

```
SUBROUTINE write_2dobc_spec
```

File
Open_Boundary_Data_2D.f90

Type
Subroutine

Purpose
Write the arrays which specify the type of condition and data, the location of the 2-D data within the data files, the amplitudes and phases in case of an harmonic expansion is used. The file is written in standard COHERENS format.

Reference
Section 16.1.1

Calling procedures
define_2dobc_spec

30.20 *Parallel_Initialisation.f90*

Initialisation procedures for parallel applications.

define_halo_comms

`SUBROUTINE define_halo_comms`

File
Parallel_Initialisation.f90

Type
Subroutine

Purpose
Define the setup for exchange communications

Reference
Section 11.4.3.2

Calling procedures
domain_decomposition

domain_coords

`SUBROUTINE domain_coords`

File
Parallel_Initialisation.f90

Type
Subroutine

Purpose
Define the domain indices of each sub-domain in the decomposition.

Reference
Section 11.2.1

Calling procedures
domain_decomposition

domain_decomposition

`SUBROUTINE domain_decomposition`

File
Parallel_Initialisation.f90

Type
Subroutine

Purpose
Create a domain decomposition for parallel applications.

Reference
Section 11.2.1

Called external procedures
define_halo_comms, domain_coords, read_partition, regular_partition, usrdef_partition, write_partition

Calling procedures
initialise_model

read_partition

`SUBROUTINE read_partition`

File
Parallel_Initialisation.f90

Type
Subroutine

Purpose
Read the arrays defining the domain decomposition from a file in standard COHERENS format.

Reference
Section 11.2.1

Calling procedures
domain_decomposition

regular_partition

```
SUBROUTINE regular_partition
```

File
Parallel_Initialisation.f90

Type
Subroutine

Purpose
Define a "simple" domain decomposition using a partitioning into nprocsx*nprocsy sub-domains.

Calling procedures
domain_decomposition

set_domain_dims

```
SUBROUTINE set_domain_dims(np,npx,npy)
INTEGER, INTENT(IN) :: np
INTEGER, INTENT(INOUT) :: npx, npy
```

File
Parallel_Initialisation.f90

Type
Subroutine

Purpose
Define the dimensions nprocxs, nprocsy of the domain decomposition in case they are not supplied by the user.

Reference
Section 11.2.1

Arguments

np	Number of processes
npx	Returned X-dimension of the decomposition
npy	Returned Y-dimension of the decomposition

Calling procedures
set_procnums

set_procnums

`SUBROUTINE set_procnums`

File
Parallel_Initialisation.f90

Type
Subroutine

Purpose
Define the values of nprocs, nprocsx, nprocsy.

Reference
Section 11.2.1

Called external procedures
set_domain_dims

Calling procedures
initialise_model

set_workers_comm

`SUBROUTINE set_workers_comm`

File
Parallel_Initialisation.f90

Type
Subroutine

Purpose
Separate active from idle processes. Define the comm_work communicator for all active processes.

Calling procedures
initialise_model

write_partition

`SUBROUTINE write_partition`

File
Parallel_Initialisation.f90

Type
Subroutine

Purpose
Write the arrays defining the domain decomposition to a file in standard COHERENS format.

Reference
Section 11.2.1

Calling procedures
domain_decomposition

30.21 *Relaxation_Zones.f90*

Routines for application of the relaxation scheme at open boundaries

define_rlxobc_spec

`SUBROUTINE define_rlxobc_spec`

File
Relaxation_Zones.f90

Type
Subroutine

Purpose
Define the zones for application of the open boundary relaxation scheme.

Reference
Section 16.3

Called external procedures
read_rlxobc_spec, usrdef_rlxobc_spec, write_rlxobc_spec

Calling procedures
initialise_model

read_rlxobc_spec

```
SUBROUTINE read_rlxobc_spec
```

File
Relaxation_Zones.f90

Type
Subroutine

Purpose
Read the arrays which define the areas for application of the open boundary relaxation scheme from a file in standard COHERENS format.

Reference
Section 16.3

Calling procedures
define_rlxobc_spec

relaxation_at_C

```
SUBROUTINE relaxation_at_C(psi,psiobu,psiobv,novars,iprofrlx)
INTEGER, INTENT(IN) :: novars
INTEGER, INTENT(IN), DIMENSION(norlxzones) :: iprofrlx
REAL, INTENT(INOUT), DIMENSION(1-nhalo:ncloc+nhalo,&
                              & 1-nhalo:nrloc+nhalo,nz,novars) :: psi
REAL, INTENT(INOUT), DIMENSION(nobu,nz,novars) :: psiobu
REAL, INTENT(INOUT), DIMENSION(nobv,nz,novars) :: psiobv
```

File
Relaxation_Zones.f90

Type
Subroutine

Purpose
Apply the relaxation scheme for a quantity or quantities at C-nodes

Reference
Section 4.10.3

Arguments

psi	Array of C-node quantity or quantities as defined on the model grid [psi]
psiobu	Vertical profiles of the C-node variable(s) at the U-open boundaries [psi]
psiobv	Vertical profiles of the C-node variable(s) at the V-open boundaries [psi]
novars	Number of C-node variables (currently 1)
iprofrlx	Array which selects the zones for application of the scheme

Calling procedures
transport_at_C_3d, transport_at_C_4d1, transport_at_C_4d2

relaxation_at_U

```
SUBROUTINE relaxation_at_U(psi,psiobu,nzdim,iprofrlx)
INTEGER, INTENT(IN) :: nzdim
INTEGER, INTENT(IN), DIMENSION(norlxzones) :: iprofrlx
REAL, INTENT(INOUT), DIMENSION(1-nhalo:ncloc+nhalo,&
                              & 1-nhalo:nrloc+nhalo,nzdim) :: psi
REAL, INTENT(INOUT), DIMENSION(nobu,nzdim) :: psiobu
```

File
Relaxation_Zones.f90

Type
Subroutine

Purpose
Apply the relaxation scheme for a quantity at the U-nodes

Reference
Section 4.10.3

Arguments

psi	U-node variable as defined on the model grid [psi]
psiobu	Vertical profiles of the U-node variables at the U-open boundaries [psi]
nzdim	Vertical dimension of the U-node variable
iprofrlx	Array which selects the zones for application of the scheme

Calling procedures
current_corr

relaxation_at_V

```
SUBROUTINE relaxation_at_V(psi,psiobv,nzdim,iprofrlx)
INTEGER, INTENT(IN) :: nzdim
INTEGER, INTENT(IN), DIMENSION(norlxzones) :: iprofrlx
REAL, INTENT(INOUT), DIMENSION(1-nhalo:ncloc+nhalo,&
                              & 1-nhalo:nrloc+nhalo,nzdim) :: psi
REAL, INTENT(INOUT), DIMENSION(nobv,nzdim) :: psiobv
```

File
Relaxation_Zones.f90

Type
Subroutine

Purpose
Apply the relaxation scheme for a quantity at V-nodes

Reference
Section 4.10.3

Arguments

psi	V-node variable as defined on the model grid	[psi]
psiobv	Vertical profiles of the V-node variables at the V-open boundaries	[psi]
nzdim	Vertical dimension of the V-node variable	
iprofrlx	Array which selects the zones for application of the scheme	

Calling procedures
current_corr

relaxation_weights

```
SUBROUTINE relaxation_weights
```

File
Relaxation_Zones.f90

Type
Subroutine

Purpose
Define the weight factors for the interpolation scheme.

Reference
Section 4.10.3

Calling procedures
initialise_model

write_rlxobc_spec

`SUBROUTINE write_rlxobc_spec`

File
Relaxation_Zones.f90

Type
Subroutine

Purpose
Write the arrays which define the areas for application of the open boundary relaxation scheme to a file in standard COHERENS format.

Reference
Section 16.3

Calling procedures
define_rlxobc_spec

30.22 *Structures_Model.f90*

Routines of the structures (dry cells, thin dams, weirs, barriers) and discharges model unit.

allocate_struc_arrays

`SUBROUTINE allocate_struc_arrays`

File
Structures_Model.f90

Type
Subroutine

Purpose
Allocate arrays for the structures and discharge model units. All arrays are global.

Calling procedures
initialise_model

define_dischr_data

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

File
Structures_Model.f90

Type
Subroutine

Purpose
Define the discharge data.

Arguments

iddesc	Key id of the corresponding data file
	io_disspc : specifier arrays
	io_disloc : discharge locations
	io_disvol : volume discharges
	io_discur : data for momentum discharge (area and direction)
	io_dissal : salinity discharge data
	io_distmp : temperature discharge data
ifil	File index of the data file
ciodatetime	Returned date and time of the data
disdata	Returned data
nodat	Number of discharge locations
novars	Number of data variables

Called external procedures
read_dischr_data, usrdef_dischr_data, write_dischr_data

Calling procedures
update_dischr_data

define_dischr_spec

`SUBROUTINE define_dischr_spec`

File
Structures_Model.f90

Type
Subroutine

Purpose
Define the following specifier arrays for the discharge module.

kdistype	type of vertical location
mdistype	flagging type

Called external procedures
read_dischr_spec, usrdef_dischr_spec, write_dischr_spec

Calling procedures
initialise_model

discharges

`SUBROUTINE discharges`

File
Structures_Model.f90

Type
Subroutine

Purpose
Define discharge locations, rates and directions.

Called external procedures
update_dischr_data

Calling procedures
coherens_main, initialise_model

dry_cells

```
SUBROUTINE dry_cells
```

File
Structures_Model.f90

Type
Subroutine

Purpose
Define dry cells.

Calling procedures
define_global_grid

local_struc_arrays

```
SUBROUTINE local_struc_arrays
```

File
Structures_Model.f90

Type
Subroutine

Purpose
Define the locations of thin dams, weirs and barriers on each local sub-domain.

Calling procedures
initialise_model

momentum_discharge_2d

```
SUBROUTINE momentum_discharge_2d(sourceatu,sourceatv)
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc) :: sourceatu, sourceatv
```

File
Structures_Model.f90

Type
Subroutine

Purpose
Define discharge source terms in the 2-D momentum equations.

Reference
Section 6.4.2

Arguments

sourceatu	Source term in the U-equation
sourceatv	Source term in the V-equation

Calling procedures
current_2d

momentum_discharge_3d

```
SUBROUTINE momentum_discharge_3d(sourceatu,sourceatv)
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc,nz) :: sourceatu, sourceatv
```

File
Structures_Model.f90

Type
Subroutine

Purpose
Define discharge source terms in the 3-D momentum equations.

Reference
Section 6.4.2

Arguments

sourceatu	Source term in the u-equation
sourceatv	Source term in the v-equation

Calling procedures
current_pred

read_dischr_data

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

File
Structures_Model.f90

Type
Subroutine

Purpose
Read discharge data from a file in standard COHERENS format.

Arguments

iddesc	Key id of the corresponding data file
	io_disspc : specifier arrays
	io_disloc : discharge locations
	io_disvol : volume discharges
	io_discur : data for momentum discharge (area and direction)
	io_dissal : salinity discharge data
	io_distmp : temperature discharge data
ifil	File index of the data file
ciodatetime	Returned date and time of the data
disdata	Returned data
nodat	Number of discharge locations
novars	Number of data variables

Calling procedures
define_dischr_data

read_dischr_spec

`SUBROUTINE read_dischr_spec`

File
Structures_Model.f90

Type
Subroutine

Purpose
Read specifier arrays for the discharge module from a file in standard COHERENS format.

Calling procedures
define_dischr_spec

read_dry_cells

`SUBROUTINE read_dry_cells`

File
Structures_Model.f90

Type
Subroutine

Purpose
Read dry cell locations from a file in standard COHERENS format.

Calling procedures
define_global_grid

read_thin_dams

`SUBROUTINE read_thin_dams`

File
Structures_Model.f90

Type
Subroutine

Purpose
Read thin dam locations from a file in standard COHERENS format.

Calling procedures
initialise_model

read_weirs

`SUBROUTINE read_weirs`

File
Structures_Model.f90

Type
Subroutine

Purpose
Read weirs/barriers locations and setup arrays from a file in standard COHERENS format.

Calling procedures
initialise_model

scalar_discharge

```
SUBROUTINE scalar_discharge(disscal,source,itype)
INTEGER, INTENT(IN) :: itype
REAL, INTENT(IN), DIMENSION(numdis) :: disscal
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc,nz) :: source
```

File
Structures_Model.f90

Type
Subroutine

Purpose
Define discharge source term in a scalar transport equation.

Reference
Section 6.4.2.1

Arguments

disscal	Source term in the scalar transport equation. Unit is [psi] for wet or [psi/s] for dry discharge	
source	Source term in the scalar transport equation	[psi/s]
itype	Type of discharge 1: wet discharge 2: dry discharge	

Calling procedures
salinity_equation, temperature_equation

surface_discharge

```
SUBROUTINE surface_discharge
```

File
Structures_Model.f90

Type
Subroutine

Purpose
Add volume discharge to the 2-D continuity equation.

Reference
Section 6.4.1

Calling procedures
surface_elevation

thin_dams

```
SUBROUTINE thin_dams(cnode)
CHARACTER (LEN=1), INTENT(IN) :: cnode
```

File
Structures_Model.f90

Type
Subroutine

Purpose
Update pointer arrays at thin dams.

Arguments

cnode	Velocity node of the thin dams ('U' or 'V')

Calling procedures
mask_function, pointer_arrays

update_dischr_data

```
SUBROUTINE update_dischr_data(iddesc,ifil,disdata,disvals,nodat,novars,&
                              & nosecsdat,update)
LOGICAL, INTENT(OUT) :: update
INTEGER, INTENT(IN) :: iddesc, ifil, nodat, novars
INTEGER (KIND=kndilong), INTENT(INOUT), DIMENSION(2) :: nosecsdat
REAL, INTENT(INOUT), DIMENSION(nodat,novars,2) :: disdata
REAL, INTENT(INOUT), DIMENSION(nodat,novars) :: disvals
```

File

Structures_Model.f90

Type

Subroutine

Purpose

Update discharge data (locations, volume, momentum and scalar discharge). Interpolate the data in space on the model grid and in time at the current simulation time (if requested).

Arguments

iddesc	Key id of the corresponding data file io_disspc : specifier arrays io_disloc : discharge locations io_disvol : volume discharges io_discur : data for momentum discharge (area and direction) io_dissal : salinity discharge data io_distmp : temperature discharge data
ifil	File index of the data file
disdata	Array of the discharge input data at the latest time earlier than the current date/time and at the earliest time later than the current date/time
disvals	Discharge input data interpolated spatially on the model grid and in time at the current time (if requested)
nodat	Number of discharge locations
novars	Number of data variables
nosecsdat	Number of seconds since the start of the simulation and the latest data time earlier than the current time and the earliest data time later than the current time
update	Returns .TRUE. if an update is made.

Called external procedures

define_dischr_data

Calling procedures

discharges, salinity_equation, temperature_equation

weirs_bstress

`SUBROUTINE weirs_bstress`

File
Structures_Model.f90

Type
Subroutine

Purpose
Evaluate bottom stress at weirs and barriers

Calling procedures
bottom_stress

weirs_depth

`SUBROUTINE weirs_depth`

File
Structures_Model.f90

Type
Subroutine

Purpose
Evaluate total water depth at weirs and barriers

Reference
Section 6.3.3

Calling procedures
water_depths

weirs_loss

`SUBROUTINE weirs_loss`

File
Structures_Model.f90

Type
Subroutine

Purpose
Evaluate energy losses at weirs and barriers

Reference
Section 6.3.1

Calling procedures
current_pred, current_2d

weirs_mask

```
SUBROUTINE weirs_mask
```

File
Structures_Model.f90

Type
Subroutine

Purpose
Update velocity pointer arrays at weirs and barriers

Called external procedures
update_pointer_arrays

Calling procedures
coherens_main

weirs_sink

```
SUBROUTINE weirs_sink(wsinkatu,wsinkatv,nzdim)
INTEGER, INTENT(IN) :: nzdim
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc,nzdim) :: wsinkatu, wsinkatv
```

File
Structures_Model.f90

Type
Subroutine

Purpose
Evaluate sink terms in the 2-D and 3-D momentum equations due to energy loss at weirs and barriers.

Reference
Section 6.3.3

Arguments

wsinkatu	sink term in the U- or u-equation
wsinkatv	sink term in the V- or v-equation
nzdim	vertical dimension of the sink terms (1 for 2-D, nz for 3-D mode)

Calling procedures
current_pred, current_2d

write_dischr_data

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

File
Structures_Model.f90

Type
Subroutine

Purpose
Write discharge data to a file in standard COHERENS format.

Arguments

iddesc	Key id of the corresponding data file	
	io_disspc	: specifier arrays
	io_disloc	: discharge locations
	io_disvol	: volume discharges
	io_discur	: data for momentum discharge (area and direction)
	io_dissal	: salinity discharge data
	io_distmp	: temperature discharge data
ifil	File index of the data file	
ciodatetime	Returned date and time of the data	

disdata	Returned data
nodat	Number of discharge locations
novars	Number of data variables

Calling procedures
define_dischr_data

write_dischr_spec

`SUBROUTINE write_dischr_spec`

File
Structures_Model.f90

Type
Subroutine

Purpose
Write specifier arrays for the discharge module to a file in standard COHERENS format.

Calling procedures
define_dischr_spec

write_dry_cells

`SUBROUTINE write_dry_cells`

File
Structures_Model.f90

Type
Subroutine

Purpose
Write dry cell locations to a file in standard COHERENS format.

Calling procedures
define_global_grid

write_thin_dams

`SUBROUTINE write_thin_dams`

File
Structures_Model.f90

Type
Subroutine

Purpose
Write thin dam locations to a file in standard COHERENS format.

Calling procedures
initialise_model

write_weirs

`SUBROUTINE write_weirs`

File
Structures_Model.f90

Type
Subroutine

Purpose
Write weirs/barriers locations and setup arrays to a file in standard COHERENS format.

Calling procedures
initialise_model

30.23 *Surface_Boundary_Data_1D.f90*

Ensemble of routines for the definition and application of surface boundary conditions (surface slopes, elevations) in case of a 1-D water column application

define_1dsur_data

```
SUBROUTINE define_1dsur_data(ciodatetime,data1d,novars)
CHARACTER (LEN=lentime), INTENT(INOUT) :: ciodatetime
INTEGER, INTENT(IN) :: novars
REAL, INTENT(INOUT), DIMENSION(novars) :: data1d
```

File
: *Surface_Boundary_Data_1D.f90*

Type
: Subroutine

Purpose
: Define the surface forcing data for 1-D applications.

Reference
: Section 17.1.2

Arguments

ciodatetime	Returned date and time of the data
data1d	Returned surface forcing data
novars	Number of data variables in the data file

Called external procedures
: read_1dsur_data, usrdef_1dsur_data, write_1dsur_data

Calling procedures
: update_1dsur_data

define_1dsur_spec

```
SUBROUTINE define_1dsur_spec
```

File
: *Surface_Boundary_Data_1D.f90*

Type
: Subroutine

Purpose
: Define the amplitudes and phases of the forcing data (if any) and the type of data in case there is an external data file.

Reference
Section 17.1.1

Called external procedures
read_1dsur_spec, usrdef_1dsur_spec, write_1dsur_spec

Calling procedures
update_1dsur_data

read_1dsur_data

```
SUBROUTINE read_1dsur_data(ciodatetime,data1d,novars)
CHARACTER (LEN=lentime), INTENT(INOUT) :: ciodatetime
INTEGER, INTENT(IN) :: novars
REAL, INTENT(INOUT), DIMENSION(novars) :: data1d
```

File
Surface_Boundary_Data_1D.f90

Type
Subroutine

Purpose
Read the surface forcing data for 1-D applications from a file in standard COHERENS format.

Reference
Section 17.1.2

Arguments

ciodatetime	Returned date and time of the data
data1d	Returned surface forcing data
novars	Number of data variables in the data file

Calling procedures
define_1dsur_data

read_1dsur_spec

```
SUBROUTINE read_1dsur_spec
```

File
Surface_Boundary_Data_1D.f90

Type
Subroutine

Purpose
Read the amplitudes and phases of the forcing data (if any) and the type of data in case there is an external data file from a file in standard COHERENS format.

Reference
Section 17.1.1

Arguments
None

Calling procedures
define_1dsur_spec

update_1dsur_data

```
SUBROUTINE update_1dsur_data
```

File
Surface_Boundary_Data_1D.f90

Type
Subroutine

Purpose
Update the surface forcing data (surface slopes, elevations) by reading new data (if needed). Interpolate at the current simulation time. Update the harmonic expansions for the surface slopes and the water elevation (if requested).

Called external procedures
astro_params, define_1dsur_data, define_1dsur_spec, water_depths

Calling procedures
current_pred, initialise_model

write_1dsur_data

```
SUBROUTINE write_1dsur_data(ciodatetime,data1d,novars)
CHARACTER (LEN=lentime), INTENT(IN) :: ciodatetime
INTEGER, INTENT(IN) :: novars
REAL, INTENT(IN), DIMENSION(3) :: data1d
```

File
Surface_Boundary_Data_1D.f90

Type
Subroutine

Purpose
Write the surface forcing data to a file in standard COHERENS format.

Reference
Section 17.1.2

Arguments

ciodatetime	Date and time of the data
data1d	Surface forcing data
novars	Number of data variables in the data file

Calling procedures
define_1dsur_data

write_1dsur_spec

`SUBROUTINE write_1dsur_spec`

File
Surface_Boundary_Data_1D.f90

Type
Subroutine

Purpose
Write the amplitudes and phases of the forcing data (if any) and the type of data in case there is an external data file to a file in standard COHERENS format.

Reference
Section 17.1.1

Calling procedures
define_1dsur_spec

30.24 *Surface_Data.f90*

Surface forcing data from a 2-D external grid.

define_surface_data

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

File
: *Surface_Data.f90*

Type
: Subroutine

Purpose
: Define (2-D) surface forcing data.

Reference
: Section 17.2.3

Arguments

iddesc	Key id of the corresponding forcing data file
ifil	File index of the data file
ciodatetime	Returned date and time of the data
surdata	Returned array of surface forcing data
n1dat	Dimension of the data grid in the X-direction
n2dat	Dimension of the data grid in the Y-direction
novars	Number of forcing data variables

Called external procedures
: read_surface_data, usrdef_surface_data, write_surface_data

Calling procedures
: update_surface_data

read_surface_data

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

File
Surface_Data.f90

Type
Subroutine

Purpose
Read the (2-D) surface forcing data from a file in standard COHERENS format.

Reference
Section 17.2.3

Arguments

iddesc	Key id of the corresponding forcing data file
ifil	File index of the data file
ciodatetime	Returned date and time of the data
surdata	Returned array of surface forcing data
n1dat	Dimension of the data grid in the X-direction
n2dat	Dimension of the data grid in the Y-direction
novars	Number of forcing data variables

Calling procedures
define_surface_data

update_surface_data

```
SUBROUTINE update_surface_data(iddesc,ifil,surindat,survals,surfgrid,&
                             & n1dat,n2dat,novars,n1grd,n2grd,&
                             & nosecsdat,datflag)
INTEGER, INTENT(IN) :: datflag, iddesc, ifil, novars, n1dat, n1grd, &
                     & n2dat, n2grd
INTEGER (KIND=kndilong), INTENT(INOUT), DIMENSION(2) :: nosecsdat
```

```
REAL, INTENT(INOUT), DIMENSION(n1dat,n2dat,novars,2) :: surindat
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc,novars) :: survals
TYPE (HRelativeCoords), INTENT(IN), DIMENSION(n1grd,n2grd) :: surfgrid
```

File

Surface_Data.f90

Type

Subroutine

Purpose

Update surface input defined on an external 2-D data grid by reading new data (if needed). Interpolate the data in space on the model grid and in time at the current simulation time (if requested).

Reference

Section 12.2.4

Arguments

iddesc	Key id of the corresponding forcing data file
ifil	File index of the data file
surindat	Array of surface data at the latest time earlier than the current date/time and at the earliest time later than the current date/time
survals	Surface data interpolated spatially on the model grid and in time at the current time (if requested)
surfgrid	Relative coordinates of the model grid with respect to the surface data grid
n1dat	Dimension of the data in the X-direction
n2dat	Dimension of the data in the Y-direction
novars	Number of forcing data variables
n1grd	Dimension of the surface grid in the X-direction
n2grd	Dimension of the surface grid in the Y-direction
nosecsdat	Number of seconds since the start of the simulation and the latest data time earlier than the current time and the earliest data time later than the current time
datflag	Type of data flagging for spatial interpolation 0: No data flags are applied

1: A data flag is applied if any of the surrounding data values is flagged

2: A data flag is applied only when all surrounding data values are flagged. Extrapolation is used if needed.

Called external procedures
define_surface_data

Calling procedures
meteo_input, temperature_equation

write_surface_data

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

File
Surface_Data.f90

Type
Subroutine

Purpose
Write the (2-D) surface forcing data to a file in standard COHERENS format.

Reference
Section 17.2.3

Arguments

iddesc	Key id of the corresponding forcing data file
ifil	File index of the data file
ciodatetime	Date and time of the data
surdata	Array of surface forcing data
n1dat	Dimension of the data grid in the X-direction
n2dat	Dimension of the data grid in the Y-direction
novars	Number of forcing data variables

Calling procedures
define_surface_data

30.25 *Surface_Fluxes.F90*

Meteorological forcing and surface fluxes.

heat_flux

`SUBROUTINE heat_flux`

File
Surface_Fluxes.F90

Type
Subroutine

Purpose
Evaluate all non-solar surface heat fluxes.

Reference
Section 4.7.3

Called external procedures
surface_exchange_coefs

Calling procedures
initialise_model, temperature_equation

meteo_input

`SUBROUTINE meteo_input`

File
Surface_Fluxes.F90

Type
Subroutine

Purpose
Update meteorological forcing data.

Called external procedures
define_surface_input_grid, surface_exchange_MO, update_surface_data

Calling procedures
coherens_main, initialise_model

salinity_flux

```
SUBROUTINE salinity_flux
```

File
Surface_Fluxes.F90

Type
Subroutine

Purpose
Surface salinity flux

Reference
Section 4.7.4

Calling procedures
salinity_equation

solar_irradiance

```
SUBROUTINE solar_irradiance
```

File
Surface_Fluxes.F90

Type
Subroutine

Purpose
Solar radiation at the surface

Reference
Section 4.6

Calling procedures
temperature_equation

surface_drag_coef

```
SUBROUTINE surface_drag_coef(wind)
REAL, INTENT(IN), DIMENSION(0:ncloc,0:nrloc) :: wind
```

File
Surface_Fluxes.F90

Type
Subroutine

Purpose
Surface drag coefficient C_{ds}

Reference
Section 4.8

Arguments

wind	Magnitude of the wind speed	[m/s]

Calling procedures
surface_stress

surface_drag_coef_0d

```
SUBROUTINE surface_drag_coef_0d(cdn,wind)
REAL, INTENT(OUT) :: cdn
REAL, INTENT(IN) :: wind
```

File
Surface_Fluxes.F90

Type
Subroutine

Purpose
Neutral surface drag coefficient C_{dn} as function of the wind speed

Reference
Section 4.8.1

Arguments

cdn	Returned neutral surface drag coefficient	
wind	Magnitude of the wind speed	[m/s]

Calling procedures
surface_exchange_MO

surface_exchange_coefs

```
SUBROUTINE surface_exchange_coefs(wind,tempdif)
REAL, INTENT(IN), DIMENSION(ncloc,nrloc) :: tempdif, wind
```

File
Surface_Fluxes.F90

Type
Subroutine

Purpose
Surface exchange coefficients C_e and C_h used in the calculation of the latent and sensible heat fluxes

Reference
Section 4.8

Arguments

wind	Magnitude of the wind speed	[m/s]
tempdif	Sea surface minus air temperature	[^{0}C]

Calling procedures
heat_flux

surface_exchange_coefs_0d

```
SUBROUTINE surface_exchange_coefs_0d(cen,chn,wind,tempdif)
REAL, INTENT(OUT) :: cen, chn
REAL, INTENT(IN) :: tempdif, wind
```

File
Surface_Fluxes.F90

Type
Subroutine

Purpose
Neutral surface exchange coefficients C_{en} and C_{hn} as function of the wind speed and the sea minus air temperature difference

Reference
Section 4.8.1

Arguments

cen	Returned neutral exchange coefficient Cen for the latent heat flux	
chn	Returned neutral exchange coefficient C_{hn} for the sensible heat flux	
wind	Magnitude of the wind speed	[m/s]
temdif	Sea surface minus air temperature difference	[^{0}C]

Calling procedures
surface_exchange_MO

surface_exchange_MO

`SUBROUTINE surface_exchange_MO`

File
Surface_Fluxes.F90

Type
Subroutine

Purpose
Solve the equations for the surface drag and exchange coefficients using the Monin-Obukhov similarity theory as function of wind speed, air minus sea temperature difference and relative humidity. The values are obtained for discretised values of the meteorological data. The results stored in tabular form.

Reference
Section 4.8.3

Called external procedures
surface_drag_coef_0d, surface_exchange_coefs_0d

Calling procedures
meteo_input

surface_stress

`SUBROUTINE surface_stress`

File
Surface_Fluxes.F90

Type
Subroutine

Purpose
Evaluate the surface stress

Reference
Section 4.7.2

Called external procedures
surface_drag_coef

Calling procedures
current_corr, current_corr_1d, current_2d, initialise_model

30.26 *Surface_Grids.f90*

Definition of external 2-D surface grids.

define_surface_input_grid

```
SUBROUTINE define_surface_input_grid(iddesc,ifil,surfgrid)
INTEGER, INTENT(IN) :: iddesc, ifil
TYPE (HRelativeCoords), INTENT(OUT), DIMENSION(ncloc,nrloc) :: surfgrid
```

File
Surface_Grids.f90

Type
Subroutine

Purpose
Define the relative coordinates of the model grid with respect to an external surface data grid

Reference
Sections 10.4.2 and 17.2

Arguments

iddesc	Key id of the surface grid file
ifil	File index of the grid file

surfgrid — Relative coordinates (on the local sub-domain) of the model grid with respect to the data grid

Called external procedures
read_surface_absgrd, read_surface_relgrd, usrdef_surface_absgrd, usrdef_surface_relgrd, write_surface_absgrd, write_surface_relgrd

Calling procedures
meteo_input, temperature_equation

define_surface_output_grid

```
SUBROUTINE define_surface_output_grid(iddesc,ifil,surfgridglb,nhdat)
INTEGER, INTENT(IN) :: iddesc, ifil, nhdat
TYPE (HRelativeCoords), INTENT(OUT), DIMENSION(nhdat,3) :: surfgridglb
```

File
Surface_Grids.f90

Type
Subroutine

Purpose
Define the relative coordinates of an external surface data grid with respect to the model grid. Intended for (future implementation of) coupling with surface wave and meteorological models.

Reference
Sections 10.5

Arguments

iddesc — Key id of the surface grid file

ifil — File index of the grid file

surfgridglb — Relative coordinates of the model grid with respect to the external grid. The second dimension denotes the nodal type of the model grid coordinates.

1: C-nodes

2: U-nodes

3: V-nodes

nhdat — Number of horizontal locations on the data grid

Called external procedures
read_surface_absgrd, read_surface_relgrd, usrdef_surface_absgrd, usrdef_surface_relgrd, write_surface_absgrd, write_surface_relgrd

Calling procedures
Currently not implemented

read_surface_absgrd

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

File
Surface_Grids.f90

Type
Subroutine

Purpose
Read the absolute coordinates of an external surface data grid from a file in standard COHERENS format.

Reference
Section 17.2.1

Arguments

iddesc	Key id of the surface grid file	
ifil	File index of the grid file	
n1dat	X-dimension of the surface data grid	
n2dat	Y-dimension of the surface data grid	
xcoord	X-coordinates of the data grid	[m or degrees longitude]
ycoord	Y-coordinates of the data grid	[m or degrees latitude]

Calling procedures
define_surface_input_grid, define_surface_output_grid

read_surface_relgrd

```
SUBROUTINE read_surface_relgrd(iddesc,ifil,surfgridglb,nx,ny,nonodes)
INTEGER, INTENT(IN) :: iddesc, ifil, nonodes, nx, ny
TYPE (HRelativeCoords), INTENT(OUT), &
                      & DIMENSION(nx,ny,nonodes) :: surfgridglb
```

File
Surface_Grids.f90

Type
Subroutine

Purpose
0 Read the relative coordinates of the model grid with respect from an external surface data grid to a file in standard COHERENS format.

Reference
Section 17.2.2

Arguments

iddesc	Key id of the surface grid file
ifil	File index of the grid file
surfgridglb	Relative coordinates of the (global) model grid with respect to the external surface data grid.
nx	X-dimension of the external surface grid
ny	Y-dimension of the external surface grid
nonodes	Number of nodes (currently set to 1)

Calling procedures
define_surface_input_grid, define_surface_output_grid

write_surface_absgrd

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

File
Surface_Grids.f90

Type
Subroutine

Purpose
Write the absolute coordinates of an external surface data grid to a file in standard COHERENS format.

Reference
Section 17.2.1

Arguments

iddesc	Key id of the surface grid file
ifil	File index of the grid file
n1dat	X-dimension of the surface data grid
n2dat	Y-dimension of the surface data grid
xcoord	X-coordinates of the data grid [m or degrees longitude]
ycoord	Y-coordinates of the data grid [m or degrees latitude]

Calling procedures
define_surface_input_grid, define_surface_output_grid

write_surface_relgrd

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

File
Surface_Grids.f90

Type
Subroutine

Purpose
Write the relative coordinates of the model grid with respect to an external surface data grid to a file in standard COHERENS format.

Reference
Section 17.2.2

Arguments

iddesc	Key id of the surface grid file
ifil	File index of the grid file
surfgridglb	Relative coordinates of the (global) model grid with respect to the external surface data grid.
nx	X-dimension of the external surface grid
ny	Y-dimension of the external surface grid
nonodes	Number of nodes (currently set to 1)

Calling procedures
define_surface_input_grid, define_surface_output_grid

30.27 *Surface_Waves.f90*

Definition of surface wave data.

wave_input

```
SUBROUTINE wave_input
```

File
Surface_Waves.f90

Type
Subroutine

Purpose
Update surface wave data.

Reference
Section 7.2.2

Called external procedures
define_surface_input_grid, update_surface_data, wave_number

Calling procedures
coherens_main, initialise_model

wave_number

```
SUBROUTINE wave_number
```

File
Surface_Waves.f90

Type
Subroutine

Purpose
Solvers the wave dispersion relation (7.9) using the approximate formula of Hunt (1979).

Calling procedures
wave

30.28 *Tidal_Forcing.F90*

Astronomical phases, nodal factors and tidal astronomical force.

astro_params

```
SUBROUTINE astro_params(index_tides,fnode_tides,phase_tides,&
                      & notides,dlonref_phase,cdatetime_tides)
CHARACTER (LEN=lentime), INTENT(IN) :: cdatetime_tides
INTEGER, INTENT(IN) :: notides
REAL, INTENT(IN) :: dlonref_phase
INTEGER, INTENT(IN), DIMENSION(notides) :: index_tides
REAL, INTENT(OUT), DIMENSION(notides) :: fnode_tides, phase_tides
```

File
Tidal_Forcing.F90

Type
Subroutine

Purpose
Calculate nodal factors and astronomical arguments at the given date and time.

Reference
Section 4.5

Arguments

index_tides	Key ids of the tidal constituents	
fnode_tides	Returned nodal correction factors	
phase_tides	Returned astronomical arguments	[radians]
notides	Number of tidal constituents	
dlonref_phase	Reference longitude to be used for the phases	[degrees]
cdatetime_tides	Calendar date and time. Time is converted to GMT if time_zone is non-zero.	

Calling procedures
astronomical_tides, harmonic_analysis_data, update_1dsur_data, update_2dobc_data

astronomical_tides

`SUBROUTINE astronomical_tides`

File
Tidal_Forcing.F90

Type
Subroutine

Purpose
Components of the astronomical tidal force

Reference
Section 4.5

Called external procedures
astro_params

Calling procedures
coherens_main, current_2d, initialise_model

30.29 *Time_Averages.f90*

Time averaged output.

time_averages

`SUBROUTINE time_averages`

File
Time_Averages.f90

Type
Subroutine

Purpose
Base program unit for time averaged output

Called internal procedures
time_averages_grid, time_averages_init, time_averages_reset, time_averages_update, time_averages_write

Calling procedures
coherens_main

time_averages_grid

`SUBROUTINE time_averages_grid`

File
Time_Averages.f90

Type
Internal subroutine

Purpose
Write the coordinates of the output grid to the data file

Calling procedures
time_averages

time_averages_init

`SUBROUTINE time_averages_init`

File
Time_Averages.f90

Type
Internal subroutine

Purpose
: Define the arrays for the setup of time averaging. Write the metadata to the data file.

Reference
: Section 20.2.1

Called external procedures
: usrdef_avr_params

Calling procedures
: time_averages

time_averages_reset

`SUBROUTINE time_averages_reset`

File
: *Time_Averages.f90*

Type
: Internal subroutine

Purpose
: Initialise buffers for storing the averaged data.

Calling procedures
: time_averages

time_averages_update

`SUBROUTINE time_averages_update`

File
: *Time_Averages.f90*

Type
: Internal subroutine

Purpose
: Add the current data to the buffers containing the current averages.

Called external procedures
: usrdef_avr0d_vals, usrdef_avr2d_vals, usrdef_avr3d_vals

Calling procedures
time_averages

time_averages_write

`SUBROUTINE time_averages_write`

File
Time_Averages.f90

Type
Internal subroutine

Purpose
Write the time-averaged output to the appropriate data files.

Calling procedures
time_averages

30.30 *Time_Series.f90*

Time series output.

time_series

`SUBROUTINE time_series`

File
Time_Series.f90

Type
Subroutine

Purpose
Write time series output to the appropriate output files.

Called internal procedures
time_series_grid, time_series_init

Called external procedures
usrdef_tsr0d_vals, usrdef_tsr2d_vals, usrdef_tsr3d_vals

Calling procedures
coherens_main

time_series_grid

`SUBROUTINE time_series_grid`

File
Time_Series.f90

Type
Internal subroutine

Purpose
Write the coordinates of the output grid to the data file

Calling procedures
time_series

time_series_init

`SUBROUTINE time_series_init`

File
Time_Series.f90

Type
Internal subroutine

Purpose
Define the arrays for the setup of time series output. Write the meta-data to the data file.

Reference
Section 20.1.1

Called external procedures
usrdef_tsr_params

Calling procedures
time_series

30.31 *Transport_Equations.F90*

Solve the transport equations for scalar and vector quantities at the different nodes.

transport_at_C_3d

```
SUBROUTINE transport_at_C_3d(psic,source,wadvatw,vdifcoefatw,iopt_hadv,&
                           & iopt_vadv,iopt_hdif,psiobu,psiobv,iprofrlx,&
                           & ibcsur,ibcbot,nbcs,nbcb,bcsur,bcbot,ivarid)
INTEGER, INTENT(IN) :: ibcbot, ibcsur, iopt_hadv, iopt_hdif, &
                     & iopt_vadv, ivarid, nbcb, nbcs
INTEGER, INTENT(IN), DIMENSION(norlxzones) :: iprofrlx
REAL, INTENT(IN), DIMENSION(nobu,nz) :: psiobu
REAL, INTENT(IN), DIMENSION(nobv,nz) :: psiobv
REAL, INTENT(INOUT), DIMENSION(1-nhalo:ncloc+nhalo,&
                             & 1-nhalo:nrloc+nhalo,nz) :: psic
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc,nz) :: source
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nz+1) :: vdifcoefatw, wadvatw
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nbcs) :: bcsur
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nbcb) :: bcbot
```

File

Transport_Equations.F90

Type

Subroutine

Purpose

Solve the advection-diffusion equation for a 3-D scalar quantity at the C-nodes.

Reference

Section 5.5

Arguments

psic	C-node quantity for which a transport equation needs to be solved	[psic]
source	Source terms in the transport equation	[psic/s]
wadvatw	Transformed vertical current ω plus (eventually) sinking/swimming vertical current at the W-nodes	[m/s]
vdifcoefatw	Vertical diffusion coefficient at the W-nodes	[m^2/s]
iopt_hadv	Switch to select horizontal advection scheme	
iopt_vadv	Switch to select vertical advection scheme	
iopt_hdif	Switch to select horizontal diffusion scheme	

psiobu — Vertical profiles at the U-open boundaries (a zero gradient condition is applied if flagged) [psic]

psiobv — Vertical profiles at the V-open boundaries (a zero gradient condition is applied if flagged) [psic]

iprofrlx — Selects whether a relaxation scheme is applied at each of the defined open boundary zones.

ibcsur — Type of surface boundary condition

- 0: zero flux (Neumann) condition
- 1: prescibed flux (Neumann) condition (5.439)
- 2: Neumann condition using a transfer velocity (5.440)
- 3: Dirichlet condition (5.441) at the first C-node below the surface
- 4: Dirichlet condition (5.442) at the surface

ibcbot — Type of bottom boundary condition

- 0: zero flux (Neumann) condition
- 1: prescibed flux (Neumann) condition (5.444)
- 2: Neumann condition using a transfer velocity (5.445)
- 3: Dirichlet condition (5.446) at the first C-node above the bottom
- 4: Dirichlet condition (5.447) at the bottom

nbcs — Last dimension of the array bcsur

nbcb — Last dimension of the array bcbot

bcsur — Data for the surface boundary condition. The third index defines the type of the data

- 1: prescribed surface flux or surface value [psic m/s] or [psic]
- 2: transfer velocity [m/s]

bcbot — Data for the bottom boundary condition. The third index defines the type of the data

- 1: prescribed bottom flux or bottom value [psic m/s] or [psic]
- 2: transfer velocity [m/s]

ivarid — Variable key id of the transported variable (used for log info only, zero for undefined)

Called external procedures
relaxation_at_C, Xadv_at_C, Xcorr_at_C, Xdif_at_C, Yadv_at_C, Ycorr_at_C, Ydif_at_C, Zadv_at_C, Zcorr_at_C, Zdif_at_C

Calling procedures
salinity_equation, temperature_equation

transport_at_C_4d1

```
SUBROUTINE transport_at_C_4d1(psic,source,wsink,vdifcoefatw,novars,&
                            & iopt_hadv,iopt_vadv,iopt_hdif,psiobu,&
                            & psiobv,iprofrlx,ibcsur,ibcbot,nbcs,nbcb,&
                            & bcsur,bcbot,settling,ivarids)
LOGICAL, INTENT(IN) :: settling
INTEGER, INTENT(IN) :: ibcbot, ibcsur, iopt_hadv, iopt_hdif,&
                     & iopt_vadv, nbcb, nbcs, novars
INTEGER, INTENT(IN), DIMENSION(novars) :: ivarids
INTEGER, INTENT(IN), DIMENSION(norlxzones) :: iprofrlx
REAL, INTENT(IN), DIMENSION(nobu,nz,novars) :: psiobu
REAL, INTENT(IN), DIMENSION(nobv,nz,novars) :: psiobv
REAL, INTENT(INOUT), DIMENSION(1-nhalo:ncloc+nhalo,&
                             & 1-nhalo:nrloc+nhalo,nz,novars) :: psic
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc,nz,novars) :: source
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nz+1) :: vdifcoefatw
REAL, INTENT(IN), DIMENSION(0:ncloc,0:nrloc,nz+1,novars) :: wsink
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nbcs,novars) :: bcsur
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nbcb,novars) :: bcbot
```

File
Transport_Equations.F90

Type
Subroutine

Purpose
Solve the advection-diffusion equation for one or more scalar quantities at the C-node. The equations are solved sequentially.

Reference
Section 5.5

Arguments

psic	C-node quantity or quantities for which transport equation(s) need(s) to be solved [psic]
source	Source terms in the transport equation(s) [psic/s]
wsink	Vertical (positive downward) sinking velocity at the W-nodes if argument settling equals .TRUE. not used otherwise [m/s]
vdifcoefatw	Vertical diffusion coefficient at the W-nodes [m^2/s]
novars	Number of variables to be transported
iopt_hadv	Switch to select horizontal advection scheme
iopt_vadv	Switch to select vertical advection scheme
iopt_hdif	Switch to select horizontal diffusion scheme
psiobu	Vertical profiles at the U-open boundaries (a zero gradient condition is applied if flagged) [psic]
psiobv	Vertical profiles at the V-open boundaries (a zero gradient condition is applied if flagged) [psic]
iprofrlx	Selects whether a relaxation scheme is applied at each of the defined open boundary zones.
ibcsur	Type of surface boundary condition 0: zero flux (Neumann) condition 1: prescibed flux (Neumann) condition (5.439) 2: Neumann condition using a transfer velocity (5.440) 3: Dirichlet condition (5.441) at the first C-node below the surface 4: Dirichlet condition (5.442) at the surface
ibcbot	Type of bottom boundary condition 0: zero flux (Neumann) condition 1: prescibed flux (Neumann) condition (5.444) 2: Neumann condition using a transfer velocity (5.445) 3: Dirichlet condition (5.446) at the first C-node above the bottom 4: Dirichlet condition (5.447) at the bottom
nbcs	Second last dimension of the array bcsur
nbcb	Second last dimension of the array bcbot

bcsur — Data for the surface boundary condition. The third index defines the type of the data

1: prescribed surface flux or surface value [psic m/s] or [psic]

2: transfer velocity [m/s]

bcbot — Data for the bottom boundary condition. The third index defines the type of the data

1: prescribed bottom flux or bottom value [psic m/s] or [psic]

2: transfer velocity [m/s]

settling — Interprets argument wsink as a (downward) settling velocity if .TRUE.

ivarids — Variable key id(s) of the transported variable(s) (used for log info only, zero for undefined)

Called external procedures

relaxation_at_C, Xadv_at_C, Xcorr_at_C, Xdif_at_C, Yadv_at_C, Ycorr_at_C, Ydif_at_C, Zadv_at_C, Zcorr_at_C, Zdif_at_C

Calling procedures

Currently not implemented

transport_at_C_4d2

```
SUBROUTINE transport_at_C_4d2(psic,source,wsink,vdifcoefatw,novars,&
                            & iopt_hadv,iopt_vadv,iopt_hdif,psiobu,&
                            & psiobv,iprofrlx,ibcsur,ibcbot,nbcs,nbcb,&
                            & bcsur,bcbot,settling,ivarid)
LOGICAL, INTENT(IN) :: settling
INTEGER, INTENT(IN) :: ibcbot, ibcsur, iopt_hadv, iopt_hdif,&
                     & iopt_vadv, ivarid, nbcb, nbcs, novars
INTEGER, INTENT(IN), DIMENSION(norlxzones) :: iprofrlx
REAL, INTENT(IN), DIMENSION(nobu,nz,novars) :: psiobu
REAL, INTENT(IN), DIMENSION(nobv,nz,novars) :: psiobv
REAL, INTENT(INOUT), DIMENSION(1-nhalo:ncloc+nhalo,&
                             & 1-nhalo:nrloc+nhalo,nz,novars) :: psic
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc,nz,novars) :: source
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nz+1) :: vdifcoefatw
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nz+1,novars) :: wsink
```

```
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nbcs,novars) :: bcsur
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nbcb,novars) :: bcbot
```

File

Transport_Equations.F90

Type

Subroutine

Purpose

Solve the advection-diffusion equation for a vector of scalar quantities at the C-node. The equations are solved simultaneously for all vector components.

Reference

Section 5.5

Arguments

psic	C-node vector of scalar quantities for which transport equations need to be solved	[psic]
source	Source terms in the transport equations	[psic/s]
wsink	Vertical (positive downward) sinking velocities at the W-nodes if argument settling equals .TRUE. not used otherwise	[m/s]
vdifcoefatw	Vertical diffusion coefficient at the W-nodes	[m^2/s]
novars	Number of variables to be transported	
iopt_hadv	Switch to select horizontal advection scheme	
iopt_vadv	Switch to select vertical advection scheme	
iopt_hdif	Switch to select horizontal diffusion scheme	
psiobu	Vertical profiles at the U-open boundaries (a zero gradient condition is applied if flagged)	[psic]
psiobv	Vertical profiles at the V-open boundaries (a zero gradient condition is applied if flagged)	[psic]
iprofrlx	Selects whether a relaxation scheme is applied at each of the defined open boundary zones.	
ibcsur	Type of surface boundary condition 0: zero flux (Neumann) condition 1: prescibed flux (Neumann) condition (5.439)	

2: Neumann condition using a transfer velocity (5.440)

3: Dirichlet condition (5.441) at the first C-node below the surface

4: Dirichlet condition (5.442) at the surface

ibcbot — Type of bottom boundary condition

0: zero flux (Neumann) condition

1: prescibed flux (Neumann) condition (5.444)

2: Neumann condition using a transfer velocity (5.445)

3: Dirichlet condition (5.446) at the first C-node above the bottom

4: Dirichlet condition (5.447) at the bottom

nbcs — Second last dimension of the array bcsur

nbcb — Second last dimension of the array bcbot

bcsur — Data for the surface boundary condition. The third index defines the type of the data

1: prescribed surface flux or surface value [psic m/s] or [psic]

2: transfer velocity [m/s]

bcbot — Data for the bottom boundary condition. The third index defines the type of the data

1: prescribed bottom flux or bottom value [psic m/s] or [psic]

2: transfer velocity [m/s]

settling — Interprets argument wsink as a (downward) settling velocity if .TRUE.

ivarid — Variable key id for the vector of C-node scalars (used for log info only, zero for undefined)

Called external procedures

relaxation_at_C, Xadv_at_C, Xcorr_at_C, Xdif_at_C, Yadv_at_C, Ycorr_at_C, Ydif_at_C, Zadv_at_C, Zcorr_at_C, Zdif_at_C

Calling procedures

Currently not implemented

transport_at_U_2d

```
SUBROUTINE transport_at_U_2d(source,sink)
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc) :: sink, source
```

File

Transport_Equations.F90

Type

Subroutine

Purpose

Solve the 2-D momentum equation for the X-component U of the depth-integrated current.

Reference

Section 5.3

Arguments

source	Explicit source terms in the U-equation	$[m^2/s^2]$
sink	Implicit sink terms in the U-equation arising from bottom stress	$[s^{-1}]$

Called external procedures

Xadv_at_U_2d, Xdif_at_U_2d, Yadv_at_U_2d, Ydif_at_U_2d

Calling procedures

current_2d

transport_at_U_3d

```
SUBROUTINE transport_at_U_3d(source,ibcsur,ibcbot,nbcs,nbcb,&
                           & bcsur,bcbot)
INTEGER, INTENT(IN) :: ibcbot, ibcsur, nbcb, nbcs
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc,nz) :: source
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nbcs) :: bcsur
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nbcb) :: bcbot
```

File

Transport_Equations.F90

Type

Subroutine

Purpose

Solve the 3-D momentum equation for the X-component u of the 3-D current.

Reference

Section 5.3

Arguments

source	Source terms in the u-equation
ibcsur	Type of surface boundary condition
	0: zero flux (Neumann) condition
	1: prescibed flux (Neumann) condition (5.245)
ibcbot	Type of bottom boundary condition
	0: zero flux (Neumann) condition
	1: prescibed flux (Neumann) condition
	2: Neumann condition using a transfer velocity (see equations (5.249), (5.251), (5.252))
nbcs	Last dimension of the array bcsur
nbcb	Last dimension of the array bcbot
bcsur	Data for the surface boundary condition. The third index defines the type of the data
	1: prescribed surface flux or surface value [m^2/s^2] or [m/s]
	2: transfer velocity [m/s]
bcbot	Data for the bottom boundary condition. The third index defines the type of the data
	1: prescribed bottom flux or bottom value [m^2/s^2] or [m/s]
	2: transfer velocity [m/s]

Called external procedures

Xadv_at_U_3d, Xdif_at_U_3d, Yadv_at_U_3d, Ydif_at_U_3d, Zadv_at_U, Zdif_at_U

Calling procedures

current_pred

transport_at_V_2d

```
SUBROUTINE transport_at_V_2d(source,sink)
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc) :: sink, source
```

File

Transport_Equations.F90

Type

Subroutine

Purpose

Solve the 2-D momentum equation for the Y-component V of the depth-integrated current.

Reference

Section 5.3

Arguments

source	Explicit source terms in the V-equation	$[m^2/s^2]$
sink	Implicit sink terms in the V-equation arising from bottom stress	$[s^{-1}]$

Called external procedures

Xadv_at_V_2d, Xdif_at_V_2d, Yadv_at_V_2d, Ydif_at_V_2d

Calling procedures

current_2d

transport_at_V_3d

```
SUBROUTINE transport_at_V_3d(source,ibcsur,ibcbot,nbcs,nbcb,&
                           & bcsur,bcbot)
INTEGER, INTENT(IN) :: ibcbot, ibcsur, nbcb, nbcs
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc,nz) :: source
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nbcs) :: bcsur
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nbcb) :: bcbot
```

File

Transport_Equations.F90

Type

Subroutine

Purpose

Solve the 3-D momentum equation for the Y-component v of the 3-D current.

Reference

Section 5.3

Arguments

source	Source terms in the v-equation
ibcsur	Type of surface boundary condition
	0: zero flux (Neumann) condition
	1: prescibed flux (Neumann) condition (5.245)
ibcbot	Type of bottom boundary condition
	0: zero flux (Neumann) condition
	1: prescibed flux (Neumann) condition
	2: Neumann condition using a transfer velocity (see equations (5.250), (5.251), (5.253))
nbcs	Last dimension of the array bcsur
nbcb	Last dimension of the array bcbot
bcsur	Data for the surface boundary condition. The third index defines the type of the data
	1: prescribed surface flux or surface value $[m^2/s^2]$ or $[m/s]$
	2: transfer velocity $[m/s]$
bcbot	Data for the bottom boundary condition. The third index defines the type of the data
	1: prescribed bottom flux or bottom value $[m^2/s^2]$ or $[m/s]$
	2: transfer velocity $[m/s]$

Called external procedures

Xadv_at_V_3d, Xdif_at_V_3d, Yadv_at_V_3d, Ydif_at_V_3d, Zadv_at_V, Zdif_at_V

Calling procedures

current_pred

transport_at_W

```
SUBROUTINE transport_at_W(psiw,source,sink,vdifcoefatw,&
                        & ibcsur,ibcbot,bcsur,bcbot,ivarid)
INTEGER, INTENT(IN) :: ibcbot, ibcsur, ivarid
REAL, INTENT(INOUT),DIMENSION(1-nhalo:ncloc+nhalo,&
                              & 1-nhalo:nrloc+nhalo,nz+1) :: psiw
REAL, INTENT(INOUT), DIMENSION(ncloc,nrloc,2:nz) :: sink, source
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nz+1) :: vdifcoefatw
REAL, INTENT(IN), DIMENSION(ncloc,nrloc) :: bcbot, bcsur
```

File

Transport_Equations.F90

Type

Subroutine

Purpose

Solve the advection-diffusion equation for a 3-D scalar (turbulent) quantity at the W-node.

Reference

Section 5.6

Arguments

psiw	W-node quantity for which a transport equation needs to be solved	[psiw]
source	Explicit source terms in the transport equation	[psic/s]
sink	Implicit sink terms in the transport equation	$[s^{-1}]$
vdifcoefatw	Vertical diffusion coefficient at the W-nodes	$[m^2/s]$
ibcsur	Type of surface boundary condition	
	0: zero flux (Neumann) condition	
	1: prescibed (Neumann) flux at the surface (see equations (5.520)–(5.522))	
	2: prescribed Neumann flux at the first C-node below the surface (5.523)	
	3: Dirichlet condition (5.524) at the surface	
	4: Dirichlet condition (5.525) at the first W-node below the surface	
ibcbot	Type of bottom boundary condition	

0: zero flux (Neumann) condition

1: prescibed (Neumann) flux at the bottom (see equations (5.526)–(5.528))

2: prescribed Neumann flux at the first C-node above the bottom (5.529)

3: Dirichlet condition (5.530) at the bottom

4: Dirichlet condition (5.531) at the first W-node above the bottom

bcsur — Imposed surface flux or surface value

bcbot — Imposed bottom flux or bottom value

ivarid — Variable key id of the transported variable (used for log info only, zero for undefined)

Called external procedures
Xadv_at_W, Xcorr_at_W, Xdif_at_W, Yadv_at_W, Ycorr_at_W, Ydif_at_W, Zadv_at_W, Zcorr_at_W, Zdif_at_W

Calling procedures
dissipation_equation, kl_equation, tke_equation

30.32 *Turbulence_Equations.F90*

dissipation_equation

`SUBROUTINE dissipation_equation`

File
Turbulence_Equations.F90

Type
Subroutine

Purpose
Solve the ε-equation (4.205).

Reference
Section 5.6

Called external procedures
transport_at_W

Calling procedures
vertical_diff_coefs

dissip_lim_conds

`SUBROUTINE dissip_lim_conds`

File
Turbulence_Equations.F90

Type
Subroutine

Purpose
Apply the limiting condition for the dissipation rate ε given by equation (4.225).

Reference
Section 4.4.3.6

Calling procedures
vertical_diff_coefs

dissip_to_zlmix

`SUBROUTINE dissip_to_zlmix`

File
Turbulence_Equations.F90

Type
Subroutine

Purpose
Calculate the mixing length l from the dissipation rate ε using equation (4.203).

Calling procedures
vertical_diff_coefs

eddy_coefs_alg

`SUBROUTINE eddy_coefs_alg`

File
Turbulence_Equations.F90

Type
Subroutine

Purpose
Calculate the eddy coefficients ν_T and λ_T from an algebraic relation

Reference
Section 4.4.2

Calling procedures
vertical_diff_coefs

eddy_coefs_tc

`SUBROUTINE eddy_coefs_tc`

File
Turbulence_Equations.F90

Type
Subroutine

Purpose
Calculate the eddy coefficients ν_T and λ_T from a RANS model.

Reference
Section 4.4.3

Calling procedures
vertical_diff_coefs

init_turbulence

`SUBROUTINE init_turbulence`

File
Turbulence_Equations.F90

Type
Subroutine

Purpose
Initialise parameters for different turbulence closures schemes

Calling procedures
vertical_diff_coefs, default_phsics

kl_equation

`SUBROUTINE kl_equation`

File
Turbulence_Equations.F90

Type
Subroutine

Purpose
Solve the kl (q^2l) equation (4.209).

Reference
Section 5.6

Called external procedures
transport_at_W

Calling procedures
vertical_diff_coefs

mixing_length

`SUBROUTINE mixing_length`

File
Turbulence_Equations.F90

Type
Subroutine

Purpose
Algebraic mixing length formulations

Reference
Section 4.4.3.5

Calling procedures
default_phsics, vertical_diff_coefs

tke_equation

`SUBROUTINE tke_equation`

File
Turbulence_Equations.F90

Type
Subroutine

Purpose
Solve the k-equation (4.204).

Reference
Section 5.6

Called external procedures
transport_at_W

Calling procedures
vertical_diff_coefs

tke_equilibrium

`SUBROUTINE tke_equilibrium`

File
Turbulence_Equations.F90

Type
Subroutine

Purpose
Calculate the turbulent energy k using the equilibrium (level "2") method.

Reference
Section 4.4.3.3

Calling procedures
vertical_diff_coefs

zlmix_lim_conds

`SUBROUTINE zlmix_lim_conds`

File
Turbulence_Equations.F90

Type
Module subroutine

Purpose
Apply the limiting condition for the mixing length l given by equation (4.225).

Reference

Calling procedures
vertical_diff_coefs

zlmix_to_dissip

`SUBROUTINE zlmlix_to_dissip`

File
Turbulence_Equations.F90

Type
Module subroutine

Purpose
Calculate the dissipation rate ε from the mixing length l using equation (4.203).

Calling procedures
default_phsics, vertical_diff_coefs