Chapter 31

Description of modules routines

31.1 Array interpolation

Perform array interpolation on model grid arrays. The names of the routines are of the form

*X*arr_at_*Y* interpolation of a section of a model grid array defined at node *X* to node *Y*

*X*var_at_*Y* interpolation of a model grid array defined at node *X* to a grid point at node *Y*

Carr_at_U

```
SUBROUTINE Carr_at_U(xin,xout,intsrce,intdest,lbounds,ubounds,&
                  & nosize,ivarid,info,outflag,hregular)
LOGICAL, INTENT(IN) :: info
LOGICAL, INTENT(IN), OPTIONAL :: hregular
INTEGER, INTENT(IN) :: intdest, intsrce, ivarid, nosize
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
REAL, INTENT(IN), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                           & lbounds(3):ubounds(3),nosize) :: xin
REAL, INTENT(OUT), DIMENSION(lbounds(1)+1:ubounds(1),lbounds(2):ubounds(2),&
                            & lbounds(3):ubounds(3),nosize) :: xout
```

File
array_interp.f90

Type
Module subroutine

Purpose

Interpolate a section of a model grid C-node array to the U-nodes.

Reference

Section 10.2

Arguments

xin	Source array at the C-nodes
xout	Destination array at the U-nodes
intsrce	Selects valid points at the source node 0: all points 1: wet points only
intdest	Selects valid points at the destination node 0: all points 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
lbounds	Lower bounds of the source array
ubounds	Upper bounds of the source array
nosize	Fourth dimension of the source array
ivarid	Variable key id (used for log info only, zero for undefined)
info	Disables/enables writing of a log info.
outflag	Output flag for invalid points, if present. Zero otherwise.
hregular	Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.

Carr_at_UV

```
SUBROUTINE Carr_at_UV(xin,xout,intsrce,intdest,lbounds,ubounds,&
                  & nosize,ivarid,info,outflag,hregular)
LOGICAL, INTENT(IN) :: info
LOGICAL, INTENT(IN), OPTIONAL :: hregular
INTEGER, INTENT(IN) :: intdest, intsrce, ivarid, nosize
```

```
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
REAL, INTENT(IN), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                           & lbounds(3):ubounds(3),nosize) :: xin
REAL, INTENT(OUT), DIMENSION(lbounds(1)+1:ubounds(1),lbounds(2)+1:ubounds(2),&
                            & lbounds(3):ubounds(3),nosize) :: xout
```

File

array_interp.f90

Type

Module subroutine

Purpose

Interpolate a section of a model grid C-node array to the UV-nodes.

Reference

Section 10.2

Arguments

xin	Source array at the C-nodes
xout	Destination array at the UV-nodes
intsrce	Selects valid points at the source node
	0: all points
	1: wet points only
intdest	Selects valid points at the destination node
	0: all points
	1: wet points only
lbounds	Lower bounds of the source array
ubounds	Upper bounds of the source array
nosize	Fourth dimension of the source array
ivarid	Variable key id (used for log info only, zero for undefined)
info	Disables/enables writing of a log info.
outflag	Output flag for invalid points, if present. Zero otherwise.
hregular	Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.

Carr_at_UW

```
SUBROUTINE Carr_at_UW(xin,xout,intdest,lbounds,ubounds,nosize,&
                  & ivarid,info,outflag,hregular,vregular)
LOGICAL, INTENT(IN) :: info
LOGICAL, INTENT(IN), OPTIONAL :: hregular, vregular
INTEGER, INTENT(IN) :: intdest, ivarid, nosize
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
REAL, INTENT(IN), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                           & lbounds(3):ubounds(3),nosize) :: xin
REAL, INTENT(OUT), DIMENSION(lbounds(1)+1:ubounds(1),lbounds(2):ubounds(2),&
                            & lbounds(3)+1:ubounds(3),nosize) :: xout
```

File
array_interp.f90

Type
Module subroutine

Purpose
Interpolate a section of a model grid C-node array to the UW-nodes.

Reference
Section 10.2

Arguments

xin	Source array at the C-nodes
xout	Destination array at the UW-nodes
intdest	Selects valid points at the destination node
	1: coastal boundaries, interior points and open boundaries only
	2: interior points only
	3: interior points and open boundaries only
	4: coastal boundaries and interior points only
lbounds	Lower bounds of the source array
ubounds	Upper bounds of the source array
nosize	Fourth dimension of the source array
ivarid	Variable key id (used for log info only, zero for undefined)

info	Disables/enables writing of a log info.
outflag	Output flag for invalid points, if present. Zero otherwise.
hregular	Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.
vregular	Flag to select uniform or area averaging in the vertical, if present. Otherwise, the type of averaging is selected by iopt_arrint_vreg.

Carr_at_V

```
SUBROUTINE Carr_at_V(xin,xout,intsrce,intdest,lbounds,ubounds,&
                  & nosize,ivarid,info,outflag,hregular)
LOGICAL, INTENT(IN) :: info
LOGICAL, INTENT(IN), OPTIONAL :: hregular
INTEGER, INTENT(IN) :: intdest, intsrce, ivarid, nosize
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
REAL, INTENT(IN), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                           & lbounds(3):ubounds(3),nosize) :: xin
REAL, INTENT(OUT), DIMENSION(lbounds(1):ubounds(1),lbounds(2)+1:ubounds(2),&
                            & lbounds(3):ubounds(3),nosize) :: xout
```

File

array_interp.f90

Type

Module subroutine

Purpose

Interpolate a section of a model grid C-node array to the V-nodes.

Reference

Section 10.2

Arguments

xin	Source array at the C-nodes
xout	Destination array at the V-nodes
intsrce	Selects valid points at the source node
	0: all points

1: wet points only

intdest — Selects valid points at the destination node

0: all points

1: coastal boundaries, interior points and open boundaries only

2: interior points only

3: interior points and open boundaries only

4: coastal boundaries and interior points only

lbounds — Lower bounds of the source array

ubounds — Upper bounds of the source array

nosize — Fourth dimension of the source array

ivarid — Variable key id (used for log info only, zero for undefined)

info — Disables/enables writing of a log info.

outflag — Output flag for invalid points, if present. Zero otherwise.

hregular — Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.

Carr_at_VW

```
SUBROUTINE Carr_at_VW(xin,xout,intdest,lbounds,ubounds,nosize,&
                    & ivarid,info,outflag,hregular,vregular)
LOGICAL, INTENT(IN) :: info
LOGICAL, INTENT(IN), OPTIONAL :: hregular, vregular
INTEGER, INTENT(IN) :: intdest, ivarid, nosize
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
REAL, INTENT(IN), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                           & lbounds(3):ubounds(3),nosize) :: xin
REAL, INTENT(OUT), DIMENSION(lbounds(1):ubounds(1),lbounds(2)+1:ubounds(2),&
                            & lbounds(3)+1:ubounds(3),nosize) :: xout
```

File

array_interp.f90

Type

Module subroutine

Purpose

Interpolate a section of a model grid C-node array to the VW-nodes.

Reference

Section 10.2

Arguments

xin	Source array at the C-nodes
xout	Destination array at the VW-nodes
intdest	Selects valid points at the destination node 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
lbounds	Lower bounds of the source array
ubounds	Upper bounds of the source array
nosize	Fourth dimension of the source array
ivarid	Variable key id (used for log info only, zero for undefined)
info	Disables/enables writing of a log info.
outflag	Output flag for invalid points, if present. Zero otherwise.
hregular	Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.
vregular	Flag to select uniform or area averaging in the vertical, if present. Otherwise, the type of averaging is selected by iopt_arrint_vreg.

Carr_at_W

```
SUBROUTINE Carr_at_W(xin,xout,lbounds,ubounds,nosize,ivarid,&
                  & info,outflag,vregular)
LOGICAL, INTENT(IN) :: info
LOGICAL, INTENT(IN), OPTIONAL :: vregular
INTEGER, INTENT(IN) :: ivarid, nosize
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
```

```
REAL, INTENT(IN), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                          & lbounds(3):ubounds(3),nosize) :: xin
REAL, INTENT(OUT), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                           & lbounds(3)+1:ubounds(3),nosize) :: xout
```

File
array_interp.f90

Type
Module subroutine

Purpose
Interpolate a section of a model grid C-node array to the W-nodes.

Reference
Section 10.2

Arguments

xin	Source array at the C-nodes
xout	Destination array at the W-nodes
lbounds	Lower bounds of the source array
ubounds	Upper bounds of the source array
nosize	Fourth dimension of the source array
ivarid	Variable key id (used for log info only, zero for undefined)
info	Disables/enables writing of a log info.
outflag	Output flag for invalid points, if present. Zero otherwise.
vregular	Flag to select uniform or area averaging in the vertical, if present. Otherwise, the type of averaging is selected by iopt_arrint_vreg.

Uarr_at_C

```
SUBROUTINE Uarr_at_C(xin,xout,intsrce,intdest,lbounds,ubounds,&
                   & nosize,ivarid,info,outflag)
LOGICAL, INTENT(IN) :: info
INTEGER, INTENT(IN) :: intdest, intsrce, ivarid, nosize
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
REAL, INTENT(IN), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                          & lbounds(3):ubounds(3),nosize) :: xin
```

```
REAL, INTENT(OUT), DIMENSION(lbounds(1):ubounds(1)-1,lbounds(2):ubounds(2),&
                           & lbounds(3):ubounds(3),nosize) :: xout
```

File

array_interp.f90

Type

Module subroutine

Purpose

Interpolate a section of a model grid U-node array to the C-nodes.

Reference

Section 10.2

Arguments

xin	Source array at the U-nodes
xout	Destination array at the C-nodes
intsrce	Selects valid points at the source node 0: all points 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
intdest	Selects valid points at the destination node 0: all points 1: wet points only
lbounds	Lower bounds of the source array
ubounds	Upper bounds of the source array
nosize	Fourth dimension of the source array
ivarid	Variable key id (used for log info only, zero for undefined)
info	Disables/enables writing of a log info.
outflag	Output flag for invalid points, if present. Zero otherwise.

Uarr_at_UV

```
SUBROUTINE Uarr_at_UV(xin,xout,intsrce,intdest,lbounds,ubounds,&
                  & nosize,ivarid,info,outflag,hregular)
LOGICAL, INTENT(IN) :: info
LOGICAL, INTENT(IN), OPTIONAL :: hregular
INTEGER, INTENT(IN) :: intdest, intsrce, ivarid, nosize
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
REAL, INTENT(IN), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                           & lbounds(3):ubounds(3),nosize) :: xin
REAL, INTENT(OUT), DIMENSION(lbounds(1):ubounds(1),lbounds(2)+1:ubounds(2),&
                            & lbounds(3):ubounds(3),nosize) :: xout
```

File

array_interp.f90

Type

Module subroutine

Purpose

Interpolate a section of a model grid U-node array to the UV-nodes.

Reference

Section 10.2

Arguments

xin	Source array at the U-nodes
xout	Destination array at the UV-nodes
intsrce	Selects valid points at the source node
	0: all points
	1: coastal boundaries, interior points and open boundaries only
	2: interior points only
	3: interior points and open boundaries only
	4: coastal boundaries and interior points only
intdest	Selects valid points at the destination node
	0: all points
	1: interior points and open boundaries only

	2: interior points and UV-node open boundaries only
lbounds	Lower bounds of the source array
ubounds	Upper bounds of the source array
nosize	Fourth dimension of the source array
ivarid	Variable key id (used for log info only, zero for undefined)
info	Disables/enables writing of a log info.
outflag	Output flag for invalid points, if present. Zero otherwise.
hregular	Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.

Uarr_at_UW

```
SUBROUTINE Uarr_at_UW(xin,xout,intsrce,intdest,lbounds,ubounds,nosize,&
                    & ivarid,info,outflag,vregular)
LOGICAL, INTENT(IN) :: info
LOGICAL, INTENT(IN), OPTIONAL :: vregular
INTEGER, INTENT(IN) :: intdest, intsrce, ivarid, nosize
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
REAL, INTENT(IN), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                           & lbounds(3):ubounds(3),nosize) :: xin
REAL, INTENT(OUT), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                            & lbounds(3)+1:ubounds(3),nosize) :: xout
```

File
array_interp.f90

Type
Module subroutine

Purpose
Interpolate a section of a model grid U-node array to the UW-nodes.

Reference
Section 10.2

Arguments

xin	Source array at the U-nodes
xout	Destination array at the UW-nodes

intsrce	Selects valid points at the source node 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
intdest	Selects valid points at the destination node 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
lbounds	Lower bounds of the source array
ubounds	Upper bounds of the source array
nosize	Fourth dimension of the source array
ivarid	Variable key id (used for log info only, zero for undefined)
info	Disables/enables writing of a log info.
outflag	Output flag for invalid points, if present. Zero otherwise.
vregular	Flag to select uniform or area averaging in the vertical, if present. Otherwise, the type of averaging is selected by iopt_arrint_vreg.

Uarr_at_V

```
SUBROUTINE Uarr_at_V(xin,xout,intsrce,intdest,lbounds,ubounds,&
                  & nosize,ivarid,info,outflag,hregular)
LOGICAL, INTENT(IN) :: info
LOGICAL, INTENT(IN), OPTIONAL :: hregular
INTEGER, INTENT(IN) :: intdest, intsrce, ivarid, nosize
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
REAL, INTENT(IN), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                           & lbounds(3):ubounds(3),nosize) :: xin
REAL, INTENT(OUT), DIMENSION(lbounds(1):ubounds(1)-1,&
                            & lbounds(2)+1:ubounds(2),&
                            & lbounds(3):ubounds(3),nosize) :: xout
```

File
array_interp.f90

Type
Module subroutine

Purpose
Interpolate a section of a model grid U-node array to the V-nodes.

Reference
Section 10.2

Arguments

xin	Source array at the U-nodes
xout	Destination array at the V-nodes
intsrce	Selects valid points at the source node 0: all points 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
intdest	Selects valid points at the destination node 0: all points 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
lbounds	Lower bounds of the source array
ubounds	Upper bounds of the source array
nosize	Fourth dimension of the source array
ivarid	Variable key id (used for log info only, zero for undefined)
info	Disables/enables writing of a log info.
outflag	Output flag for invalid points, if present. Zero otherwise.
hregular	Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.

Uarr_at_W

```
SUBROUTINE Uarr_at_W(xin,xout,intsrce,lbounds,ubounds,nosize,&
                  & ivarid,info,outflag,vregular)
LOGICAL, INTENT(IN) :: info
LOGICAL, INTENT(IN), OPTIONAL :: vregular
INTEGER, INTENT(IN) :: intsrce, ivarid, nosize
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
REAL, INTENT(IN), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                           & lbounds(3):ubounds(3),nosize) :: xin
REAL, INTENT(OUT), DIMENSION(lbounds(1):ubounds(1)-1,lbounds(2):ubounds(2),&
                            & lbounds(3)+1:ubounds(3),nosize) :: xout
```

File

array_interp.f90

Type

Module subroutine

Purpose

Interpolate a section of a model grid U-node array to the W-nodes.

Reference

Section 10.2

Arguments

xin	Source array at the U-nodes
xout	Destination array at the W-nodes
intsrce	Selects valid points at the source node
	1: coastal boundaries, interior points and open boundaries only
	2: interior points only
	3: interior points and open boundaries only
	4: coastal boundaries and interior points only
lbounds	Lower bounds of the source array
ubounds	Upper bounds of the source array
nosize	Fourth dimension of the source array
ivarid	Variable key id (used for log info only, zero for undefined)

info	Disables/enables writing of a log info.
outflag	Output flag for invalid points, if present. Zero otherwise.
vregular	Flag to select uniform or area averaging in the vertical, if present. Otherwise, the type of averaging is selected by iopt_arrint_vreg.

UVarr_at_C

```
SUBROUTINE UVarr_at_C(xin,xout,intsrce,intdest,lbounds,ubounds,&
                    & nosize,ivarid,info,outflag)
LOGICAL, INTENT(IN) :: info
INTEGER, INTENT(IN) :: intdest, intsrce, ivarid, nosize
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
REAL, INTENT(IN), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                           & lbounds(3):ubounds(3),nosize) :: xin
REAL, INTENT(OUT), DIMENSION(lbounds(1):ubounds(1)-1,lbounds(2):ubounds(2)-1,&
                            & lbounds(3):ubounds(3),nosize) :: xout
```

File

array_interp.f90

Type

Module subroutine

Purpose

Interpolate a section of a model grid UV-node array to the C-nodes.

Reference

Section 10.2

Arguments

xin	Source array at the UV-nodes
xout	Destination array at the C-nodes
intsrce	Selects valid points at the source node
	0: all points
	1: wet points only
intdest	Selects valid points at the destination node
	0: all points

	1: wet points
lbounds	Lower bounds of the source array
ubounds	Upper bounds of the source array
nosize	Fourth dimension of the source array
ivarid	Variable key id (used for log info only, zero for undefined)
info	Disables/enables writing of a log info.
outflag	Output flag for invalid points, if present. Zero otherwise.

UVarr_at_U

```
SUBROUTINE UVarr_at_U(xin,xout,intsrce,intdest,lbounds,ubounds,&
                   & nosize,ivarid,info,outflag)
LOGICAL, INTENT(IN) :: info
INTEGER, INTENT(IN) :: intdest, intsrce, ivarid, nosize
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
REAL, INTENT(IN), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                           & lbounds(3):ubounds(3),nosize) :: xin
REAL, INTENT(OUT), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2)-1,&
                            & lbounds(3):ubounds(3),nosize) :: xout
```

File

array_interp.f90

Type

Module subroutine

Purpose

Interpolate a section of a model grid UV-node array to the U-nodes.

Reference

Section 10.2

Arguments

xin	Source array at the UV-nodes
xout	Destination array at the U-nodes
intsrce	Selects valid points at the source node
	0: all points
	1: interior points and open boundaries only

	2: interior points and UV-node open boundaries only
intdest	Selects valid points at the destination node
	0: all points
	1: coastal boundaries, interior points and open boundaries only
	2: interior points only
	3: interior points and open boundaries only
	4: coastal boundaries and interior points only
lbounds	Lower bounds of the source array
ubounds	Upper bounds of the source array
nosize	Fourth dimension of the source array
ivarid	Variable key id (used for log info only, zero for undefined)
info	Disables/enables writing of a log info.
outflag	Output flag for invalid points, if present. Zero otherwise.

UVarr_at_V

```
SUBROUTINE UVarr_at_V(xin,xout,intsrce,intdest,lbounds,ubounds,nosize,&
                    & ivarid,info,outflag)
LOGICAL, INTENT(IN) :: info
INTEGER, INTENT(IN) :: intdest, intsrce, ivarid, nosize
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
REAL, INTENT(IN), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                           & lbounds(3):ubounds(3),nosize) :: xin
REAL, INTENT(OUT), DIMENSION(lbounds(1):ubounds(1)-1,lbounds(2):ubounds(2),&
                            & lbounds(3):ubounds(3),nosize) :: xout
```

File
array_interp.f90

Type
Module subroutine

Purpose
Interpolate a section of a model grid UV-node array to the V-nodes.

Reference
Section 10.2

Arguments

xin	Source array at the UV-nodes
xout	Destination array at the V-nodes
intsrce	Selects valid points at the source node
	0: all points
	1: interior points and open boundaries only
	2: interior points and UV-node open boundaries only
intdest	Selects valid points at the destination node
	0: all points
	1: coastal boundaries, interior points and open
	2: interior points only
	3: interior points and open boundaries only
	4: coastal boundaries and interior points only
lbounds	Lower bounds of the source array
ubounds	Upper bounds of the source array
nosize	Fourth dimension of the source array
ivarid	Variable key id (used for log info only, zero for undefined)
info	Disables/enables writing of a log info.
outflag	Output flag for invalid points, if present. Zero otherwise.

UWarr_at_U

```
SUBROUTINE UWarr_at_U(xin,xout,intsrce,intdest,lbounds,ubounds,nosize,&
                  & ivarid,info,outflag)
LOGICAL, INTENT(IN) :: info
INTEGER, INTENT(IN) :: intdest, intsrce, ivarid, nosize
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
REAL, INTENT(IN), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                           & lbounds(3):ubounds(3),nosize) :: xin
REAL, INTENT(OUT), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                            & lbounds(3):ubounds(3)-1,nosize) :: xout
```

File

array_interp.f90

Type
Module subroutine

Purpose
Interpolate a section of a model grid UW-node array to the U-nodes.

Reference
Section 10.2

Arguments

xin	Source array at the UW-nodes
xout	Destination array at the U-nodes
intsrce	Selects valid points at the source node
	1: coastal boundaries, interior points and open boundaries only
	2: interior points only
	3: interior points and open boundaries only
	4: coastal boundaries and interior points only
intdest	Selects valid points at the destination node
	1: coastal boundaries, interior points and open boundaries only
	2: interior points only
	3: interior points and open boundaries only
	4: coastal boundaries and interior points only
lbounds	Lower bounds of the source array
ubounds	Upper bounds of the source array
nosize	Fourth dimension of the source array
ivarid	Variable key id (used for log info only, zero for undefined)
info	Disables/enables writing of a log info.
outflag	Output flag for invalid points, if present. Zero otherwise.

UWarr_at_W

```
SUBROUTINE UWarr_at_W(xin,xout,intsrce,lbounds,ubounds,nosize,&
                    & ivarid,info,outflag)
LOGICAL, INTENT(IN) :: info
```

```
INTEGER, INTENT(IN) :: intsrce, ivarid, nosize
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
REAL, INTENT(IN), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                           & lbounds(3):ubounds(3),nosize) :: xin
REAL, INTENT(OUT), DIMENSION(lbounds(1):ubounds(1)-1,lbounds(2):ubounds(2),&
                            & lbounds(3):ubounds(3),nosize) :: xout
```

File

array_interp.f90

Type

Module subroutine

Purpose

Interpolate a section of a model grid UW-node array to the W-nodes.

Reference

Section 10.2

Arguments

xin	Source array at the UW-nodes
xout	Destination array at the W-nodes
intsrce	Selects valid points at the source node 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
lbounds	Lower bounds of the source array
ubounds	Upper bounds of the source array
nosize	Fourth dimension of the source array
ivarid	Variable key id (used for log info only, zero for undefined)
info	Disables/enables writing of a log info.
outflag	Output flag for invalid points, if present. Zero otherwise.

Varr_at_C

```
SUBROUTINE Varr_at_C(xin,xout,intsrce,intdest,lbounds,ubounds,&
                  & nosize,ivarid,info,outflag)
LOGICAL, INTENT(IN) :: info
INTEGER, INTENT(IN) :: intdest, intsrce, ivarid, nosize
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
REAL, INTENT(IN), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                           & lbounds(3):ubounds(3),nosize) :: xin
REAL, INTENT(OUT), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2)-1,&
                            & lbounds(3):ubounds(3),nosize) :: xout
```

File

array_interp.f90

Type

Module subroutine

Purpose

Interpolate a section of a model grid V-node array to the C-nodes.

Reference

Section 10.2

Arguments

xin	Source array at the V-nodes
xout	Destination array at the C-nodes
intsrce	Selects valid points at the source node
	0: all points
	1: coastal boundaries, interior points and open boundaries only
	2: interior points only
	3: interior points and open boundaries only
	4: coastal boundaries and interior points only
intdest	Selects valid points at the destination node
	0: all points
	1: wet points only
lbounds	Lower bounds of the source array

ubounds	Upper bounds of the source array
nosize	Fourth dimension of the source array
ivarid	Variable key id (used for log info only, zero for undefined)
info	Disables/enables writing of a log info.
outflag	Output flag for invalid points, if present. Zero otherwise.

Varr_at_U

```
SUBROUTINE Varr_at_U(xin,xout,intsrce,intdest,lbounds,ubounds,&
                  & nosize,ivarid,info,outflag,hregular)
LOGICAL, INTENT(IN) :: info
LOGICAL, INTENT(IN), OPTIONAL :: hregular
INTEGER, INTENT(IN) :: intdest, intsrce, ivarid, nosize
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
REAL, INTENT(IN), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                           & lbounds(3):ubounds(3),nosize) :: xin
REAL, INTENT(OUT), DIMENSION(lbounds(1)+1:ubounds(1),&
                            & lbounds(2):ubounds(2)-1,&
                            & lbounds(3):ubounds(3),nosize) :: xout
```

File

array_interp.f90

Type

Module subroutine

Purpose

Interpolate a section of a model grid V-node array to the U-nodes.

Reference

Section 10.2

Arguments

xin	Source array at the V-nodes
xout	Destination array at the U-nodes
intsrce	Selects valid points at the source node
	0: all points
	1: coastal boundaries, interior points and open boundaries only

2: interior points only

3: interior points and open boundaries only

4: coastal boundaries and interior points only

intdest — Selects valid points at the destination node

0: all points

1: coastal boundaries, interior points and open boundaries only

2: interior points only

3: interior points and open boundaries only

4: coastal boundaries and interior points only

lbounds — Lower bounds of the source array

ubounds — Upper bounds of the source array

nosize — Fourth dimension of the source array

ivarid — Variable key id (used for log info only, zero for undefined)

info — Disables/enables writing of a log info.

outflag — Output flag for invalid points, if present. Zero otherwise.

hregular — Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.

Varr_at_UV

```
SUBROUTINE Varr_at_UV(xin,xout,intsrce,intdest,lbounds,ubounds,&
                  & nosize,ivarid,info,outflag,hregular)
LOGICAL, INTENT(IN) :: info
LOGICAL, INTENT(IN), OPTIONAL :: hregular
INTEGER, INTENT(IN) :: intdest, intsrce, ivarid, nosize
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
REAL, INTENT(IN), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                         & lbounds(3):ubounds(3),nosize) :: xin
REAL, INTENT(OUT), DIMENSION(lbounds(1)+1:ubounds(1),lbounds(2):ubounds(2),&
                          & lbounds(3):ubounds(3),nosize) :: xout
```

File

array_interp.f90

Type
Module subroutine

Purpose
Interpolate a section of a model grid V-node array to the UV-nodes.

Reference
Section 10.2

Arguments

xin	Source array at the V-nodes
xout	Destination array at the UV-nodes
intsrce	Selects valid points at the source node 0: all points 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
intdest	Selects valid points at the destination node 0: all points 1: interior points and open boundaries only 2: interior points and UV-node open boundaries only
lbounds	Lower bounds of the source array
ubounds	Upper bounds of the source array
nosize	Fourth dimension of the source array
ivarid	Variable key id (used for log info only, zero for undefined)
info	Disables/enables writing of a log info.
outflag	Output flag for invalid points, if present. Zero otherwise.
hregular	Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.

Varr_at_VW

```
SUBROUTINE Varr_at_VW(xin,xout,intsrce,intdest,lbounds,ubounds,&
                    & nosize,ivarid,info,outflag,vregular)
LOGICAL, INTENT(IN) :: info
LOGICAL, INTENT(IN), OPTIONAL :: vregular
INTEGER, INTENT(IN) :: intdest, intsrce, ivarid, nosize
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
REAL, INTENT(IN), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                           & lbounds(3):ubounds(3),nosize) :: xin
REAL, INTENT(OUT), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                            & lbounds(3)+1:ubounds(3),nosize) :: xout
```

File

array_interp.f90

Type

Module subroutine

Purpose

Interpolate a section of a model grid V-node array to the VW-nodes.

Reference

Section 10.2

Arguments

xin	Source array at the V-nodes
xout	Destination array at the VW-nodes
intsrce	Selects valid points at the source node
	1: coastal boundaries, interior points and open boundaries only
	2: interior points only
	3: interior points and open boundaries only
	4: coastal boundaries and interior points only
intdest	Selects valid points at the destination node
	1: coastal boundaries, interior points and open boundaries only
	2: interior points only

	3: interior points and open boundaries only
	4: coastal boundaries and interior points only
lbounds	Lower bounds of the source array
ubounds	Upper bounds of the source array
nosize	Fourth dimension of the source array
ivarid	Variable key id (used for log info only, zero for undefined)
info	Disables/enables writing of a log info.
outflag	Output flag for invalid points, if present. Zero otherwise.
vregular	Flag to select uniform or area averaging in the vertical, if present. Otherwise, the type of averaging is selected by iopt_arrint_vreg.

Varr_at_W

```
SUBROUTINE Varr_at_W(xin,xout,intsrce,lbounds,ubounds,nosize,&
                  & ivarid,info,outflag,vregular)
LOGICAL, INTENT(IN) :: info
LOGICAL, INTENT(IN), OPTIONAL :: vregular
INTEGER, INTENT(IN) :: intsrce, ivarid, nosize
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
REAL, INTENT(IN), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                           & lbounds(3):ubounds(3),nosize) :: xin
REAL, INTENT(OUT), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2)-1,&
                            & lbounds(3)+1:ubounds(3),nosize) :: xout
```

File

array_interp.f90

Type

Module subroutine

Purpose

Interpolate a section of a model grid V-node array to the W-nodes.

Reference

Section 10.2

Arguments

xin	Source array at the V-nodes

xout	Destination array at the W-nodes
intsrce	Selects valid points at the source node 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
lbounds	Lower bounds of the source array
ubounds	Upper bounds of the source array
nosize	Fourth dimension of the source array
ivarid	Variable key id (used for log info only, zero for undefined)
info	Disables/enables writing of a log info.
outflag	Output flag for invalid points, if present. Zero otherwise.
vregular	Flag to select uniform or area averaging in the vertical, if present. Otherwise, the type of averaging is selected by iopt_arrint_vreg.

VWarr_at_V

```
SUBROUTINE VWarr_at_V(xin,xout,intsrce,intdest,lbounds,ubounds,&
                   & nosize,ivarid,info,outflag)
LOGICAL, INTENT(IN) :: info
INTEGER, INTENT(IN) :: intdest, intsrce, ivarid, nosize
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
REAL, INTENT(IN), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                           & lbounds(3):ubounds(3),nosize) :: xin
REAL, INTENT(OUT), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                            & lbounds(3):ubounds(3)-1,nosize) :: xout
```

File

array_interp.f90

Type

Module subroutine

Purpose

Interpolate a section of a model grid VW-node array to the V-nodes.

Reference
Section 10.2

Arguments

xin	Source array at the VW-nodes
xout	Destination array at the V-nodes
intsrce	Selects valid points at the source node
	1: coastal boundaries, interior points and open boundaries only
	2: interior points only
	3: interior points and open boundaries only
	4: coastal boundaries and interior points only
intdest	Selects valid points at the destination node
	1: coastal boundaries, interior points and open boundaries only
	2: interior points only
	3: interior points and open boundaries only
	4: coastal boundaries and interior points only
lbounds	Lower bounds of the source array
ubounds	Upper bounds of the source array
nosize	Fourth dimension of the source array
ivarid	Variable key id (used for log info only, zero for undefined)
info	Disables/enables writing of a log info.
outflag	Output flag for invalid points, if present. Zero otherwise.

VWarr_at_W

```
SUBROUTINE VWarr_at_W(xin,xout,intsrce,lbounds,ubounds,nosize,&
                  & ivarid,info,outflag)
LOGICAL, INTENT(IN) :: info
INTEGER, INTENT(IN) :: intsrce, ivarid, nosize
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
REAL, INTENT(IN), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                         & lbounds(3):ubounds(3),nosize) :: xin
REAL, INTENT(OUT), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2)-1,&
                          & lbounds(3):ubounds(3),nosize) :: xout
```

File
array_interp.f90

Type
Module subroutine

Purpose
Interpolate a section of a model grid VW-node array to the W-nodes.

Reference
Section 10.2

Arguments

xin	Source array at the VW-nodes
xout	Destination array at the W-nodes
intsrce	Selects valid points at the source node
	1: coastal boundaries, interior points and open boundaries only
	2: interior points only
	3: interior points and open boundaries only
	4: coastal boundaries and interior points only
lbounds	Lower bounds of the source array
ubounds	Upper bounds of the source array
nosize	Fourth dimension of the source array
ivarid	Variable key id (used for log info only, zero for undefined)
info	Disables/enables writing of a log info.
outflag	Output flag for invalid points, if present. Zero otherwise.

Warr_at_C

```
SUBROUTINE Carr_at_W(xin,xout,lbounds,ubounds,nosize,ivarid,&
                  & info,outflag)
LOGICAL, INTENT(IN) :: info
INTEGER, INTENT(IN) :: ivarid, nosize
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
REAL, INTENT(IN), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                         & lbounds(3):ubounds(3),nosize) :: xin
```

```
REAL, INTENT(OUT), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                            & lbounds(3):ubounds(3)-1,nosize) :: xout
```

File
array_interp.f90

Type
Module subroutine

Purpose
Interpolate a section of a model grid W-node array to the C-nodes.

Reference
Section 10.2

Arguments

xin	Source array at the W-nodes
xout	Destination array at the C-nodes
lbounds	Lower bounds of the source array
ubounds	Upper bounds of the source array
nosize	Fourth dimension of the source array
ivarid	Variable key id (used for log info only, zero for undefined)
info	Disables/enables writing of a log info.
outflag	Output flag for invalid points, if present. Zero otherwise.

Warr_at_U

```
SUBROUTINE
Warr_at_U(xin,xout,intdest,lbounds,ubounds,nosize,ivarid,&
       & info,outflag,hregular)
LOGICAL, INTENT(IN) :: info
LOGICAL, INTENT(IN), OPTIONAL :: hregular
INTEGER, INTENT(IN) :: intdest, ivarid, nosize
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
REAL, INTENT(IN), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                           & lbounds(3):ubounds(3),nosize) :: xin
REAL, INTENT(OUT), DIMENSION(lbounds(1)+1:ubounds(1),lbounds(2):ubounds(2),&
                            & lbounds(3):ubounds(3)-1,nosize) :: xout
```

File
array_interp.f90

Type
Module subroutine

Purpose
Interpolate a section of a model grid W-node array to the U-nodes.

Reference
Section 10.2

Arguments

xin	Source array at the W-nodes
xout	Destination array at the U-nodes
intdest	Selects valid points at the destination node 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
lbounds	Lower bounds of the source array
ubounds	Upper bounds of the source array
nosize	Fourth dimension of the source array
ivarid	Variable key id (used for log info only, zero for undefined)
info	Disables/enables writing of a log info.
outflag	Output flag for invalid points, if present. Zero otherwise.
hregular	Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.

Warr_at_UW

```
SUBROUTINE Warr_at_UW(xin,xout,intdest,lbounds,ubounds,nosize,&
                  & ivarid,info,outflag,hregular)
LOGICAL, INTENT(IN) :: info
LOGICAL, INTENT(IN), OPTIONAL :: hregular
INTEGER, INTENT(IN) :: intdest, ivarid, nosize
```

```
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
REAL, INTENT(IN), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                           & lbounds(3):ubounds(3),nosize) :: xin
REAL, INTENT(OUT), DIMENSION(lbounds(1)+1:ubounds(1),lbounds(2):ubounds(2),&
                            & lbounds(3):ubounds(3),nosize) :: xout
```

File

array_interp.f90

Type

Module subroutine

Purpose

Interpolate a section of a model grid W-node array to the UW-nodes.

Reference

Section 10.2

Arguments

xin	Source array at the W-nodes
xout	Destination array at the UW-nodes
intdest	Selects valid points at the destination node 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
lbounds	Lower bounds of the source array
ubounds	Upper bounds of the source array
nosize	Fourth dimension of the source array
ivarid	Variable key id (used for log info only, zero for undefined)
info	Disables/enables writing of a log info.
outflag	Output flag for invalid points, if present. Zero otherwise.
hregular	Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.

Warr_at_V

```
SUBROUTINE
Warr_at_V(xin,xout,intdest,lbounds,ubounds,nosize,ivarid,&
       & info,outflag,hregular)
LOGICAL, INTENT(IN) :: info
LOGICAL, INTENT(IN), OPTIONAL :: hregular
INTEGER, INTENT(IN) :: intdest, ivarid, nosize
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
REAL, INTENT(IN), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                           & lbounds(3):ubounds(3),nosize) :: xin
REAL, INTENT(OUT), DIMENSION(lbounds(1):ubounds(1),lbounds(2)+1:ubounds(2),&
                            & lbounds(3):ubounds(3)-1,nosize) :: xout
```

File

array_interp.f90

Type

Module subroutine

Purpose

Interpolate a section of a model grid W-node array to the V-nodes.

Reference

Section 10.2

Arguments

xin	Source array at the W-nodes
xout	Destination array at the V-nodes
intdest	Selects valid points at the destination node
	1: coastal boundaries, interior points and open boundaries only
	2: interior points only
	3: interior points and open boundaries only
	4: coastal boundaries and interior points only
lbounds	Lower bounds of the source array
ubounds	Upper bounds of the source array
nosize	Fourth dimension of the source array
ivarid	Variable key id (used for log info only, zero for undefined)

info	Disables/enables writing of a log info.
outflag	Output flag for invalid points, if present. Zero otherwise.
hregular	Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.

Warr_at_VW

```
SUBROUTINE Warr_at_VW(xin,xout,intdest,lbounds,ubounds,nosize,&
                   & ivarid,info,outflag,hregular)
LOGICAL, INTENT(IN) :: info
LOGICAL, INTENT(IN), OPTIONAL :: hregular
INTEGER, INTENT(IN) :: intdest, ivarid, nosize
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
REAL, INTENT(IN), DIMENSION(lbounds(1):ubounds(1),lbounds(2):ubounds(2),&
                           & lbounds(3):ubounds(3),nosize) :: xin
REAL, INTENT(OUT), DIMENSION(lbounds(1):ubounds(1),lbounds(2)+1:ubounds(2),&
                            & lbounds(3):ubounds(3),nosize) :: xout
```

File

array_interp.f90

Type

Module subroutine

Purpose

Interpolate a section of a model grid W-node array to the VW-nodes.

Reference

Section 10.2

Arguments

xin	Source array at the W-nodes
xout	Destination array at the VW-nodes
intdest	Selects valid points at the destination node

1: coastal boundaries, interior points and open boundaries only

2: interior points only

3: interior points and open boundaries only

	4: coastal boundaries and interior points only
lbounds	Lower bounds of the source array
ubounds	Upper bounds of the source array
nosize	Fourth dimension of the source array
ivarid	Variable key id (used for log info only, zero for undefined)
info	Disables/enables writing of a log info.
outflag	Output flag for invalid points, if present. Zero otherwise.
hregular	Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.

Cvar_at_U

```
FUNCTION Cvar_at_U(xin,i,j,k,intsrce,intdest,outflag,hregular)
LOGICAL, INTENT(IN), OPTIONAL :: hregular
INTEGER, INTENT(IN) :: i, intdest, intsrce, j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2) :: xin
REAL :: Cvar_at_U
```

File
array_interp.f90

Type
Module function

Purpose
Interpolate a C-node variable to a U-nodal point.

Reference
Section 10.2

Arguments

xin	Source array at the C-nodes
i	Lower X-index of the source array
j	Y-index of the source array
k	Vertical index of the source array
intsrce	Selects valid points at the source node
	0: all points

1: wet points only

intdest — Selects valid points at the destination node

0: all points

1: coastal boundaries, interior points and open boundaries only

2: interior points only

3: interior points and open boundaries only

4: coastal boundaries and interior points only

outflag — Output flag for invalid points, if present. Zero otherwise.

hregular — Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.

Cvar_at_U — Interpolated value at the U-node destination point

Cvar_at_UV

```
FUNCTION Cvar_at_UV(xin,i,j,k,intsrce,intdest,outflag,hregular)
LOGICAL, INTENT(IN), OPTIONAL :: hregular
INTEGER, INTENT(IN) :: i, intdest, intsrce, j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2,2) :: xin
REAL :: Cvar_at_UV
```

File

array_interp.f90

Type

Module function

Purpose

Interpolate a C-node variable to a UV-nodal point.

Reference

Section 10.2

Arguments

xin — Source array at the C-nodes

i — Lower X-index of the source array

j — Lower Y-index of the source array

k	Vertical index of the source array
intsrce	Selects valid points at the source node
	0: all points
	1: wet points only
intdest	Selects valid points at the destination node
	0: all points
	1: wet points only
outflag	Output flag for invalid points, if present. Zero otherwise.
hregular	Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.
Cvar_at_UV	Interpolated value at the UV-node destination point

Cvar_at_UW

```
FUNCTION Cvar_at_UW(xin,i,j,k,intdest,outflag,hregular,&
                  & vregular)
LOGICAL, INTENT(IN), OPTIONAL :: hregular, vregular
INTEGER, INTENT(IN) :: i, intdest, j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2,2) :: xin
REAL :: Cvar_at_UW
```

File

array_interp.f90

Type

Module function

Purpose

Interpolate a C-node variable to a UW-nodal point.

Reference

Section 10.2

Arguments

xin	Source array at the C-nodes
i	Lower X-index of the source array
j	Y-index of the source array

k	Lower vertical index of the source array
intdest	Selects valid points at the destination node 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
outflag	Output flag for invalid points, if present. Zero otherwise.
hregular	Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.
vregular	Flag to select uniform or area averaging in the vertical, if present. Otherwise, the type of averaging is selected by iopt_arrint_vreg.
Cvar_at_UW	Interpolated value at the UW-node destination point

Cvar_at_V

```
FUNCTION Cvar_at_V(xin,i,j,k,intsrce,intdest,outflag,hregular)
LOGICAL, INTENT(IN), OPTIONAL :: hregular
INTEGER, INTENT(IN) :: i, intdest, intsrce, j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2) :: xin
REAL :: Cvar_at_V
```

File

array_interp.f90

Type

Module function

Purpose

Interpolate a C-node variable to a V-nodal point.

Reference

Section 10.2

Arguments

xin	Source array at the C-nodes

i	X-index of the source array
j	Lower Y-index of the source array
k	Vertical index of the source array
intsrce	Selects valid points at the source node
	0: all points
	1: wet points only
intdest	Selects valid points at the destination node
	0: all points
	1: coastal boundaries, interior points and open boundaries only
	2: interior points only
	3: interior points and open boundaries only
	4: coastal boundaries and interior points only
outflag	Output flag for invalid points, if present. Zero otherwise.
hregular	Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.
Cvar_at_V	Interpolated value at the V-node destination point

Cvar_at_VW

```
FUNCTION Cvar_at_VW(xin,i,j,k,intdest,outflag,hregular,&
                 & vregular)
LOGICAL, INTENT(IN), OPTIONAL :: hregular, vregular
INTEGER, INTENT(IN) :: i, intdest, j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2,2) :: xin
REAL :: Cvar_at_VW
```

File
array_interp.f90

Type
Module function

Purpose
Interpolate a C-node variable to a VW-nodal point.

Reference
Section 10.2

Arguments

xin	Source array at the C-nodes
i	X-index of the source array
j	Lower Y-index of the source array
k	Lower vertical index of the source array
intdest	Selects valid points at the destination node 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
outflag	Output flag for invalid points, if present. Zero otherwise.
hregular	Flag to select uniform or area averaging in the horizontal. Otherwise, the type of averaging is selected by iopt_arrint_hreg.
vregular	Flag to select uniform or area averaging in the vertical, if present. Otherwise, the type of averaging is selected by iopt_arrint_vreg.
Cvar_at_VW	Interpolated value at the VW-node destination point

Cvar_at_W

```
FUNCTION Cvar_at_W(xin,i,j,k,outflag,vregular)
LOGICAL, INTENT(IN), OPTIONAL :: vregular
INTEGER, INTENT(IN) :: i, j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2) :: xin
REAL :: Cvar_at_W
```

File
array_interp.f90

Type
Module function

Purpose
Interpolate a C-node variable to a W-nodal point.

Reference
Section 10.2

Arguments

xin	Source array at the C-nodes
i	X-index of the source array
j	Y-index of the source array
k	Lower vertical index of the source array
outflag	Output flag for invalid points, if present. Zero otherwise.
vregular	Flag to select uniform or area averaging in the vertical, if present. Otherwise, the type of averaging is selected by iopt_arrint_vreg.
Cvar_at_W	Interpolated value at the W-node destination point

Uvar_at_C

```
FUNCTION Uvar_at_C(xin,i,j,k,intsrce,intdest,outflag)
INTEGER, INTENT(IN) :: i, intdest, intsrce, j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2) :: xin
REAL :: Uvar_at_C
```

File
array_interp.f90

Type
Module function

Purpose
Interpolate a U-node variable to a C-nodal point.

Reference
Section 10.2

Arguments

xin	Source array at the U-nodes
i	Lower X-index of the source array
j	Y-index of the source array
k	Vertical index of the source array

intsrce	Selects valid points at the source node 0: all points 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
intdest	Selects valid points at the destination node 0: all points 1: wet points only
outflag	Output flag for invalid points, if present. Zero otherwise.
Uvar_at_C	Interpolated value at the C-node destination point

Uvar_at_UV

```
FUNCTION Uvar_at_UV(xin,i,j,k,intsrce,intdest,outflag,hregular)
LOGICAL, INTENT(IN), OPTIONAL :: hregular
INTEGER, INTENT(IN) :: i, intdest, intsrce, j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2) :: xin
REAL :: Uvar_at_UV
```

File
array_interp.f90

Type
Module function

Purpose
Interpolate a U-node variable to a UV-nodal point.

Reference
Section 10.2

Arguments

xin	Source array at the U-nodes
i	X-index of the source array
j	Lower Y-index of the source array

k	Vertical index of the source array
intsrce	Selects valid points at the source node 0: all points 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
intdest	Selects valid points at the destination node 0: all points 1: interior points and open boundaries only 2: interior points and UV-node open boundaries only
outflag	Output flag for invalid points, if present. Zero otherwise.
hregular	Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.
Uvar_at_UV	Interpolated value at the UV-node destination point

Uvar_at_UW

```
FUNCTION Uvar_at_UW(xin,i,j,k,intsrce,intdest,outflag,vregular)
LOGICAL, INTENT(IN), OPTIONAL :: vregular
INTEGER, INTENT(IN) :: i, intdest, intsrce,j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2) :: xin
REAL :: Uvar_at_UW
```

File
array_interp.f90

Type
Module function

Purpose
Interpolate a U-node variable to a UW-nodal point.

Reference
Section 10.2

Arguments

xin	Source array at the U-nodes
i	X-index of the source array
j	Y-index of the source array
k	Lower vertical index of the source array
intsrce	Selects valid points at the source node 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
intdest	Selects valid points at the destination node 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
outflag	Output flag for invalid points, if present. Zero otherwise.
vregular	Flag to select uniform or area averaging in the vertical, if present. Otherwise, the type of averaging is selected by iopt_arrint_vreg.
Uvar_at_UW	Interpolated value at the UW-node destination point

Uvar_at_V

```
FUNCTION Uvar_at_V(xin,i,j,k,intsrce,intdest,outflag,hregular)
LOGICAL, INTENT(IN), OPTIONAL :: hregular
INTEGER, INTENT(IN) :: i, intdest, intsrce, j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2,2) :: xin
REAL :: Uvar_at_V
```

File

array_interp.f90

Type

Module function

Purpose

Interpolate a U-node variable to a V-nodal point.

Reference

Section 10.2

Arguments

xin	Source array at the U-nodes
i	Lower X-index of the source array
j	Lower Y-index of the source array
k	Vertical index of the source array
intsrce	Selects valid points at the source node 0: all points! 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
intdest	Selects valid points at the destination node 0: all points 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
outflag	Output flag for invalid points, if present. Zero otherwise.
hregular	Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.
Uvar_at_V	Interpolated value at the V-node destination point

Uvar_at_W

```
FUNCTION Uvar_at_W(xin,i,j,k,intsrce,outflag,vregular)
LOGICAL, INTENT(IN), OPTIONAL :: vregular
INTEGER, INTENT(IN) :: i, intsrce, j, k
```

```
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2,2) :: xin
REAL :: Uvar_at_W
```

File
array_interp.f90

Type
Module function

Purpose
Interpolate a U-node variable to a W-nodal point.

Reference
Section 10.2

Arguments

xin	Source array at the U-nodes
i	Lower X-index of the source array
j	Y-index of the source array
k	Lower vertical index of the source array
intsrce	Selects valid points at the source node 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
outflag	Output flag for invalid points, if present. Zero otherwise.
vregular	Flag to select uniform or area averaging in the vertical, if present. Otherwise, the type of averaging is selected by iopt_arrint_vreg.
Uvar_at_W	Interpolated value at the W-node destination point

UVvar_at_C

```
FUNCTION UVvar_at_C(xin,i,j,k,intsrce,intdest,outflag)
INTEGER, INTENT(IN) :: i, intdest, intsrce, j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2,2) :: xin
REAL :: UVvar_at_C
```

File
array_interp.f90

Type
Module function

Purpose
Interpolate a UV-node variable to a C-nodal point.

Reference
Section 10.2

Arguments

xin	Source array at the U-nodes
i	Lower X-index of the source array
j	Lower Y-index of the source array
k	Vertical index of the source array
intsrce	Selects valid points at the source node
	0: all points
	1: wet points only
intdest	Selects valid points at the destination node
	0: all points
	1: wet points only
outflag	Output flag for invalid points, if present. Zero otherwise.
UVvar_at_C	Interpolated value at the C-node destination point

UVvar_at_U

```
FUNCTION UVvar_at_U(xin,i,j,k,intsrce,intdest,outflag)
INTEGER, INTENT(IN) :: i, intdest, intsrce, j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2) :: xin
REAL :: UVvar_at_U
```

File
array_interp.f90

Type
Module function

Purpose
Interpolate a UV-node variable to a U-nodal point.

Reference
Section 10.2

Arguments

xin	Source array at the UV-nodes
i	X-index of the source array
j	Lower Y-index of the source array
k	Vertical index of the source array
intsrce	Selects valid points at the source node
	0: all points
	1: interior points and open boundaries only
	2: interior points and UV-node open boundaries only
intdest	Selects valid points at the destination node
	0: all points
	1: coastal boundaries, interior points and open boundaries only
	2: interior points only
	3: interior points and open boundaries only
	4: coastal boundaries and interior points only
outflag	Output flag for invalid points, if present. Zero otherwise.
UVvar_at_U	Interpolated value at the U-node destination point

UVvar_at_V

```
FUNCTION UVvar_at_V(xin,i,j,k,intsrce,intdest,outflag)
INTEGER, INTENT(IN) :: i, intdest, intsrce, j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2) :: xin
REAL :: UVvar_at_V
```

File
array_interp.f90

Type
Module function

Purpose
Interpolate a UV-node variable to a V-nodal point.

Reference
Section 10.2

Arguments

xin	Source array at the UV-nodes
i	Lower X-index of the source array
j	Y-index of the source array
k	Vertical index of the source array
intsrce	Selects valid points at the source node
	0: all points
	1: interior points and open boundaries only
	2: interior points and UV-node open boundaries only
intdest	Selects valid points at the source node
	0: all points
	1: coastal boundaries, interior points and open
	2: interior points only
	3: interior points and open boundaries only
	4: coastal boundaries and interior points only
outflag	Output flag for invalid points, if present. Zero otherwise.
UVvar_at_V	Interpolated value at the V-node destination point

UWvar_at_U

```
FUNCTION UWvar_at_U(xin,i,j,k,intsrce,intdest,outflag)
INTEGER, INTENT(IN) :: i, intdest, intsrce, j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2) :: xin
REAL :: UWvar_at_U
```

File
array_interp.f90

Type
Module function

Purpose

Interpolate a UW-node variable to a U-nodal point.

Reference

Section 10.2

Arguments

xin	Source array at the UW-nodes
i	X-index of the source array
j	Y-index of the source array
k	Lower vertical index of the source array
intsrce	Selects valid points at the source node
	1: coastal boundaries, interior points and open boundaries only
	2: interior points only
	3: interior points and open boundaries only
	4: coastal boundaries and interior points only
intdest	Selects valid points at the destination node
	1: coastal boundaries, interior points and open boundaries only
	2: interior points only
	3: interior points and open boundaries only
	4: coastal boundaries and interior points only
outflag	Output flag for invalid points, if present. Zero otherwise.
UWvar_at_U	Interpolated value at the U-node destination point

UWvar_at_W

```
FUNCTION UWvar_at_W(xin,i,j,k,intsrce,outflag)
INTEGER, INTENT(IN) :: i, intsrce, j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2) :: xin
REAL :: UWvar_at_W
```

File

array_interp.f90

Type
Module function

Purpose
Interpolate a UW-node variable to a W-nodal point.

Reference
Section 10.2

Arguments

xin	Source array at the UW-nodes
i	Lower X-index of the source array
j	Y-index of the source array
k	Vertical index of the source array
intsrce	Selects valid points at the source node
	1: coastal boundaries, interior points and open boundaries only
	2: interior points only
	3: interior points and open boundaries only
	4: coastal boundaries and interior points only
outflag	Output flag for invalid points, if present. Zero otherwise.
UWvar_at_W	Interpolated value at the W-node destination point

Vvar_at_C

```
FUNCTION Vvar_at_C(xin,i,j,k,intsrce,intdest,outflag)
INTEGER, INTENT(IN) :: i, intdest, intsrce, j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2) :: xin
REAL :: Vvar_at_C
```

File
array_interp.f90

Type
Module function

Purpose
Interpolate a V-node variable to a C-nodal point.

Reference
Section 10.2

Arguments

xin	Source array at the V-nodes
i	X-index of the source array
j	Lower Y-index of the source array
k	Vertical index of the source array
intsrce	Selects valid points at the source node 0: all points 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
intdest	Selects valid points at the destination node 0: all points 1: wet points only
outflag	Output flag for invalid points, if present. Zero otherwise.
Vvar_at_C	Interpolated value at the C-node destination point

Vvar_at_U

```
FUNCTION Vvar_at_U(xin,i,j,k,intsrce,intdest,outflag,hregular)
LOGICAL, INTENT(IN), OPTIONAL :: hregular
INTEGER, INTENT(IN) :: i, intdest, intsrce, j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2,2) :: xin
REAL :: Vvar_at_U
```

File
array_interp.f90

Type
Module function

Purpose
Interpolate a V-node variable to a U-nodal point.

Reference
Section 10.2

Arguments

xin	Source array at the V-nodes
i	Lower X-index of the source array
j	Lower Y-index of the source array
k	Vertical index of the source array
intsrce	Selects valid points at the source node 0: all points 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
intdest	Selects valid points at the destination node 0: all points 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
outflag	Output flag for invalid points, if present. Zero otherwise.
hregular	Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.
Vvar_at_U	Interpolated value at the U-node destination point

Vvar_at_UV

```
FUNCTION Vvar_at_UV(xin,i,j,k,intsrce,intdest,outflag,hregular)
LOGICAL, INTENT(IN), OPTIONAL :: hregular
INTEGER, INTENT(IN) :: i, intdest, intsrce, j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2) :: xin
REAL :: Vvar_at_UV
```

File
array_interp.f90

Type
Module function

Purpose
Interpolate a V-node variable to a UV-nodal point.

Reference
Section 10.2

Arguments

xin	Source array at the V-nodes
i	Lower X-index of the source array
j	Y-index of the source array
k	Vertical index of the source array
intsrce	Selects valid points at the source node 0: all points 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
intdest	Selects valid points at the destination node 0: all points 1: interior points and open boundaries only 2: interior points and UV-open boundaries only
outflag	Output flag for invalid points, if present. Zero otherwise.
hregular	Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.
Vvar_at_UV	Interpolated value at the UV-node destination point

Vvar_at_VW

```
FUNCTION Vvar_at_VW(xin,i,j,k,intsrce,intdest,outflag,vregular)
LOGICAL, INTENT(IN), OPTIONAL :: vregular
INTEGER, INTENT(IN) :: i, intdest, intsrce, j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2) :: xin
REAL :: Vvar_at_VW
```

File

array_interp.f90

Type

Module function

Purpose

Interpolate a V-node variable to a VW-nodal point.

Reference

Section 10.2

Arguments

xin	Source array at the V-nodes
i	X-index of the source array
j	Y-index of the source array
k	Lower vertical index of the source array
intsrce	Selects valid points at the source node
	1: coastal boundaries, interior points and open boundaries only
	2: interior points only
	3: interior points and open boundaries only
	4: coastal boundaries and interior points only
intdest	Selects valid points at the destination node
	1: coastal boundaries, interior points and open boundaries only
	2: interior points only
	3: interior points and open boundaries only
	4: coastal boundaries and interior points only

outflag	Output flag for invalid points, if present. Zero otherwise.
vregular	Flag to select uniform or area averaging in the vertical, if present. Otherwise, the type of averaging is selected by iopt_arrint_vreg.
Vvar_at_VW	Interpolated value at the VW-node destination point

Vvar_at_W

```
FUNCTION Vvar_at_W(xin,i,j,k,intsrce,outflag,vregular)
LOGICAL, INTENT(IN), OPTIONAL :: vregular
INTEGER, INTENT(IN) :: i, intsrce, j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2,2) :: xin
REAL :: Vvar_at_W
```

File

array_interp.f90

Type

Module function

Purpose

Interpolate a V-node variable to a W-nodal point.

Reference

Section 10.2

Arguments

xin	Source array at the V-nodes
i	X-index of the source array
j	Lower Y-index of the source array
k	Lower vertical index of the source array
intsrce	Selects valid points at the source node 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
outflag	Output flag for invalid points, if present. Zero otherwise.

vregular — Flag to select uniform or area averaging in the vertical, if present. Otherwise, the type of averaging is selected by iopt_arrint_vreg.

Vvar_at_W — Interpolated value at the W-node destination point

VWvar_at_V

```
FUNCTION VWvar_at_V(xin,i,j,k,intsrce,intdest,outflag)
INTEGER, INTENT(IN) :: i, intdest, intsrce, j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2) :: xin
REAL :: VWvar_at_V
```

File

array_interp.f90

Type

Module function

Purpose

Interpolate a VW-node variable to a V-nodal point.

Reference

Section 10.2

Arguments

xin — Source array at the VW-nodes

i — X-index of the source array

j — Y-index of the source array

k — Lower vertical index of the source array

intsrce — Selects valid points at the source node

- 1: coastal boundaries, interior points and open boundaries only
- 2: interior points only
- 3: interior points and open boundaries only
- 4: coastal boundaries and interior points only

intdest — Selects valid points at the destination node

- 1: coastal boundaries, interior points and open boundaries only

2: interior points only
3: interior points and open boundaries only
4: coastal boundaries and interior points only

outflag Output flag for invalid points, if present. Zero otherwise.

VWvar_at_V Interpolated value at the V-node destination point

VWvar_at_W

```
FUNCTION VWvar_at_W(xin,i,j,k,intsrce,outflag)
INTEGER, INTENT(IN) :: i, intsrce, j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2) :: xin
REAL :: VWvar_at_W
```

File
array_interp.f90

Type
Module function

Purpose
Interpolate a VW-node variable to a W-nodal point.

Reference
Section 10.2

Arguments

xin Source array at the VW-nodes

i X-index of the source array

j Lower Y-index of the source array

k Vertical index of the source array

intsrce Selects valid points at the source node
1: coastal boundaries, interior points and open boundaries only
2: interior points only
3: interior points and open boundaries only
4: coastal boundaries and interior points only

outflag Output flag for invalid points, if present. Zero otherwise.

VWvar_at_W Interpolated value at the W-node destination point

Wvar_at_C

```
FUNCTION Wvar_at_C(xin,i,j,outflag)
INTEGER, INTENT(IN) :: i, j
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2) :: xin
REAL :: Wvar_at_C
```

File
array_interp.f90

Type
Module function

Purpose
Interpolate a W-node variable to a C-nodal point.

Reference
Section 10.2

Arguments

xin	Source array at the W-nodes
i	X-index of the source array
j	Y-index of the source array
outflag	Output flag for invalid points, if present. Zero otherwise.
Wvar_at_C	Interpolated value at the C-node destination point

Wvar_at_U

```
FUNCTION Wvar_at_U(xin,i,j,k,intdest,outflag,hregular)
LOGICAL, INTENT(IN), OPTIONAL :: hregular
INTEGER, INTENT(IN) :: i, intdest, j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2,2) :: xin
REAL :: Wvar_at_U
```

File
array_interp.f90

Type
Module function

Purpose
Interpolate a W-node variable to a U-nodal point.

Reference
Section 10.2

Arguments

xin	Source array at the W-nodes
i	Lower X-index of the source array
j	Y-index of the source array
k	Lower vertical index of the source array
intdest	Selects valid points at the destination node 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
outflag	Output flag for invalid points, if present. Zero otherwise.
hregular	Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.
Wvar_at_U	Interpolated value at the U-node destination point

Wvar_at_UW

```
FUNCTION Wvar_at_UW(xin,i,j,k,intdest,outflag,hregular)
LOGICAL, INTENT(IN), OPTIONAL :: hregular
INTEGER, INTENT(IN) :: i, intdest, j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2) :: xin
REAL :: Wvar_at_UW
```

File
array_interp.f90

Type
Module function

Purpose
Interpolate a W-node variable to a UW-nodal point.

Reference
Section 10.2

Arguments

xin	Source array at the W-nodes
i	Lower X-index of the source array
j	Y-index of the source array
k	Vertical index of the source array
intdest	Selects valid points at the destination node 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
outflag	Output flag for invalid points, if present. Zero otherwise.
hregular	Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.
Wvar_at_UW	Interpolated value at the UW-node destination point

Wvar_at_V

```
FUNCTION Wvar_at_V(xin,i,j,k,intdest,outflag,hregular)
LOGICAL, INTENT(IN), OPTIONAL :: hregular
INTEGER, INTENT(IN) :: i, intdest, j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2,2) :: xin
REAL :: Wvar_at_V
```

File
array_interp.f90

Type
Module function

Purpose
Interpolate a W-node variable to a V-nodal point.

Reference
Section 10.2

Arguments

xin	Source array at the W-nodes
i	X-index of the source array
j	Lower Y-index of the source array
k	Lower vertical index of the source array
intdest	Selects valid points at the destination node 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
outflag	Output flag for invalid points, if present. Zero otherwise.
hregular	Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.
Wvar_at_V	Interpolated value at the V-node destination point

Wvar_at_VW

```
FUNCTION Wvar_at_VW(xin,i,j,k,intdest,outflag,hregular)
LOGICAL, INTENT(IN), OPTIONAL :: hregular
INTEGER, INTENT(IN) :: i, intdest, j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2) :: xin
REAL :: Wvar_at_VW
```

File
array_interp.f90

Type
Module function

Purpose
Interpolate a W-node variable to a VW-nodal point.

Reference
Section 10.2

Arguments

xin	Source array at the W-nodes
i	X-index of the source array
j	Lower Y-index of the source array
k	Vertical index of the source array
intdest	Selects valid points at the destination node 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
outflag	Output flag for invalid points, if present. Zero otherwise.
hregular	Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.
Wvar_at_VW	Interpolated value at the VW-node destination point

31.2 NetCDF routine library

The purpose is to provide "aliases" for each routine of the netCDF library. In this way, a future upgrade to a newer version of netCDF can be implemented within this file without affecting the other parts of the COHERENS source code to a large extent. The implemented version of netCDF is compatible with versions no earlier than 3.6.

The alias of a netCDF routine, whose name starts with NF90_, has the prefix cf90_ in its name.

cf90_abort

```
SUBROUTINE cf90_abort(ncid)
INTEGER, INTENT(IN) :: ncid
```

File
cf90_routines.F90

Type
Module subroutine

Purpose
Aborts a netCDF routine call in case an error has been detected. This will be followed by an abortion of the program.

Arguments

ncid	netCDF file ID

netcdf call
NF90_abort

cf90_close

```
SUBROUTINE cf90_close(ncid)
INTEGER, INTENT(IN) :: ncid
```

File
cf90_routines.F90

Type
Module subroutine

Purpose
Closes an open netCDF data file.

Arguments

ncid	netCDF file ID

netcdf call
NF90_close

cf90_copy_att

```
SUBROUTINE cf90_copy_att(ncid_in,varid_in,name,ncid_out,varid_out)
CHARACTER (LEN=*), INTENT(IN) :: name
INTEGER, INTENT(IN) :: ncid_in, ncid_out, varid_in, varid_out
```

File
cf90_routines.F90

Type
Module subroutine

Purpose
Copies the attribute of variable in a netCDF file to the attribute of another variable attribute in another file.

Arguments

ncid_in	netCDF ID of the source file
varid_in	netCDF variable ID in the source file
name	Atrribute name
ncid_out	netCDF ID of destination file
varid_out	netCDF variable ID in the destination file

netcdf call
NF90_copy_att

cf90_create

```
SUBROUTINE cf90_create(path,cmode,ncid,initialsize,chunksize)
CHARACTER (LEN=*), INTENT(IN) :: path
INTEGER, INTENT(IN) :: cmode
INTEGER, INTENT(OUT) :: ncid
INTEGER, INTENT(IN), OPTIONAL :: initialsize
INTEGER, INTENT(INOUT), OPTIONAL :: chunksize
```

File
cf90_routines.F90

Type
Module subroutine

Purpose
Opens a new netCDF file.

Reference
NetCDF user manual (Pincus & Rew, 2008)

Arguments

path	Name of the new netCDF file
cmode	Creation mode

ncid	netCDF file ID
initialsize	Initial file size in bytes
chunksize	Chunksize

netcdf call
NF90_create

cf90_def_dim

```
SUBROUTINE cf90_def_dim(ncid,name,len,dimid)
CHARACTER (LEN=*), INTENT(IN) :: name
INTEGER, INTENT(IN) :: len, ncid
INTEGER, INTENT(OUT) :: dimid
```

File
cf90_routines.F90

Type
Module subroutine

Purpose
Adds a new dimension to an open netCDF file.

Arguments

ncid	netCDF file ID
name	Name of the new dimension
len	Value (length) of the new dimension
dimid	Returned netCDF ID of the new dimension.

netcdf call
NF90_def_dim

cf90_def_var

```
SUBROUTINE cf90_def_var(ncid,name,xtype,dimids,varid)
CHARACTER (LEN=*), INTENT(IN) :: name
INTEGER, INTENT(IN) :: ncid, xtype
INTEGER, INTENT(OUT) :: varid
INTEGER, INTENT(IN), DIMENSION(:) :: dimids
```

File
cf90_routines.F90

Type
Module subroutine

Purpose
Adds a new variable to an open netCDF file.

Arguments

ncid	netCDF file ID
name	Name of the new variable
xtype	Variables's netCDF data type
dimids	The netCDF IDs of the variable's dimensions.
varid	Returned netCDF variable ID

netcdf call
NF90_def_var

cf90_del_att

```
SUBROUTINE cf90_del_att(ncid,varid,name)
CHARACTER (LEN=*), INTENT(IN) :: name
INTEGER, INTENT(IN) :: ncid, varid
```

File
cf90_routines.F90

Type
Module subroutine

Purpose
Deletes a attribute from a netCDF file.

Arguments

ncid	netCDF file ID
varid	netCDF ID of the attribute's variable, NF_GLOBAL (global_NF90) for a global attribute
name	Name of the attribute

netcdf call
NF90_del_att

cf90_enddef

```
SUBROUTINE cf90_enddef(ncid,h_minfree,v_align,v_minfree,r_align)
INTEGER, INTENT(IN) :: ncid
INTEGER, INTENT(IN), OPTIONAL :: h_minfree, r_align, v_align, v_minfree
```

File
cf90_routines.F90

Type
Module subroutine

Purpose
Take an open netCDF dataset out of define mode.

Reference
NetCDF user manual (Pincus & Rew, 2008)

Arguments

ncid	netCDF file ID
h_minfree	Size in bytes of the pad at the end of the header
v_align	Alignment of the start of the data section for fixed size variables
v_minfree	Size in bytes of the pad at the end of the data section for fixed size variables
r_align	Alignment of the start of the data section in case of an unlimited dimension

netcdf call
NF90_enddef

cf90_error

```
SUBROUTINE cf90_error(cf90name)
CHARACTER (LEN=*), INTENT(IN) :: cf90name
```

File
cf90_routines.F90

Type
Module subroutine

Purpose
Report an error in a netCDF routine.

Arguments

cf90name Name of the netCDF routine where the error occurred.

netcdf call
NF90_strerror

cf90_get_att_chars

```
SUBROUTINE cf90_get_att_chars(ncid,varid,name,lenstr,chardat)
CHARACTER (LEN=*), INTENT(IN) :: name
INTEGER, INTENT(IN) :: ncid, lenstr, varid
CHARACTER (LEN=lenstr), INTENT(OUT) [,DIMENSION(:)] :: chardat
```

File
cf90_routines.F90

Type
Generic module subroutine

Purpose
Get the string value of a scalar or vector character-valued attribute.

Arguments

ncid	netCDF file ID
varid	netCDF ID of the attribute's variable, NF_GLOBAL (global_NF90) for a global attribute
name	Name of the attribute
lenstr	Length of the scalar or vector string data
chardat	Attribute values

Non-generic versions

cf90_get_att_chars_0d Scalar string values

cf90_get_att_chars_1d Vector string values

netcdf call
NF90_get_att

cf90_get_att

```
SUBROUTINE cf90_get_att(ncid,varid,name,values)
CHARACTER (LEN=*), INTENT(IN) :: name
INTEGER, INTENT(IN) :: ncid, varid
INTEGER or REAL, INTENT(OUT)[, DIMENSION[:] ::  values
```

File
cf90_routines.F90

Type
Generic module subroutine

Purpose
Read the numeric value of a scalar or vector attribute.

Arguments

ncid	netCDF file ID
varid	Id of the attribute's variable, NF_GLOBAL (global_NF90) for a global attribute
name	Name of the attribute
values	Attribute values

Non-generic versions

cf90_get_att_int_0d	Scalar integer values
cf90_get_att_int_1d	Vector integer values
cf90_get_att_real_0d	Scalar real values
cf90_get_att_real_1d	Vector real values

netcdf call
NF90_get_att

cf90_get_var_chars

```
SUBROUTINE cf90_get_att(ncid,varid,chardat,timerec)
INTEGER, INTENT(IN) :: ncid, timerec, varid
CHARACTER (LEN=*), INTENT(OUT) :: chardat
```

File
cf90_routines.F90

Type
Module subroutine

Purpose
Read a string variable from an open netCDF file.

Arguments

ncid	netCDF file ID
varid	netCDF ID of the attribute's variable, NF_GLOBAL (global_NF90) for a global attribute
chardat	Returned string value
timerec	Time record number

netcdf call
NF90_get_var

cf90_get_var

```
SUBROUTINE cf90_get_var(ncid,varid,values,timerec,start,&
                      & count,stride)
INTEGER, INTENT(IN) :: ncid, timerec, varid
INTEGER, INTENT(IN), OPTIONAL, DIMENSION([rank of values]) :: &
                      & count, start, stride
INTEGER or REAL, INTENT(OUT)[, DIMENSION[:[,:[,:[,:]]]]] ::  values
```

File
cf90_routines.F90

Type
Generic module subroutine

Purpose
Read an integer or real scalar or array (upto rank 4) variable from an open netCDF file.

Arguments

ncid	netCDF file ID
varid	Variable ID
values	Returned integer or real scalar of array data
timerec	Time record number

start	Vector of spatial start indices
count	Number of data along each array dimension
stride	Sampling interval along each spatial direction

The size of the optional start, count, stride arguments must be equal to rank of the returned data array.

Non-generic versions

cf90_get_var_int_0d	Scalar integer values. The optional arguments are not available.
cf90_get_var_int_1d	Vector integer values
cf90_get_var_int_2d	2-D array of integer values
cf90_get_var_int_3d	3-D array of integer values
cf90_get_var_int_4d	4-D array of integer values
cf90_get_var_real_0d	Scalar real values. The optional arguments are not available.
cf90_get_var_real_1d	Vector real values
cf90_get_var_real_2d	2-D array of real values
cf90_get_var_real_3d	3-D array of real values
cf90_get_var_real_4d	4-D array of real values

netcdf call

NF90_get_var

cf90_inq_attname

```
SUBROUTINE cf90_inq_attname(ncid,varid,attnum,name)
CHARACTER (LEN=*), INTENT(OUT) :: name
INTEGER, INTENT(IN) :: attnum, ncid, varid
```

File

cf90_routines.F90

Type

Module subroutine

Purpose

Returns the name of an attribute given its variable ID and number.

Arguments

ncid	netCDF file ID
varid	Id of the associated variable
attnum	Attribute number
name	Attribute name

netcdf call
NF90_inq_attname

cf90_inq_dimid

```
SUBROUTINE cf90_inq_dimid(ncid,name,dimid)
CHARACTER (LEN=*), INTENT(IN) :: name
INTEGER, INTENT(IN) :: ncid
INTEGER, INTENT(OUT) :: dimid
```

File
cf90_routines.F90

Type
Module subroutine

Purpose
Returns the dimension ID given the dimension's name.

Arguments

ncid	netCDF file ID
name	Dimension name
dimid	Returned netCDF dimension ID

netcdf call
NF90_inq_dimid

cf90_inq_libvers

```
FUNCTION cf90_inq_libvers()
CHARACTER (LEN=80) :: cf90_inq_libvers
```

File
cf90_routines.F90

Type
Module function

Purpose
Returns the netCDF library version.

netcdf call
NF90_inq_libvers

cf90_inq_varid

```
SUBROUTINE cf90_inq_varid(ncid,name,varid)
CHARACTER (LEN=*), INTENT(IN) :: name
INTEGER, INTENT(IN) :: ncid
INTEGER, INTENT(OUT) :: varid
```

File
cf90_routines.F90

Type
Module subroutine

Purpose
Returns the ID of a netCDF variable given its name.

Arguments

ncid	netCDF file ID
name	Variable name
varid	Returned netCDF variable ID

netcdf call
NF90_inq_varid

cf90_inquire

```
SUBROUTINE cf90_inquire(ncid,ndimensions,nvariables,&
                      & nattributes,unlimiteddimid)
INTEGER, INTENT(IN) :: ncid
INTEGER, INTENT(OUT), OPTIONAL :: nattributes, ndimensions,&
                                & nvariables, unlimiteddimid
```

File
cf90_routines.F90

Type
Module subroutine

Purpose
Returns information about an open netCDF file given its netCDF file ID.

Arguments

ncid	netCDF file ID
ndimensions	Returned number of dimensions
nvariables	Returned number of variables
nattributes	Returned number of global attributes
unlimiteddimid	Returned netCDF ID of the unlimited dimension (if any)

netcdf call
NF90_inquire

cf90_inquire_attribute

```
SUBROUTINE cf90_inquire_attribute(ncid,varid,name,xtype,len,attnum)
CHARACTER (LEN=*), INTENT(IN) :: name
INTEGER, INTENT(IN) :: ncid, varid
INTEGER, INTENT(OUT), OPTIONAL :: attnum, len, xtype
```

File
cf90_routines.F90

Type
Module subroutine

Purpose
Returns information about a netCDF attribute.

Arguments

ncid	netCDF file ID
varid	netCDF ID of the associated variable
name	Attribute name
xtype	Returned netCDF data type of the attribute
len	Returned number of values stored in the attribute
attnum	Returned attribute number

netcdf call
NF90_inquire_attribute

cf90_inquire_dimension

```
SUBROUTINE cf90_inquire_dimension(ncid,dimid,name,len)
INTEGER, INTENT(IN) :: dimid, ncid
CHARACTER (LEN=*), INTENT(OUT), OPTIONAL :: name
INTEGER, INTENT(OUT), OPTIONAL :: len
```

File
cf90_routines.F90

Type
Module subroutine

Purpose
Returns information about a netCDF dimension.

Arguments

ncid	netCDF file ID
dimid	netCDF dimension ID
name	Returned dimension name
len	Returned dimension length

netcdf call
NF90_inquire_dimension

cf90_inquire_variable

```
SUBROUTINE cf90_inquire_variable(ncid,varid,name,xtype,&
                                 & ndims,dimids,natts)
INTEGER, INTENT(IN) :: ncid, varid
CHARACTER (LEN=*), INTENT(OUT), OPTIONAL :: name
INTEGER, INTENT(OUT), OPTIONAL :: natts, ndims, xtype
INTEGER, INTENT(OUT), OPTIONAL, DIMENSION(:) :: dimids
```

File
cf90_routines.F90

Type
Module subroutine

Purpose
Returns information about a netCDF variable.

Arguments

ncid	netCDF file ID
varid	netCDF variable ID
name	Returned variable name
xtype	Returned netCDF data type
ndims	Returned variable rank
dimids	Returned netCDF IDs of the variable's dimensions
natts	Returned number of associated attributes

netcdf call

NF90_inquire_variable

cf90_open

```
SUBROUTINE cf90_open(path,mode,ncid,chunksize)
CHARACTER (LEN=*), INTENT(IN) :: path
INTEGER, INTENT(IN) :: mode
INTEGER, INTENT(OUT) :: ncid
INTEGER, INTENT(INOUT), OPTIONAL :: chunksize
```

File

cf90_routines.F90

Type

Module subroutine

Purpose

Opens an existing netCDF dataset.

Reference

NetCDF user manual (Pincus & Rew, 2008)

Arguments

path	Name of the netCDF file
mode	Access mode
ncid	netCDF file ID
chunksize	Chunksize

netcdf call

NF90_open

cf90_put_att_chars

```
SUBROUTINE cf90_put_att_chars(ncid,varid,name,lenstr,chardat)
CHARACTER (LEN=*), INTENT(IN) :: name
INTEGER, INTENT(IN) :: lenstr, ncid, varid
CHARACTER (LEN=lenstr), INTENT(IN) [,DIMENSION(:)] ::  chardat
```

File
: *cf90_routines.F90*

Type
: Generic module subroutine

Purpose
: Write or change the value of a scalar or vector character-valued attribute.

Arguments

ncid	netCDF file ID
varid	netCDF ID of the attribute's variable, NF_GLOBAL (global_NF90) for a global attribute
name	Name of the attribute
lenstr	Length of the scalar or vector string data
chardat	Attribute values

Non-generic versions

cf90_put_att_chars_0d Scalar string values

cf90_put_att_chars_1d Vector string values

netcdf call
: NF90_put_att

cf90_put_att

```
SUBROUTINE cf90_put_att(ncid,varid,name,values)
CHARACTER (LEN=*), INTENT(IN) :: name
INTEGER, INTENT(IN) :: ncid, varid
INTEGER or REAL, INTENT(OUT)[, DIMENSION[:]] ::  values
```

File
: *cf90_routines.F90*

Type
Generic module subroutine

Purpose
Write the numeric value of a scalar or vector attribute to an open netCDF file.

Arguments

ncid	netCDF file ID
varid	ID of the attribute's variable, NF_GLOBAL (global_NF90) for a global attribute
name	Name of the attribute
values	Attribute values

Non-generic versions

cf90_put_att_int_0d	Scalar integer values
cf90_put_att_int_1d	Vector integer values
cf90_put_att_real_0d	Scalar real values
cf90_put_att_real_1d	Vector real values

netcdf call
NF90_put_att

cf90_put_var_chars

```
SUBROUTINE cf90_put_var_chars(ncid,varid,chardat,timerec)
CHARACTER (LEN=*), INTENT(IN) :: chardat
INTEGER, INTENT(IN) :: ncid, timerec, varid
```

File
cf90_routines.F90

Type
Module subroutine

Purpose
Write a string variable to an open netCDF file.

Arguments

ncid	netCDF file ID

varid	Variable ID
chardat	Character output data
timerec	Time record number

netcdf call
NF90_put_var

cf90_put_var

```
SUBROUTINE cf90_put_var(ncid,varid,values,timerec,start,&
                      & count,stride)
INTEGER, INTENT(IN) :: ncid, timerec, varid
INTEGER, INTENT(IN), OPTIONAL, DIMENSION([rank of values]) :: &
                      & count, start, stride
INTEGER or REAL, INTENT(OUT)[, DIMENSION[:[,:[,:[,:]]]] ::  values
```

File
cf90_routines.F90

Type
Generic module subroutine

Purpose
Write an integer or real scalar or array (upto rank 4) variable to an open netCDF file.

Arguments

ncid	netCDF file ID
varid	Variable ID
values	Integer or real, scalar or array output data
timerec	Time record number
start	Vector of spatial start indices
count	Number of data along each array dimension
stride	Sampling interval along each spatial direction

The size of the optional start, count, stride arguments must be equal to rank of the output data array.

Non-generic versions

cf90_put_var_int_0d	Scalar integer values. The optional arguments are not available.

cf90_put_var_int_1d	Vector integer values
cf90_put_var_int_2d	2-D array of integer values
cf90_put_var_int_3d	3-D array of integer values
cf90_put_var_int_4d	4-D array of integer values
cf90_put_var_real_0d	Scalar real values. The optional arguments are not available.
cf90_put_var_real_1d	Vector real values
cf90_put_var_real_2d	2-D array of real values
cf90_put_var_real_3d	3-D array of real values
cf90_put_var_real_4d	4-D array of real values

netcdf call
NF90_put_var

cf90_redef

```
SUBROUTINE cf90_redef(ncid)
INTEGER, INTENT(IN) :: ncid
```

File
cf90_routines.F90

Type
Module subroutine

Purpose
Puts an open netCDF file in define mode.

Arguments

ncid	netCDF file ID

netcdf call
NF90_redef

cf90_rename_att

```
SUBROUTINE cf90_rename_att(ncid,varid,curname,newname)
CHARACTER (LEN=*), INTENT(IN) :: curname, newname
INTEGER, INTENT(IN) :: ncid, varid
```

File
cf90_routines.F90

Type
Module subroutine

Purpose
Renames an existing attribute.

Arguments

ncid	netCDF file ID
varid	netCDF ID of the associated variable or NF_GLOBAL (global_NF90) for a global attribute
curname	current name of the attribute
newname	new name of the attribute

netcdf call
NF90_rename_att

cf90_rename_dim

```
SUBROUTINE cf90_rename_dim(ncid,dimid,name)
CHARACTER (LEN=*), INTENT(OUT) :: name
INTEGER, INTENT(IN) :: dimid, ncid
```

File
cf90_routines.F90

Type
Module subroutine

Purpose
Renames an existing dimension.

Arguments

ncid	netCDF file ID
dimid	the dimension's netCDF ID
name	new dimension name

netcdf call
NF90_rename_dim

cf90_rename_var

```
SUBROUTINE cf90_rename_var(ncid,varid,newname)
CHARACTER (LEN=*), INTENT(OUT) :: newname
INTEGER, INTENT(IN) :: ncid, varid
```

File
cf90_routines.F90

Type
Module subroutine

Purpose
Changes the name of an existing variable.

Arguments

ncid	netCDF file ID
varid	the variable's netCDF ID
newname	new variable name

netcdf call
NF90_rename_var

cf90_set_fill

```
SUBROUTINE cf90_set_fill(ncid,fillmode,old_mode)
INTEGER, INTENT(IN) :: fillmode, ncid
INTEGER, INTENT(OUT) :: old_mode
```

File
cf90_routines.F90

Type
Module subroutine

Purpose
Resets the fill mode for a netCDF dataset open for writing.

Arguments

ncid	netCDF file ID
fillmode	New fill mode for the netCDF file
old_mode	Returned old fill mode of the netCDF file

netcdf call
NF90_set_fill

cf90_sync

```
SUBROUTINE cf90_sync(ncid)
INTEGER, INTENT(IN) :: ncid
```

File
cf90_routines.F90

Type
Module subroutine

Purpose
Resets the fill mode for a netCDF dataset open for writing.

Arguments

ncid	netCDF file ID

netcdf call
NF90_sync

31.3 Checking of model setup

Checks all kind of model setup for possible errors. One or more error messages are written followed by a program abort.

check_grid_arrays

```
SUBROUTINE check_grid_arrays
```

File
check_model.f90

Type
Module subroutine

Purpose
Check model grid arrays, as e.g. defined in usrdef_grid.

check_hrel_coords

```
SUBROUTINE check_hrel_coords(hcoords,nhdat,varname)
CHARACTER (LEN=*), INTENT(IN) :: varname
INTEGER, INTENT(IN) :: nhdat
TYPE (HRelativeCoords), INTENT(IN), DIMENSION(nhdat) :: hcoords
```

File
check_model.f90

Type
Module subroutine

Purpose
Check an derived type array with the horizontal relative coordinates of a number of data points with respect to the model grid.

Arguments

hcoords	Derived type array containing the relative coordinates
nhdat	Number of horizontal data locations.
varname	Name of the derived type array

check_initial_conditions

```
SUBROUTINE check_initial_conditions
```

File
check_model.f90

Type
Module subroutine

Purpose
Check initial condition arrays, as e.g. defined in usrdef_phsics.

check_mod_filepars

```
SUBROUTINE check_mod_filepars
```

File
check_model.f90

Type
: Module subroutine

Purpose
: Check the attributes of model forcing files, as e.g. defined in usrdef_mod_params.

check_mod_params

```
SUBROUTINE check_mod_params
```

File
: *check_model.f90*

Type
: Module subroutine

Purpose
: Check model setup parameters (switches, date and time, physical model parameters, ...), as e.g. defined in usrdef_mod_params.

check_out_filepars

```
SUBROUTINE check_out_filepars(filepars,arrname,maxvars)
CHARACTER (LEN=*), INTENT(IN) :: arrname
INTEGER, INTENT(IN), OPTIONAL :: maxvars
TYPE (FileParams), INTENT(INOUT), DIMENSION(:[,:])  ::  filepars
```

File
: *check_model.f90*

Type
: Generic module subroutine

Purpose
: Check the attributes of user-defined output files.

Arguments

filepars	Derived type array of file attributes
arrname	Name of the derived type array
maxvars	Maximum allowed number of variables in the file

Non-generic versions

check_out_filepars_1d	Vector array
check_out_filepars_2d	Two-dimensional array

check_out_gpars

```
SUBROUTINE check_out_gpars(outgpars,arrname,maxstats)
CHARACTER (LEN=*) :: arrname
INTEGER, INTENT(IN) :: maxstats
TYPE (OutGridParams), INTENT(IN), DIMENSION(:) :: outgpars
```

File
check_model.f90

Type
Module subroutine

Purpose
Check the attributes of user-defined output grids.

Arguments

outgpars	Derived type array of output grid attributes
arrname	Name of the derived type array
maxstats	Maximum number of output stations (as given by the user-defined parameter nostatstsr, nostatsavr or nostatsanal)

check_partition

```
SUBROUTINE check_partition
```

File
check_model.f90

Type
Module subroutine

Purpose
Check whether all wet and open boundary points are inside the domain of a local process.

check_statlocs

```
SUBROUTINE check_statlocs(outlocs,arrname)
CHARACTER (LEN=*), INTENT(IN) :: arrname
TYPE (StationLocs), INTENT(IN), DIMENSION(:) :: outlocs
```

File
check_model.f90

Type
Module subroutine

Purpose
Check whether the stations locations are inside the computational domain.

Arguments

outlocs	Derived type array of output station attributes
arrname	Name of the derived type array

check_struct_locs

```
SUBROUTINE check_struct_locs
```

File
check_model.f90

Type
Module subroutine

Purpose
Check whether thin dams, weirs and barrier locations are inside the computational domain and surrounded by wet cells.

check_surface_grid

```
SUBROUTINE check_surface_grid(surfgrid,n1dat,n2dat,varname)
CHARACTER (LEN=*), INTENT(IN) :: varname
INTEGER, INTENT(IN) :: n1dat, n2dat
TYPE (HReleativeCoords), INTENT(INOUT), DIMENSION(:,:)  ::  surfgrid
```

File
check_model.f90

Type
Generic module subroutine

Purpose
Check the relative coordinates of model grid points with respect to an external data grid.

Arguments

surfgrid	Derived type array containing the relative coordinates
n1dat	X-dimension of the external data grid
n2dat	Y-dimension of the external data grid
varname	Name of the derived type array

Non-generic versions

check_surface_grid_1d	Vector array
check_surface_grid_2d	Two-dimensional array

check_variables

```
SUBROUTINE check_variables(varatts,arrname)
CHARACTER (LEN=*), INTENT(IN) :: arrname
TYPE (VariableAtts), INTENT(IN), DIMENSION(:) :: varatts
```

File

check_model.f90

Type

Module subroutine

Purpose

Check a derived type vector array with variable attributes.

Arguments

varatts	Derived type array of variable attributes
arrname	Name of the derived type array

31.4 CIF utility routines

Ensemble of routines for reading data to or converting data from a central input file.

check_cif_lbound_vars

```
SUBROUTINE check_cif_lbound_novars(iddesc,novars,novarsmin)
INTEGER, INTENT(IN) :: iddesc, novars, novarsmin
```

File
cif_routines.f90

Type
Module subroutine

Purpose
Check whether the number of parameters on the input line of a CIF is not lower than a minimum limit.

Arguments

iddesc	Key id of the CIF
novars	Number of data values read from a line of the CIF
novarsmin	Minimum number of data values to be extracted from the input line.

conv_from_chars

```
SUBROUTINE conv_from_chars(string,cifdat,iddesc,lvar)
CHARACTER (LEN=*), INTENT(IN) :: string
CHARACTER (LEN=*), INTEGER, LOGICAL, or REAL, INTENT(OUT) ::  cifdat
INTEGER, INTENT(IN) :: iddesc
INTEGER, INTENT(INOUT) :: lvar
```

File
cif_routines.f90

Type
Generic module subroutine

Purpose
Convert a data value from the CIF into the appropriate numerical or non-numerical format.

Arguments

string	Data value in character format on input
cifdat	Returned data value in numerical (INTEGER or REAL) or non-numerical (LOGICAL or CHARACTER) format
iddesc	Key id of the CIF
lvar	Number of the variable on the CIF input line

Non-generic versions

conv_from_chars_chars	Convert to a character string
conv_from_chars_int	Convert to an integer value
conv_from_chars_log	Convert to a logical value
conv_from_chars_real	Convert to a real value

conv_from_chars_gridpars

```
SUBROUTINE conv_from_chars_gridpars(cvals,gridpars,iddesc,lvar)
CHARACTER (LEN=*), INTENT(IN), DIMENSION(20) :: cvals
INTEGER, INTENT(IN) :: iddesc
INTEGER, INTENT(INOUT) :: lvar
TYPE (OutGridParams), INTENT(INOUT) :: gridpars
```

File

cif_routines.f90

Type

Module subroutine

Purpose

Convert data values from the CIF to the components of a derived type variable of type OutGridParams, representing a user-defined output grid.

Arguments

cvals	CIF data in string format
gridpars	Returned components in the appropriate numerical or non-numerical format
iddesc	Key id of the CIF
lvar	Number of the variable on the CIF input line

conv_from_chars_infiles

```
SUBROUTINE conv_from_chars_infiles(cvals,filepars,iddesc,lvar)
CHARACTER (LEN=*), INTENT(IN), DIMENSION(10) :: cvals
INTEGER, INTENT(IN) :: iddesc
INTEGER, INTENT(INOUT) :: lvar
TYPE (FileParams), INTENT(INOUT) :: filepars
```

File

cif_routines.f90

Type
: Module subroutine

Purpose
: Convert data values from the CIF to the components of a derived type variable of type FileParams, representing a model forcing file.

Arguments

cvals	CIF data in string format
filepars	Returned components in the appropriate numerical or non-numerical format
iddesc	Key id of the CIF
lvar	Number of the variable on the CIF input line

conv_from_chars_outfiles

```
SUBROUTINE conv_from_chars_outfiles(cvals,filepars,iddesc,lvar)
CHARACTER (LEN=*), INTENT(IN), DIMENSION(6) :: cvals
INTEGER, INTENT(IN) :: iddesc
INTEGER, INTENT(INOUT) :: lvar
TYPE (FileParams), INTENT(INOUT) :: filepars
```

File
: *cif_routines.f90*

Type
: Module subroutine

Purpose
: Convert data values from the CIF to the components of a derived type variable of type FileParams, representing a user output file.

Arguments

cvals	CIF data in string format
filepars	Returned components in the appropriate numerical or non-numerical format
iddesc	Key id of the CIF
lvar	Number of the variable on the CIF input line

conv_from_chars_statlocs

```
SUBROUTINE conv_from_chars_statlocs(cvals,statlocs,iddesc,lvar)
CHARACTER (LEN=*), INTENT(IN), DIMENSION(3) :: cvals
INTEGER, INTENT(IN) :: iddesc
INTEGER, INTENT(INOUT) :: lvar
TYPE (StationLocs), INTENT(INOUT) :: statlocs
```

File
: *cif_routines.f90*

Type
: Module subroutine

Purpose
: Convert data values from the CIF to the components of a derived type variable of type StationLocs, representing output data stations.

Arguments

cvals	CIF data in string format
statlocs	Returned components in the appropriate numerical or non-numerical format
iddesc	Key id of the CIF
lvar	Number of the variable on the CIF input line

conv_from_chars_surfgrd

```
SUBROUTINE conv_from_chars_statlocs(cvals,surfgrd,iddesc,lvar)
CHARACTER (LEN=*), INTENT(IN), DIMENSION(7) :: cvals
INTEGER, INTENT(IN) :: iddesc
INTEGER, INTENT(INOUT) :: lvar
TYPE (GridParams), INTENT(INOUT) :: surfgrd
```

File
: *cif_routines.f90*

Type
: Module subroutine

Purpose
: Convert data values from the CIF to the components of a derived type variable of type GridParams, representing a surface grid.

Arguments

cvals	CIF data in string format
surfgrd	Returned components in the appropriate numerical or non-numerical format
iddesc	Key id of the CIF
lvar	Number of the variable on the CIF input line

conv_from_chars_varatts

```
SUBROUTINE conv_from_chars_varatts(cvals,varatts,iddesc,lvar)
CHARACTER (LEN=*), INTENT(IN), DIMENSION(10) :: cvals
INTEGER, INTENT(IN) :: iddesc
INTEGER, INTENT(INOUT) :: lvar
TYPE (VariableAtts), INTENT(INOUT) :: varatts
```

File

cif_routines.f90

Type

Module subroutine

Purpose

Convert data values from the CIF to the components of a derived type variable of type VariableAtts.

Arguments

cvals	CIF data in string format
varatts	Returned components in the appropriate numerical or non-numerical format
iddesc	Key id of the CIF
lvar	Number of the variable on the CIF input line

conv_to_chars

```
SUBROUTINE conv_to_chars(string,cifdat)
CHARACTER (LEN=*), INTENT(OUT) :: string
INTEGER, LOGICAL or REAL, [DIMENSION(:),] INTENT(IN) ::  cifdat
```

File

cif_routines.f90

Type
Generic module subroutine

Purpose
Convert a INTEGER, LOGICAL or REAL scalar or vector model variable to a character string.

Arguments

string	Returned data string
cifdat	Scalar or vector data on input

Non-generic versions

conv_to_chars_int_0d	Integer scalar
conv_to_chars_int_1d	Integer vector
conv_to_chars_log_0d	Logical scalar
conv_to_chars_log_1d	Logical vector
conv_to_chars_real_0d	Real scalar
conv_to_chars_real_1d	Real vector

conv_to_chars_gridpars

```
SUBROUTINE conv_to_chars_gridpars(cvals,gridpars)
CHARACTER (LEN=*), INTENT(OUT), DIMENSION(20) :: cvals
TYPE (OutGridParams), INTENT(IN) :: gridpars
```

File
cif_routines.f90

Type
Module subroutine

Purpose
Convert the attributes of a user-defined output grid to a vector of string data.

Arguments

cvals	Returned data in string format
gridpars	Attributes of the output grid

conv_to_chars_infiles

```
SUBROUTINE conv_to_chars_infiles(cvals,filepars)
CHARACTER (LEN=*), INTENT(OUT), DIMENSION(10) :: cvals
TYPE (FileParams), INTENT(IN) :: filepars
```

File
cif_routines.f90

Type
Module subroutine

Purpose
Convert the attributes of a model forcing file to a vector of string data.

Arguments

cvals	Returned data in string format
filepars	Attributes of the forcing file

conv_to_chars_outfiles

```
SUBROUTINE conv_to_chars_outfiles(cvals,filepars)
CHARACTER (LEN=*), INTENT(OUT), DIMENSION(6) :: cvals
TYPE (FileParams), INTENT(IN) :: filepars
```

File
cif_routines.f90

Type
Module subroutine

Purpose
Convert the attributes of a usder-defined output file to a vector of string data.

Arguments

cvals	Returned data in string format
filepars	Attributes of the output file

conv_to_chars_statlocs

```
SUBROUTINE conv_to_chars_statlocs(cvals,statlocs)
CHARACTER (LEN=*), INTENT(OUT), DIMENSION(3) :: cvals
TYPE (StationLocs), INTENT(IN) :: statlocs
```

File
cif_routines.f90

Type
Module subroutine

Purpose
Convert the attributes of a user-defined output station to a vector of string data.

Arguments

cvals	Returned data in string format
statlocs	Attributes of the output station

conv_to_chars_surfgrd

```
SUBROUTINE conv_to_chars_surfgrd(cvals,surfgrd)
CHARACTER (LEN=*), INTENT(OUT), DIMENSION(7) :: cvals
TYPE (GridParams), INTENT(IN) :: surfgrd
```

File
cif_routines.f90

Type
Module subroutine

Purpose
Convert the attributes of a surface grid to a vector of string data.

Arguments

cvals	Returned data in string format
surfgrd	Attributes of the surface grid

conv_to_chars_varatts

```
SUBROUTINE conv_to_chars_varatts(cvals,varatts)
CHARACTER (LEN=*), INTENT(OUT), DIMENSION(10) :: cvals
TYPE (VariableAtts), INTENT(IN) :: varatts
```

File
cif_routines.f90

Type
Module subroutine

Purpose
Convert the attributes of a model variable to a vector of string data.

Arguments

cvals	Returned data in string format
varatts	Attributes of the variable

error_cif_var

```
SUBROUTINE error_cif_var(iddesc,lvar)
INTEGER, INTENT(IN) :: iddesc, lvar
```

File
cif_routines.f90

Type
Module subroutine

Purpose
Write an error message if a variable in a CIF cannot be read

Arguments

iddesc	Key id of the CIF
lvar	Number of the variable on the CIF input line

read_cif_line

```
SUBROUTINE read_cif_line(iddesc,cline,cvals,numvars,cname)
CHARACTER (LEN=lencifline), INTENT(INOUT) :: cline
CHARACTER (LEN=lenname), INTENT(OUT), OPTIONAL :: cname
CHARACTER (LEN=lencifvar), INTENT(OUT), DIMENSION(:) :: cvals
INTEGER, INTENT(IN) :: iddesc
INTEGER, INTENT(OUT) :: numvars
```

File
cif_routines.f90

Type
Module subroutine

Purpose
Parse a series of variables, separated by a delimiter as character data on a data input line

Arguments

iddesc	Key id of the CIF
cline	Input line string
cvals	Returned data value(s) in string format
numvars	Returned number of data variable(s)
cname	Returned name of the data variable(s) (if requested)

write_cif_line

```
SUBROUTINE write_cif_line(iddesc,cvals,cname)
CHARACTER (LEN=*), INTENT(IN) :: cname
CHARACTER (LEN=*), INTENT(INOUT), DIMENSION(:) :: cvals
INTEGER, INTENT(IN) :: iddesc
```

File
cif_routines.f90

Type
Module subroutine

Purpose
Write one or more string data to a CIF

Arguments

iddesc	Key id of the CIF
cvals	Data value(s) in string format
cname	Name of the data variable(s)

31.5 MPI routine library

The purpose is to provide "aliases" for the routines of the MPI library, used in the present version of the program. In this way, a future upgrade to a newer version of MPI can be implemented within this file without affecting the other parts of the COHERENS source code to much. The current implemented version of MPI is Version 1.1. For a detailed description of the MPI library routines see (MPI, 1995).

comms_allgather_int

```
SUBROUTINE comms_allgather_int(sendbuf,count,nprocs,recvbuf,&
                               & comm,ivarid)
INTEGER, INTENT(IN) :: comm, count, ivarid, nprocs
INTEGER, INTENT(IN), DIMENSION(count) :: sendbuf
INTEGER, INTENT(OUT), DIMENSION(count,nprocs) :: recvbuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Gather (collect) integer data on all processes in communicator comm.

Arguments

sendbuf	Starting address of send buffer
count	Number of data in send and receive buffers
nprocs	Number of processes
recvbuf	Starting address of receive buffer
comm	Communicator id
ivarid	Variable key id (used for log info only, zero for undefined)

MPI call
MPI_allgather

comms_allgather_log

```
SUBROUTINE comms_allgather_log(sendbuf,count,nprocs,recvbuf,&
                               & comm,ivarid)
INTEGER, INTENT(IN) :: comm, count, ivarid, nprocs
LOGICAL, INTENT(IN), DIMENSION(count) :: sendbuf
LOGICAL, INTENT(OUT), DIMENSION(count,nprocs) :: recvbuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Gather (collect) logical data on all processes in communicator comm.

Arguments

sendbuf	Starting address of send buffer
count	Number of data in send and receive buffers
nprocs	Number of processes
recvbuf	Starting address of receive buffer
comm	Communicator id
ivarid	Variable key id (used for log info only, zero for undefined)

MPI call
MPI_allgather

comms_allgather_real

```
SUBROUTINE comms_allgather_log(sendbuf,count,nprocs,recvbuf,&
                               & comm,ivarid)
INTEGER, INTENT(IN) :: comm, count, ivarid, nprocs
REAL, INTENT(IN), DIMENSION(count) :: sendbuf
REAL, INTENT(OUT), DIMENSION(count,nprocs) :: recvbuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Gather (collect) real data on all processes in communicator comm.

Arguments

sendbuf	Starting address of send buffer
count	Number of data in send and receive buffers
nprocs	Number of processes
recvbuf	Starting address of receive buffer
comm	Communicator id
ivarid	Variable key id (used for log info only, zero for undefined)

MPI call
MPI_allgather

comms_barrier

```
SUBROUTINE comms_barrier(comm)
INTEGER, INTENT(IN) :: comm
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Synchronise processes in communicator comm.

Arguments

comm	Communicator id

MPI call
MPI_barrier

comms_bcast_char

```
SUBROUTINE comms_bcast_char(cbuf,count,root,comm,ivarid)
INTEGER, INTENT(IN) :: comm, count, ivarid, root
CHARACTER, INTENT(INOUT), DIMENSION(count) :: cbuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Broadcast (copy) character data from the root process to all other processes in communicator comm.

Arguments

cbuf	Starting address of buffer
count	Number of data in send and receive buffers
root	Process id of root process
comm	Communicator id
ivarid	Variable key id (used for log info only, zero for undefined)

MPI call
MPI_bcast

comms_bcast_int

```
SUBROUTINE comms_bcast_int(ibuf,count,root,comm,ivarid)
INTEGER, INTENT(IN) :: comm, count, ivarid, root
INTEGER, INTENT(INOUT), DIMENSION(count) :: ibuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Broadcast (copy) integer data from the root process to all other processes in communicator comm.

Arguments

ibuf	Starting address of buffer
count	Number of data in send and receive buffers
root	Process id of root process
comm	Communicator id
ivarid	Variable key id (used for log info only, zero for undefined)

MPI call
MPI_bcast

comms_bcast_log

```
SUBROUTINE comms_bcast_log(lbuf,count,root,comm,ivarid)
INTEGER, INTENT(IN) :: comm, count, ivarid, root
LOGICAL, INTENT(INOUT), DIMENSION(count) :: lbuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Broadcast (copy) logical data from the root process to all other processes in communicator comm.

Arguments

lbuf	Starting address of buffer
count	Number of data in send and receive buffers
root	Process id of root process
comm	Communicator id
ivarid	Variable key id (used for log info only, zero for undefined)

MPI call
MPI_bcast

comms_bcast_real

```
SUBROUTINE comms_bcast_real(rbuf,count,root,comm,ivarid)
INTEGER, INTENT(IN) :: comm, count, ivarid, root
REAL, INTENT(INOUT), DIMENSION(count) :: rbuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Broadcast (copy) real data from the root process to all other processes in communicator comm.

Arguments

rbuf	Starting address of buffer
count	Number of data in send and receive buffers
root	Process id of root process
comm	Communicator id
ivarid	Variable key id (used for log info only, zero for undefined)

MPI call
MPI_bcast

comms_comm_free

```
SUBROUTINE comms_comm_free(comm)
INTEGER, INTENT(INOUT) :: comm
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Deallocates (frees) communicator comm.

Arguments

comm	Communicator id

MPI call
MPI_comm_free

comms_comm_rank

```
SUBROUTINE comms_comm_rank(comm,rank)
INTEGER, INTENT(IN) :: comm
INTEGER, INTENT(OUT) :: rank
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Returns the rank (process id) of the calling process.

Arguments

comm	Communicator id
rank	Returned rank

MPI call
MPI_comm_rank

comms_comm_size

```
SUBROUTINE comms_comm_size(comm,size)
INTEGER, INTENT(IN) :: comm
INTEGER, INTENT(OUT) :: size
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Returns the number of processes in communicator comm.

Arguments

comm	Communicator id
rank	Returned number of processes

MPI call
MPI_comm_size

comms_comm_split

```
SUBROUTINE comms_comm_split(comm,color,key,newcomm)
INTEGER, INTENT(IN) :: color, comm, key
INTEGER, INTENT(OUT) :: newcomm
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Create a new communicator newcomm by partinioning of communicator comm.

Arguments

comm	Id of existing communicator
color	Control parameter for subset assignment
key	Control parameter for rank assignment
newcomm	Id of new communicator

MPI call
MPI_comm_split

comms_dims_create

```
SUBROUTINE comms_dimms_create(nnodes,ndims,dims)
INTEGER, INTENT(IN) :: ndims, nnodes
INTEGER, INTENT(INOUT), DIMENSION(ndims) :: dims
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Create a Cartesian (parallel) grid of dimension ndims and a total of nnodes nodes and store the domain dimensions in dims.

Arguments

ndims	Dimension of the Cartesian grid
nnodes	Number of nodes for creating the grid
dims	Vector with the number of nodes (processes) per grid direction

MPI call
MPI_dims_create

comms_errhandler

```
SUBROUTINE comms_errhandler(comm)
INTEGER, INTENT(IN) :: comm
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Sets the error handler to MPI_errors_return.

Arguments

comm	Communicator id

MPI calls
MPI_errhandler_free, MPI_errhandler_get, MPI_errhandler_set

comms_finalize

```
SUBROUTINE comms_finalize
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Finalise MPI.

MPI call
MPI_finalize

comms_initialise

```
SUBROUTINE comms_initialise
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Initialise MPI.

MPI calls
MPI_init, MPI_initialized

comms_irecv_char

```
SUBROUTINE comms_irecv_char(cbuf,count,source,tag,comm,request,ivarid)
INTEGER, INTENT(IN) :: comm, count, ivarid, source, tag
INTEGER, INTENT(OUT) :: request
CHARACTER, INTENT(OUT), DIMENSION(count) :: cbuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Performs a non-blocking receive of character data from process source.

Arguments

cbuf	Starting address of receive buffer
count	Number of data in receive buffer
source	Rank of source process
tag	Message tag
comm	Communicator id
request	Communication request
ivarid	Variable key id (used for log info only, zero for undefined)

MPI call
MPI_irecv

comms_irecv_int

```
SUBROUTINE comms_irecv_int(ibuf,count,source,tag,comm,request,ivarid)
INTEGER, INTENT(IN) :: comm, count, ivarid, source, tag
INTEGER, INTENT(OUT) :: request
INTEGER, INTENT(OUT), DIMENSION(count) :: ibuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Performs a non-blocking receive of integer data from process source.

Arguments

ibuf	Starting address of receive buffer
count	Number of data in receive buffer
source	Rank of source process
tag	Message tag
comm	Communicator id
request	Communication request
ivarid	Variable key id (used for log info only, zero for undefined)

MPI call
MPI_irecv

comms_irecv_log

```
SUBROUTINE comms_irecv_int(lbuf,count,source,tag,comm,request,ivarid)
INTEGER, INTENT(IN) :: comm, count, ivarid, source, tag
INTEGER, INTENT(OUT) :: request
LOGICAL, INTENT(OUT), DIMENSION(count) :: lbuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Performs a non-blocking receive of logical data from process source.

Arguments

lbuf	Starting address of receive buffer
count	Number of data in receive buffer

source	Rank of source process
tag	Message tag
comm	Communicator id
request	Communication request
ivarid	Variable key id (used for log info only, zero for undefined)

MPI call
MPI_irecv

comms_irecv_real

```
SUBROUTINE comms_irecv_real(rbuf,count,source,tag,comm,request,ivarid)
INTEGER, INTENT(IN) :: comm, count, ivarid, source, tag
INTEGER, INTENT(OUT) :: request
REAL, INTENT(OUT), DIMENSION(count) :: rbuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Performs a non-blocking receive of real data from process source.

Arguments

rbuf	Starting address of receive buffer
count	Number of data in receive buffer
source	Rank of source process
tag	Message tag
comm	Communicator id
request	Communication request
ivarid	Variable key id (used for log info only, zero for undefined)

MPI call
MPI_irecv

comms_isend_char

```
SUBROUTINE comms_isend_char(cbuf,count,dest,tag,comm,mode,&
                          & request,ivarid)
INTEGER, INTENT(IN) :: comm, count, dest, ivarid, mode, tag
INTEGER, INTENT(OUT) :: request
CHARACTER, INTENT(IN), DIMENSION(count) :: cbuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Performs a non-blocking send of character data to process dest.

Arguments

cbuf	Starting address of send buffer
count	Number of data in send buffer
dest	Rank of destination process
tag	Message tag
comm	Communicator id
mode	Selects type of send
	1: standard send
	2: synchronous send
request	Communication request
ivarid	Variable key id (used for log info only, zero for undefined)

MPI calls
MPI_isend, MPI_issend

comms_isend_int

```
SUBROUTINE comms_isend_int(ibuf,count,dest,tag,comm,mode,&
                         & request,ivarid)
INTEGER, INTENT(IN) :: comm, count, dest, ivarid, mode, tag
INTEGER, INTENT(OUT) :: request
INTEGER, INTENT(IN), DIMENSION(count) :: ibuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Performs a non-blocking send of integer data to process dest.

Arguments

ibuf	Starting address of send buffer
count	Number of data in send buffer
dest	Rank of destination process
tag	Message tag
comm	Communicator id
mode	Selects type of send
	1: standard send
	2: synchronous send
request	Communication request
ivarid	Variable key id (used for log info only, zero for undefined)

MPI calls
MPI_isend, MPI_issend

comms_isend_log

```
SUBROUTINE comms_isend_log(lbuf,count,dest,tag,comm,mode,&
                         & request,ivarid)
INTEGER, INTENT(IN) :: comm, count, dest, ivarid, mode, tag
INTEGER, INTENT(OUT) :: request
LOGICAL, INTENT(IN), DIMENSION(count) :: lbuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Performs a non-blocking send of logical data to process dest.

Arguments

lbuf	Starting address of send buffer
count	Number of data in send buffer
dest	Rank of destination process
tag	Message tag
comm	Communicator id
mode	Selects type of send
	1: standard send
	2: synchronous send
request	Communication request
ivarid	Variable key id (used for log info only, zero for undefined)

MPI calls

MPI_isend, MPI_issend

comms_isend_real

```
SUBROUTINE comms_isend_real(rbuf,count,dest,tag,comm,mode,&
                          & request,ivarid)
INTEGER, INTENT(IN) :: comm, count, dest, ivarid, mode, tag
INTEGER, INTENT(OUT) :: request
REAL, INTENT(IN), DIMENSION(count) :: rbuf
```

File

comms_MPI.F90

Type

Module subroutine

Purpose

Performs a non-blocking send of real data to process dest.

Arguments

rbuf	Starting address of send buffer
count	Number of data in send buffer
dest	Rank of destination process
tag	Message tag
comm	Communicator id

mode	Selects type of send
	1: standard send
	2: synchronous send
request	Communication request
ivarid	Variable key id (used for log info only, zero for undefined)

MPI calls
MPI_isend, MPI_issend

comms_recv_char

```
SUBROUTINE comms_recv_char(cbuf,count,source,tag,comm,ivarid)
INTEGER, INTENT(IN) :: comm, count, ivarid, source, tag
CHARACTER, INTENT(OUT), DIMENSION(count) :: cbuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Performs a blocking receive of character data from process source.

Arguments

cbuf	Starting address of receive buffer
count	Number of data in receive buffer
source	Rank of source process
tag	Message tag
comm	Communicator id
ivarid	Variable key id (used for log info only, zero for undefined)

MPI call
MPI_recv

comms_recv_int

```
SUBROUTINE comms_recv_int(ibuf,count,source,tag,comm,ivarid)
INTEGER, INTENT(IN) :: comm, count, ivarid, source, tag
INTEGER, INTENT(OUT), DIMENSION(count) :: ibuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Performs a blocking receive of integer data from process source.

Arguments

ibuf	Starting address of receive buffer
count	Number of data in receive buffer
source	Rank of source process
tag	Message tag
comm	Communicator id
ivarid	Variable key id (used for log info only, zero for undefined)

MPI call
MPI_recv

comms_recv_log

```
SUBROUTINE comms_recv_log(lbuf,count,source,tag,comm,ivarid)
INTEGER, INTENT(IN) :: comm, count, ivarid, source, tag
LOGICAL, INTENT(OUT), DIMENSION(count) :: lbuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Performs a blocking receive of logical data from process source.

Arguments

lbuf	Starting address of receive buffer
count	Number of data in receive buffer
source	Rank of source process
tag	Message tag

comm	Communicator id
ivarid	Variable key id (used for log info only, zero for undefined)

MPI call
MPI_recv

comms_recv_real

```
SUBROUTINE comms_recv_real(rbuf,count,source,tag,comm,ivarid)
INTEGER, INTENT(IN) :: comm, count, ivarid, source, tag
REAL, INTENT(OUT), DIMENSION(count) :: rbuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Performs a blocking receive of real data from process source.

Arguments

rbuf	Starting address of receive buffer
count	Number of data in receive buffer
source	Rank of source process
tag	Message tag
comm	Communicator id
ivarid	Variable key id (used for log info only, zero for undefined)

MPI call
MPI_recv

comms_reduce_int

```
SUBROUTINE comms_reduce_int(sendbuf,recvbuf,iop,comm,root,ivarid)
INTEGER, INTENT(IN) :: comm, iop, ivarid
INTEGER, INTENT(IN), OPTIONAL :: root
INTEGER, INTENT(IN) :: sendbuf
INTEGER, INTENT(OUT) :: recvbuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Performs a global reduction operation on integer data.

Arguments

sendbuf	Address of send buffer
recvbuf	Address of receive buffer
iop	Type of reduce operation
	1 : maximum (MPI_max)
	2 : minimum (MPI_min)
	3 : sum (MPI_sum)
	4 : product (MPI_prod)
	6 : logical and (MPI_band)
	8 : bit-wise or (MPI_bor)
	10: bot-wise xor (MPI_bxor)
comm	Communicator id
root	Rank of root process to which the result is returned, if present. Otherwise the result is returned to all processes.
ivarid	Variable key id (used for log info only, zero for undefined)

MPI calls
MPI_reduce, MPI_allreduce

comms_reduce_2int

```
SUBROUTINE comms_reduce_2int(sendbuf,recvbuf,iop,comm,root,ivarid)
INTEGER, INTENT(IN) :: comm, iop, ivarid
INTEGER, INTENT(IN), OPTIONAL :: root
INTEGER, INTENT(IN), DIMENSION(2) :: sendbuf
INTEGER, INTENT(OUT), DIMENSION(2) :: recvbuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Performs a global reduction operation on pairs of integer data.

Arguments

sendbuf	Address of send buffer
recvbuf	Address of receive buffer
iop	Type of reduce operation
	11: maximum value and location (MPI_maxloc)
	12: minimum value and location (MPI_minloc)
comm	Communicator id
root	Rank of root process to which the result is returned, if present. Otherwise the result is returned to all processes.
ivarid	Variable key id (used for log info only, zero for undefined)

MPI calls
MPI_reduce, MPI_allreduce

comms_reduce_log

```
SUBROUTINE comms_reduce_log(sendbuf,recvbuf,iop,comm,root,ivarid)
LOGICAL, INTENT(IN) :: sendbuf
LOGICAL, INTENT(OUT) :: recvbuf
INTEGER, INTENT(IN) :: comm, iop, ivarid
INTEGER, INTENT(IN), OPTIONAL :: root
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Performs a global reduction operation on logical data.

Arguments

sendbuf	Address of send buffer
recvbuf	Address of receive buffer

iop	Type of reduce operation
	5: logical and (MPI_land)
	7: logical or (MPI_lor)
	9: logical xor (MPI_lxor)
comm	Communicator id
root	Rank of root process to which the result is returned, if present. Otherwise the result is returned to all processes.
ivarid	Variable key id (used for log info only, zero for undefined)

MPI calls
MPI_reduce, MPI_allreduce

comms_reduce_real

```
SUBROUTINE comms_reduce_real(sendbuf,recvbuf,iop,comm,root,ivarid)
INTEGER, INTENT(IN) :: comm, iop, ivarid
INTEGER, INTENT(IN), OPTIONAL :: root
REAL, INTENT(IN) :: sendbuf
REAL, INTENT(OUT) :: recvbuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Performs a global reduction operation on real data.

Arguments

sendbuf	Address of send buffer
recvbuf	Address of receive buffer
iop	Type of reduce operation
	1: maximum (MPI_max)
	2: minimum (MPI_min)
	3: sum (MPI_sum)
	4: product (MPI_prod)
comm	Communicator id

root	Rank of root process to which the result is returned, if present. Otherwise the result is returned to all processes.
ivarid	Variable key id (used for log info only, zero for undefined)

MPI calls

MPI_reduce, MPI_allreduce

comms_reduce_2real

```
SUBROUTINE comms_reduce_2real(sendbuf,recvbuf,iop,comm,root,ivarid)
INTEGER, INTENT(IN) :: comm, iop, ivarid
INTEGER, INTENT(IN), OPTIONAL :: root
REAL, INTENT(IN), DIMENSION(2) :: sendbuf
REAL, INTENT(OUT), DIMENSION(2) :: recvbuf
```

File

comms_MPI.F90

Type

Module subroutine

Purpose

Performs a global reduction operation on pairs of real data.

Arguments

sendbuf	Address of send buffer
recvbuf	Address of receive buffer
iop	Type of reduce operation
	11: maximum value and location (MPI_maxloc)
	12: minimum value and location (MPI_minloc)
comm	Communicator id
root	Rank of root process to which the result is returned, if present. Otherwise the result is returned to all processes.
ivarid	Variable key id (used for log info only, zero for undefined)

MPI calls

MPI_allreduce, MPI_reduce

comms_send_char

```
SUBROUTINE comms_send_char(cbuf,count,dest,tag,comm,mode,ivarid)
INTEGER, INTENT(IN) :: comm, count, dest, ivarid, mode, tag
CHARACTER, INTENT(IN), DIMENSION(count) :: cbuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Performs a blocking send of character data to process dest.

Arguments

cbuf	Starting address of send buffer
count	Number of data in send buffer
dest	Rank of destination process
tag	Message tag
comm	Communicator id
mode	Selects type of send
	1: standard send
	2: synchronous send
ivarid	Variable key id (used for log info only, zero for undefined)

MPI calls
MPI_send, MPI_ssend

comms_send_int

```
SUBROUTINE comms_send_int(ibuf,count,dest,tag,comm,mode,ivarid)
INTEGER, INTENT(IN) :: comm, count, dest, ivarid, mode, tag
INTEGER, INTENT(IN), DIMENSION(count) :: ibuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Performs a blocking send of integer data to process dest.

Arguments

ibuf	Starting address of send buffer
count	Number of data in send buffer
dest	Rank of destination process
tag	Message tag
comm	Communicator id
mode	Selects type of send
	1: standard send
	2: synchronous send
ivarid	Variable key id (used for log info only, zero for undefined)

MPI calls
MPI_send, MPI_ssend

comms_send_log

```
SUBROUTINE comms_send_log(lbuf,count,dest,tag,comm,mode,ivarid)
INTEGER, INTENT(IN) :: comm, count, dest, ivarid, mode, tag
LOGICAL, INTENT(IN), DIMENSION(count) :: lbuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Performs a blocking send of logical data to process dest.

Arguments

lbuf	Starting address of send buffer
count	Number of data in send buffer
dest	Rank of destination process
tag	Message tag
comm	Communicator id

mode	Selects type of send
	1: standard send
	2: synchronous send
ivarid	Variable key id (used for log info only, zero for undefined)

MPI calls
MPI_send, MPI_ssend

comms_send_real

```
SUBROUTINE comms_send_real(rbuf,count,dest,tag,comm,mode,ivarid)
INTEGER, INTENT(IN) :: comm, count, dest, ivarid, mode, tag
REAL, INTENT(IN), DIMENSION(count) :: rbuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Performs a blocking send of real data to process dest.

Arguments

rbuf	Starting address of send buffer
count	Number of data in send buffer
dest	Rank of destination process
tag	Message tag
comm	Communicator id
mode	Selects type of send
	1: standard send
	2: synchronous send
ivarid	Variable key id (used for log info only, zero for undefined)

MPI calls
MPI_send, MPI_ssend

comms_sendrecv_char

```
SUBROUTINE comms_sendrecv_char(sendbuf,sendcount,dest,sendtag,&
                            & recvbuf,recvcount,source,recvtag,&
                            & comm,ivarid)
INTEGER, INTENT(IN) :: comm, dest, ivarid, recvcount, recvtag,&
                     & sendcount, sendtag, source
CHARACTER, INTENT(IN), DIMENSION(sendcount) :: sendbuf
CHARACTER, INTENT(OUT), DIMENSION(recvcount) :: recvbuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Combines a send and a receive operation on character data.

Arguments

sendbuf	Starting address of send buffer
sendcount	Number of data in send buffer
dest	Rank of destination process
sendtag	Message tag of send operation
recvbuf	Starting address of receive buffer
recvcount	Number of data in receive buffer
source	Rank of source process
recvtag	Message tag of receive operation
comm	Communicator id
ivarid	Variable key id (used for log info only, zero for undefined)

MPI call
MPI_sendrecv

comms_sendrecv_int

```
SUBROUTINE comms_sendrecv_int(sendbuf,sendcount,dest,sendtag,&
                            & recvbuf,recvcount,source,recvtag,&
                            & comm,ivarid)
INTEGER, INTENT(IN) :: comm, dest, ivarid, recvcount, recvtag,&
```

```
                    & sendcount, sendtag, source
INTEGER, INTENT(IN), DIMENSION(sendcount) :: sendbuf
INTEGER, INTENT(OUT), DIMENSION(recvcount) :: recvbuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Combines a send and a receive operation on integer data.

Arguments

sendbuf	Starting address of send buffer
sendcount	Number of data in send buffer
dest	Rank of destination process
sendtag	Message tag of send operation
recvbuf	Starting address of receive buffer
recvcount	Number of data in receive buffer
source	Rank of source process
recvtag	Message tag of receive operation
comm	Communicator id
ivarid	Variable key id (used for log info only, zero for undefined)

MPI call
MPI_sendrecv

comms_sendrecv_log

```
SUBROUTINE comms_sendrecv_log(sendbuf,sendcount,dest,sendtag,&
                            & recvbuf,recvcount,source,recvtag,&
                            & comm,ivarid)
INTEGER, INTENT(IN) :: comm, dest, ivarid, recvcount, recvtag,&
                    & sendcount, sendtag, source
LOGICAL, INTENT(IN), DIMENSION(sendcount) :: sendbuf
LOGICAL, INTENT(OUT), DIMENSION(recvcount) :: recvbuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Combines a send and a receive operation on logical data.

Arguments

sendbuf	Starting address of send buffer
sendcount	Number of data in send buffer
dest	Rank of destination process
sendtag	Message tag of send operation
recvbuf	Starting address of receive buffer
recvcount	Number of data in receive buffer
source	Rank of source process
recvtag	Message tag of receive operation
comm	Communicator id
ivarid	Variable key id (used for log info only, zero for undefined)

MPI call
MPI_sendrecv

comms_sendrecv_real

```
SUBROUTINE comms_sendrecv_real(sendbuf,sendcount,dest,sendtag,&
                             & recvbuf,recvcount,source,recvtag,&
                             & comm,ivarid)
INTEGER, INTENT(IN) :: comm, dest, ivarid, recvcount, recvtag,&
                     & sendcount, sendtag, source
REAL, INTENT(IN), DIMENSION(sendcount) :: sendbuf
REAL, INTENT(OUT), DIMENSION(recvcount) :: recvbuf
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Combines a send and a receive operation on real data.

Arguments

sendbuf	Starting address of send buffer
sendcount	Number of data in send buffer
dest	Rank of destination process
sendtag	Message tag of send operation
recvbuf	Starting address of receive buffer
recvcount	Number of data in receive buffer
source	Rank of source process
recvtag	Message tag of receive operation
comm	Communicator id
ivarid	Variable key id (used for log info only, zero for undefined)

MPI call
MPI_sendrecv

comms_waitall

```
SUBROUTINE comms_waitall(count,requests)
INTEGER, INTENT(IN) :: count
INTEGER, INTENT(INOUT), DIMENSION(count) :: requests
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Complete communications associated with an array of requests.

Arguments

count	Number of requests
requests	Array of requests

MPI call
MPI_waitall

error_MPI

```
SUBROUTINE error_MPI(mpiname,abort)
LOGICAL, INTENT(IN), OPTIONAL :: abort
CHARACTER (LEN=*), INTENT(IN) :: mpiname
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Report an error in a MPI routine. The program is aborted either in this routine or in the calling routine.

Arguments

mpiname	Name of the MPI routine where the error occurred
abort	Abort the program if present and .TRUE.. If not present, abortion is executed only if iopt_MPI_abort is set to 1.

MPI calls
MPI_error_string

error_MPI_comm

```
SUBROUTINE error_MPI(mpiname,idsrce,iddest)
CHARACTER (LEN=*), INTENT(IN) :: mpiname
INTEGER, INTENT(IN) :: iddest, idsrce
```

File
comms_MPI.F90

Type
Module subroutine

Purpose
Report an error in a MPI send or receive communication.

Arguments

mpiname	Name of the MPI routine where the error occurred
idsrce	Rank of the source (sending) process
iddest	Rank of the destination (receiving) process

31.6 Initialisation of derived types variables

Initialise derived type scalar and array variables. The intention is to provide an initial value for each component of the derived type variable. These values are not to be considered as default ones. Default values are set by the routines in *default_model.f90* (see Section 31.7 below).

exchange_comms_init

```
SUBROUTINE exchangecomms_init(comms)
TYPE (ExchComms), INTENT(OUT), DIMENSION(:) :: comms
```

File
datatypes_init.f90

Type
Module subroutine

Purpose
Initialise parameters for exchange communications.

Arguments

comms	Derived type variable to be initialised

filepars_init

```
SUBROUTINE filepars_init(filepars)
TYPE (FileParams), INTENT(INOUT)[, DIMENSION(:[,:[,:]])]  ::  filepars
```

File
datatypes_init.f90

Type
Generic module subroutine

Purpose
Initialise file attributes.

Arguments

filepars	Derived type variable (scalar or array) to be initialised

Non-generic versions

filepars_init_0d	Derived type scalar

filepars_init_1d Derived type vector

filepars_init_2d Derived type 2-D array

filepars_init_3d Derived type 3-D array

gridpars_init

```
SUBROUTINE gridpars_init(gridpars)
TYPE (GridParams), INTENT(OUT), DIMENSION(:,:) :: gridpars
```

File

datatypes_init.f90

Type

Module subroutine

Purpose

Initialise attributes of 2-D external data grids.

Arguments

gridpars Derived type variable to be initialised

hrelativecoords_init

```
SUBROUTINE hrelativecoords_init(hgrid,flag_undef)
LOGICAL, INTENT(IN) :: flag_undef
TYPE (FileParams), INTENT(OUT)[, DIMENSION(:[,:])]  ::  hgrd
```

File

datatypes_init.f90

Type

Generic module subroutine

Purpose

Initialise the horizontal relative coordinates of one or more data (model grid) locations.

Arguments

hgrid Derived type variable (scalar or array) to be initialised

flag_undef If .TRUE., values are set to undefined values. Otherwise they are set to zero.

Non-generic versions

hrelativecoords_init_0d Derived type scalar

hrelativecoords_init_1d Derived type vector

hrelativecoords_init_2d Derived type 2-D array

outgpars_init

```
SUBROUTINE outgpars_init(outgpars)
TYPE (OutGridParams), INTENT(OUT), DIMENSION(:) :: outgpars
```

File

datatypes_init.f90

Type

Module subroutine

Purpose

Initialise attributes of a user-defined output grid.

Arguments

outgpars Derived type array to be initialised

statlocs_init

```
SUBROUTINE statlocs_init(statlocs)
TYPE (StationLocs), INTENT(OUT), DIMENSION(:) :: statlocs
```

File

datatypes_init.f90

Type

Module subroutine

Purpose

Initialise attributes of user-defined output stations.

Arguments

outgpars Derived type array to be initialised

varatts_init

```
SUBROUTINE varatts_init(varatts)
TYPE (VariableAtts), INTENT(OUT)[, DIMENSION(:)] ::  varatts
```

File
datatypes_init.f90

Type
Generic module subroutine

Purpose
Initialise the attributes of program variables.

Arguments

varatts	Derived type variable (scalar or vector) to be initialised

Non-generic versions

varatts_init_0d	Derived type scalar
varatts_init_1d	Derived type vector

vrelativecoords_init

```
SUBROUTINE vrelativecoords_init(vgrid,flag_undef)
LOGICAL, INTENT(IN) :: flag_undef
TYPE (VRelativeCoords), INTENT(OUT) &
                    & [,DIMENSION(:[,:[,:[,]]])]  ::  vgrid
```

File
datatypes_init.f90

Type
Generic module subroutine

Purpose
Initialise the vertical relative coordinates of one or more data (model grid) locations.

Arguments

vgrid	Derived type variable (scalar or array) to be initialised
flag_undef	If .TRUE., values are set to undefined values. Otherwise they are set to zero.

Non-generic versions

vrelativecoords_init_0d Derived type scalar

vrelativecoords_init_1d Derived type vector

vrelativecoords_init_2d Derived type 2-D array

vrelativecoords_init_3d Derived type 3-D array

vrelativecoords_init_4d Derived type 4-D array

31.7 Default model setup

Default settings of all model parameters

default_ellvars

```
SUBROUTINE default_ellvars(ellvars)
TYPE (VariableAtts), INTENT(OUT), DIMENSION(14) :: ellvars
```

File

default_model.f90

Type

Module subroutine

Purpose

Attributes of tidal ellipse variables

Arguments

ellvars Derived type array containing the attributes of the tidal ellipse parameters. These defaults cannot be changed.

default_init_params

```
SUBROUTINE default_init_params
```

File

default_model.f90

Type

Module subroutine

Purpose

Default settings of parameters for monitoring

default_mod_params

```
SUBROUTINE default_mod_params
```

File
default_model.f90

Type
Module subroutine

Purpose
Default settings of model setup parameters (which can be re-defined in usrdef_mod_params)

default_out_files

```
SUBROUTINE default_out_files(filepars,file_type)
CHARACTER (LEN=2), INTENT(IN) :: file_type
TYPE (FileParams), INTENT(OUT), DIMENSION(:[,:])  ::  filepars
```

File
default_model.f90

Type
Generic module subroutine

Purpose
Default attributes of user-defined output files

Arguments

filepars	Derived type variable (vector or 2-D array) in which the attributes are stored
file_type	Type of user output
	'TS': time series
	'TA': time averaged
	'HR': harmonic residuals
	'HA': harmonic amplitudes
	'HP': harmonic phases
	'HE': tidal ellipse parameters

Non-generic versions

default_out_files_1d	Derived type vector
default_out_files_2d	Derived type 2-D array

default_out_files_grd

```
SUBROUTINE default_out_files_grd(filepars,file_type)
CHARACTER (LEN=2), INTENT(IN) :: file_type
TYPE (FileParams), INTENT(OUT), DIMENSION(:) :: filepars
```

File
default_model.f90

Type
Module subroutine

Purpose
Default settings for the attributes of an output grid file

Arguments

filepars	Output grid file
file_type	Type of user output
	'TS': time series
	'TA': time averaged
	'HR': harmonic residuals
	'HA': harmonic amplitudes
	'HP': harmonic phases
	'HE': tidal ellipse parameters

default_out_gpars

```
SUBROUTINE default_out_gpars(outgpars)
TYPE (OutGridParams), INTENT(OUT), DIMENSION(:) :: outgpars
```

File
default_model.f90

Type
Module subroutine

Purpose
Default settings for the attributes of a user-defined output grid

Arguments

outgpars	Attrinutes of the output grid

31.8 Error checking routines

Ensemble of routines for performing different kinds of error checking or to check for suspicious values of model setup parameters. Error or warning messages are written if requested.

check_space_limits

```
SUBROUTINE check_space_limits(slims,arrname,limmax)
CHARACTER (LEN=*), INTENT(IN) :: arrname
INTEGER, INTENT(IN) :: limmax
INTEGER, INTENT(IN), DIMENSION(3) :: slims
```

File
error_routines.F90

Type
Module subroutine

Purpose
Check the start/end/step values defining a spatial section on the model grid.

Arguments

slims	Start/end/increment values defining the array section
arrname	Array name
limmax	Maximum allowed value for the end index

check_space_limits_arr_struc

```
SUBROUTINE check_space_limits_arr_struc(slims,arrname,compname,&
                                        & limmax,ndims,indx)
CHARACTER (LEN=*), INTENT(IN) :: arrname, compname
INTEGER, INTENT(IN) :: limmax, ndims
INTEGER, INTENT(IN), DIMENSION(3) :: slims
INTEGER, INTENT(IN), DIMENSION(ndims) :: indx
```

File
error_routines.F90

Type
Module subroutine

Purpose

Check the start/end/step values, stored in a component of a derived type array, defining a spatial section on the model grid.

Arguments

slims	Start/end/increment values defining the array section
arrname	Name of the derived type array
compname	Name of the derived type component
limmax	Maximum allowed value for the end index
ndims	Rank of the derived type array
indx	Vector index of the derived type array

check_time_limits

```
SUBROUTINE check_time_limits(tlims,arrname,minstep,maxstep)
CHARACTER (LEN=*), INTENT(IN) :: arrname
INTEGER, INTENT(IN) :: maxstep, minstep
INTEGER, INTENT(INOUT), DIMENSION(3) :: tlims
```

File

error_routines.F90

Type

Module subroutine

Purpose

Check start/end/increment time index values.

Arguments

tlims	Start/end/increment values
arrname	Name of the tlims variable
minstep	Minimum allowed value for the start index
maxstep	Maximum allowed value for the end index

check_time_limits_arr_struc

```
SUBROUTINE check_time_limits_arr_struc(tlims,arrname,compname,&
                                       & minstep,maxstep,ndims,indx)
CHARACTER (LEN=*), INTENT(IN) :: arrname, compname
```

```
INTEGER, INTENT(IN) :: maxstep, minstep, ndims
INTEGER, INTENT(INOUT), DIMENSION(3) :: tlims
INTEGER, INTENT(IN), DIMENSION(ndims) :: indx
```

File
error_routines.F90

Type
Module subroutine

Purpose
Check the start/end/increment time index values, stored in a component of a derived type array.

Arguments

tlims	Start/end/increment values
arrname	Name of the derived type array
compname	Name of the derived type component
minstep	Minimum allowed value for the start index
maxstep	Maximum allowed value for the end index
ndims	Rank of the derived type array
indx	Vector index of the derived type array

error_abort

```
SUBROUTINE error_abort(prname,icode)
CHARACTER (LEN=*), INTENT(IN) :: prname
INTEGER, INTENT(IN) :: icode
```

File
error_routines.F90

Type
Module subroutine

Purpose
If errors have been detected from a previous error checking routine or by some error coding within a program routine, the error message associated with error code icode is displayed and the program is aborted.

Arguments

prname	Name of the routines where the error occurred
icode	Error code key id number

error_alloc

```
SUBROUTINE error_alloc(arrname,ndims,nshape,data_type,lenstr,abort)
LOGICAL, INTENT(IN), OPTIONAL :: abort
CHARACTER (LEN=*), INTENT(IN) :: arrname
INTEGER, INTENT(IN) :: data_type, ndims
INTEGER, INTENT(IN), OPTIONAL :: lenstr
INTEGER, INTENT(IN), DIMENSION(ndims) :: nshape
```

File
: *error_routines.F90*

Type
: Module subroutine

Purpose
: Write an appropriate error message when an error occurred during allocation of an array.

Arguments

arrname	Name of the array
ndims	Rank of the array
nshape	Shape of the array
data_type	COHERENS data type id
lenstr	Length of the array strings if the array contains character data
abort	Abort the program if present and .TRUE.. Otherwise the program is aborted in the calling routine.

error_alloc_struc

```
SUBROUTINE error_alloc_struc(arrname,ndims,nshape,structype,abort)
LOGICAL, INTENT(IN), OPTIONAL :: abort
CHARACTER (LEN=*), INTENT(IN) :: arrname, structype
INTEGER, INTENT(IN) :: ndims
INTEGER, INTENT(IN), DIMENSION(ndims) :: nshape
```

File
: *error_routines.F90*

Type
: Module subroutine

Purpose

Write an appropriate error message when an error occurred during allocation of a derived type array.

Arguments

arrname	Name of the array
ndims	Rank of the array
nshape	Shape of the array
structype	TYPE of the derived type array
abort	Abort the program if present and .TRUE.. Otherwise the program is aborted in the calling routine.

error_arg_var

```
SUBROUTINE error_arg_var(val,argname,fix)
CHARACTER (LEN=*), INTENT(IN) :: argname
CHARACTER (LEN=*) or INTEGER or LOGICAL, INTENT(IN) ::  val
CHARACTER (LEN=*) or INTEGER or LOGICAL , INTENT(IN),&
                                        & OPTIONAL ::&  fix
```

File

error_routines.F90

Type

Generic module subroutine

Purpose

Checks whether argument *val* in a routine call equals *fix*. It is clear that the two arguments must be of the same type and have the same string length in case they are of type CHARACTER. Argument *fix* is OPTIONAL only in case of CHARACTER arguments.

Arguments

val	Value of the routine argument
argname	FORTRAN name of the argument
fix	Allowed value for *val*. If not present and *val* is of type CHARACTER, its value is automatically taken as invalid.

Non-generic versions

error_arg_var_char The arguments are of type CHARACTER

error_arg_var_int	The arguments are of type INTEGER
error_arg_var_log	The arguments are of type LOGICAL

error_array_index

```
SUBROUTINE error_array_index(ival,arrname,minval,maxval,ndim)
CHARACTER (LEN=*), INTENT(IN) :: arrname
INTEGER, INTENT(IN) :: ival, minval, maxval, ndim
```

File

error_routines.F90

Type

Module subroutine

Purpose

Checks whether the value the index ival belonging to dimension ndim of an array is between the bounds minval and maxval.

Arguments

ival	Array index to be checked
arrname	Name of the array
minval	Lower array bound
maxval	Upper array bound
ndim	Array dimension to be checked

error_diff_vals_arrlist

```
SUBROUTINE error_diff_vals_arrlist(ilist,arrname,ndims,idim,&
                                   & indx,nozero)
LOGICAL, INTENT(IN), OPTIONAL :: nozero
CHARACTER (LEN=*), INTENT(IN) :: arrname
INTEGER, INTENT(IN) :: idim, ndims
INTEGER, INTENT(IN), DIMENSION(ndims) :: indx
INTEGER, INTENT(IN), DIMENSION(:) :: ilist
```

File

error_routines.F90

Type

Module subroutine

Purpose

Checks if the elements of the vector ilist are all different. Zero values are not allowed unless nozero is present and .TRUE.. The routine is called to check the values along a certain array dimension of an INTEGER array.

Arguments

ilist	List of integer values to be checked
arrname	Name of the array
ndims	Rank of the array
idim	Array dimension to which ilist applies
indx	Array index vector with the index for dimension idim omitted
nozero	Zeros are allowed only if present and .TRUE.

error_diff_vals_varlist

```
SUBROUTINE error_diff_vals_varlist(ilist,listname,nozero)
LOGICAL, INTENT(IN), OPTIONAL :: nozero
CHARACTER (LEN=*), INTENT(IN) :: listname
INTEGER, INTENT(IN), DIMENSION(:) :: ilist
```

File

error_routines.F90

Type

Module subroutine

Purpose

Checks if the elements of the vector ilist are all different. Zero values are not allowed unless nozero is present and .TRUE..

Arguments

ilist	List of integer values to be checked
listname	Name of the vector list
nozero	Zeros are allowed only if present and .TRUE.

error_dim_arr

```
SUBROUTINE error_dim_arr(nrank,arrname,nfix)
CHARACTER (LEN=*), INTENT(IN) :: arrname
INTEGER, INTENT(IN) :: nfix, nrank
```

File
error_routines.F90

Type
Module subroutine

Purpose
Checks whether an array has the rank given by nfix.

Arguments

nrank	Rank of the array
arrname	Name of the array
nfix	Assumed rank of the array

error_file

```
SUBROUTINE error_file(ierr,iounit,filepars,abort)
LOGICAL, INTENT(IN), OPTIONAL :: abort
INTEGER, INTENT(IN) :: ierr
INTEGER, INTENT(IN), OPTIONAL :: iounit
TYPE(FileParams), INTENT(IN), OPTIONAL :: filepars
```

File
error_routines.F90

Type
Module subroutine

Purpose
Report an input/output error detected during a file read/write.

Arguments

ierr	Error code (as given by its key id of the form ierrno_*)
iounit	File unit number (must be present if filepars is not present)
filepars	File attributes (must be present if iounit is not present)
abort	The program is aborted unless abort is present and .FALSE.

error_lbound_arr

```
SUBROUTINE error_lbound_arr(val,arrname,minval,matchmin,ndims,indx)
LOGICAL, INTENT(IN) :: matchmin
CHARACTER (LEN=*), INTENT(IN) :: arrname
INTEGER or REAL, INTENT(IN) ::  minval, val
INTEGER, INTENT(IN) :: ndims
INTEGER, INTENT(IN), DIMENSION(ndims) :: indx
```

File

error_routines.F90

Type

Generic module subroutine

Purpose

Checks whether the value *val* of an array element is greater than or equal to *minval* if matchmin is .TRUE. or greater than *minval* if matchmin is .FALSE..

Arguments

val	Value of the array element
arrname	Array name
minval	Lower bound for *val*
matchmin	If .TRUE., *val* may be equal to *minval*.
ndims	Rank of the array
indx	Vector index of the array element

Non-generic versions

error_lbound_arr_int val and minval are of type INTEGER

error_lbound_arr_real val and minval are of type REAL

error_lbound_arr_struc

```
SUBROUTINE error_lbound_arr_struc(val,arrname,compname,minval,&
                                  & matchmin,ndims,indx)
LOGICAL, INTENT(IN) :: matchmin
CHARACTER (LEN=*), INTENT(IN) :: arrname, compname
INTEGER or REAL, INTENT(IN) ::  minval, val
INTEGER, INTENT(IN) :: ndims
INTEGER, INTENT(IN), DIMENSION(ndims) :: indx
```

File

error_routines.F90

Type

Generic module subroutine

Purpose

Checks whether the value *val* of a component of a derived array element is greater than or equal to *minval* if matchmin is .TRUE. or greater than *minval* if matchmin is .FALSE..

Arguments

val	Value of the component of the array element
arrname	Name of the derived type array
compname	Name of the derived type component
minval	Lower bound for *val*
matchmin	If .TRUE., *val* may be equal to *minval*.
ndims	Rank of the array
indx	Vector index of the array element

Non-generic versions

error_lbound_arr_struc_int	val and minval are of type INTEGER
error_lbound_arr_struc_real	val and minval are of type REAL

error_lbound_var

```
SUBROUTINE error_lbound_var(val,varname,minval,matchmin)
LOGICAL, INTENT(IN) :: matchmin
CHARACTER (LEN=*), INTENT(IN) :: varname
INTEGER or REAL, INTENT(IN) ::  minval, val
```

File

error_routines.F90

Type

Generic module subroutine

Purpose

Checks whether *val* is greater than or equal to *minval* if matchmin is .TRUE. or greater than *minval* if matchmin is .FALSE..

Arguments

val	Value of the (scalar) variable
varname	Variable name
minval	Lower bound for *val*
matchmin	If .TRUE., *val* may be equal to *minval*.

Non-generic versions

error_lbound_var_int	val and minval are of type INTEGER
error_lbound_var_real	val and minval are of type REAL

error_limits_arr

```
SUBROUTINE error_limits_arr(val,arrname,minval,maxval,ndims,indx)
CHARACTER (LEN=*), INTENT(IN) :: arrname
INTEGER or REAL, INTENT(IN) ::  maxval, minval, val
INTEGER, INTENT(IN) :: ndims
INTEGER, INTENT(IN), DIMENSION(ndims) :: indx
```

File

error_routines.F90

Type

Generic module subroutine

Purpose

Checks whether the value *val* of an array element is not lower than *minval* and not greater than *maxval*.

Arguments

val	Value of the array element
arrname	Array name
minval	Lower bound for *val*
maxval	Upper bound for *val*
ndims	Rank of the array
indx	Vector index of the array element

Non-generic versions

error_limits_arr_int	val, minval, maxval are of type INTEGER
error_limits_arr_real	val, minval, maxval are of type REAL

error_limits_arr_struc

```
SUBROUTINE error_limits_arr_struc(val,arrname,compname,minval,&
                                  & maxval,ndims,indx)
CHARACTER (LEN=*), INTENT(IN) :: arrname, compname
INTEGER or REAL, INTENT(IN) ::  maxval, minval, val
INTEGER, INTENT(IN) :: ndims
INTEGER, INTENT(IN), DIMENSION(ndims) :: indx
```

File

error_routines.F90

Type

Generic module subroutine

Purpose

Checks whether the value *val* of a component of a derived array element is not lower than *minval* and not greater than *maxval*.

Arguments

val	Value of the component of the derived type array element
arrname	Name of the derived type array
compname	Name of the derived type component
minval	Lower bound for *val*
maxval	Upper bound for *val*
ndims	Rank of the array
indx	Vector index of the array element

Non-generic versions

error_limits_arr_struc_int	val, minval, maxval are of type INTEGER
error_limits_arr_struc_real	val, minval, maxval are of type REAL

error_limits_var

```
SUBROUTINE error_limits_var(val,varname,minval,maxval)
CHARACTER (LEN=*), INTENT(IN) :: varname
INTEGER or REAL, INTENT(IN) ::  maxval, minval, val
```

File

error_routines.F90

Type
Generic module subroutine

Purpose
Checks whether *val* is not lower than *minval* and not greater than *maxval*.

Arguments

val	Value of the (scalar) variable
varname	Variable name
minval	Lower bound for *val*
maxval	Upper bound for *val*

Non-generic versions

error_limits_var_int	val, minval, maxval are of type INTEGER
error_limits_var_real	val, minval, maxval are of type REAL

error_mult

```
SUBROUTINE error_mult(ival,varname,multval)
CHARACTER (LEN=*), INTENT(IN) :: varname
INTEGER, INTENT(IN) :: ival, multval
```

File
error_routines.F90

Type
Module subroutine

Purpose
Checks whether ival is a divisor of multval.

Arguments

ival	Integer input variable
varname	Variable name
multval	Integer multiple of ival

error_proc

```
SUBROUTINE error_proc(ierr,abort)
LOGICAL, INTENT(IN), OPTIONAL :: abort
INTEGER, INTENT(IN) :: ierr
```

File
error_routines.F90

Type
Module subroutine

Purpose
Write the number of the error code and the name of the routine where the error occurred to the error file.

Arguments

ierr	Number of the error code
abort	Abort the program if present and .TRUE.

error_shape

```
SUBROUTINE error_shape(arrshape,arrname,fixshape,ndim)
CHARACTER (LEN=*), INTENT(IN) :: arrname
INTEGER, INTENT(IN) :: ndim
INTEGER, INTENT(IN) , DIMENSION(ndim) :: arrshape, fixshape
```

File
error_routines.F90

Type
Module subroutine

Purpose
Check whether shape arrshape equals the vector shape.

Arguments

arrshape	Shape of the array
arrname	Array name
fixshape	Vector to which arrshape should be equal
ndim	Array rank

error_ubound_arr

```
SUBROUTINE error_ubound_arr(val,arrname,maxval,matchmax,&
                          & ndims,indx)
LOGICAL, INTENT(IN) :: matchmax
CHARACTER (LEN=*), INTENT(IN) :: arrname
INTEGER or REAL, INTENT(IN) ::  maxval, val
INTEGER, INTENT(IN) :: ndims
INTEGER, INTENT(IN), DIMENSION(ndims) :: indx
```

File

error_routines.F90

Type

Generic module subroutine

Purpose

Check whether the value *val* of an array element is not greater than *maxval* if matchmax is .TRUE. or lower than *maxval* if matchmax is .FALSE..

Arguments

val	Value of the array element
arrname	Array name
maxval	Upper bound for *val*
matchmax	If .TRUE., *val* may be equal to *maxval*.
ndims	Rank of the array
indx	Vector index of the array element

Non-generic versions

error_ubound_arr_int val and maxval are of type INTEGER

error_ubound_arr_real val and maxval are of type REAL

error_ubound_arr_struc

```
SUBROUTINE error_ubound_arr_struc(val,arrname,compname,maxval,&
                                & matchmax,ndims,indx)
LOGICAL, INTENT(IN) :: matchmax
CHARACTER (LEN=*), INTENT(IN) :: arrname, compname
INTEGER or REAL, INTENT(IN) ::  maxval, val
INTEGER, INTENT(IN) :: ndims
INTEGER, INTENT(IN), DIMENSION(ndims) :: indx
```

File
error_routines.F90

Type
Generic module subroutine

Purpose
Check whether the value *val* of a component of a derived type array element is not greater than *maxval* if matchmax is .TRUE. or lower than *maxval* if matchmax is .FALSE..

Arguments

val	Value of the component of the array element
arrname	Name of the derived type array
compname	Name of the derived type component
maxval	Upper bound for *val*
matchmax	If .TRUE., *val* may be equal to *maxval*.
ndims	Rank of the array
indx	Vector index of the array element

Non-generic versions

error_ubound_arr_struc_int val and maxval are of type INTEGER

error_ubound_arr_struc_real val and maxval are of type REAL

error_ubound_var

```
SUBROUTINE error_ubound_var(val,varname,maxval,matchmax)
LOGICAL, INTENT(IN) :: matchmax
CHARACTER (LEN=*), INTENT(IN) :: varname
INTEGER or REAL, INTENT(IN) ::  maxval, val
```

File
error_routines.F90

Type
Generic module subroutine

Purpose
Check whether *val* is not greater than *maxval* if matchmax is .TRUE. or lower than *maxval* if matchmax is .FALSE..

Arguments

val	Value of the (scalar) variable
varname	Variable name
maxval	Upper bound for *val*
matchmax	If .TRUE., *val* may be equal to *maxval*.

Non-generic versions

error_ubound_var_int	val and maxval are of type INTEGER
error_ubound_var_real	val and maxval are of type REAL

error_vals_arr_char

```
SUBROUTINE error_vals_arr_char(cval,arrname,string,ndims,indx)
CHARACTER (LEN=*), INTENT(IN) :: arrname, cval, string
INTEGER, INTENT(IN) :: ndims
INTEGER, INTENT(IN), DIMENSION(ndims) :: indx
```

File

error_routines.F90

Type

Module subroutine

Purpose

Check whether the character string array element cval (without trailing blanks) has an allowed value given by string.

Arguments

cval	Character string or integer array element
arrname	Array name
string	Character string to which cval should be equal
ndims	Array rank
indx	Vector index of the array element

error_vals_arr_int

```
SUBROUTINE error_vals_arr_int(ival,arrname,nlist,ndims,ifix,indx)
CHARACTER (LEN=*), INTENT(IN) :: arrname
INTEGER, INTENT(IN) :: ival, ndims, nlist
INTEGER, INTENT(IN), DIMENSION(nlist) :: ifix
INTEGER, INTENT(IN), DIMENSION(ndims) :: indx
```

File
error_routines.F90

Type
Module subroutine

Purpose
Check whether an element ival of an integer array equals one of the values in the vector list ifix.

Arguments

ival	Integer array element
arrname	Array name
nlist	Size of the vector ifix
ndims	Array rank
ifix	Vector list of integers which should contain ival
indx	Vector index of the array element

error_vals_arr_struc_char

```
SUBROUTINE error_vals_arr_struc_char(cval,arrname,compname,string,ndims,indx)
CHARACTER (LEN=*), INTENT(IN) :: arrname, compname, cval, string
INTEGER, INTENT(IN) :: ndims
INTEGER, INTENT(IN), DIMENSION(ndims) :: indx
```

File
error_routines.F90

Type
Module subroutine

Purpose
Check whether the string component cval (without trailing blanks) of a derived type array has an allowed value given by string.

Arguments

cval	Character string component of the derived type array
arrname	Name of derived type array
compname	Name of the derived type component

string	Character string to which cval (without trailing blanks) should be equal.
ndims	Array rank
indx	Vector index of the array element

error_vals_arr_struc_int

```
SUBROUTINE error_vals_arr_struc_int(ival,arrname,compname,nlist,ndims,&
                                    & ifix,indx)
CHARACTER (LEN=*), INTENT(IN) :: arrname, compname
INTEGER, INTENT(IN) :: ival, ndims, nlist
INTEGER, INTENT(IN), DIMENSION(nlist) :: ifix
INTEGER, INTENT(IN), DIMENSION(ndims) :: indx
```

File
error_routines.F90

Type
Module subroutine

Purpose
Check whether the integer component ival of a derived type array equals one of the values in the list ifix.

Arguments

ival	Integer component of the derived type array
arrname	Name of derived type array
compname	Name of the derived type component
nlist	Size of the vector ifix
ndims	Array rank
ifix	Vector list of integers which should contain ival
indx	Vector index of the array element

error_vals_var_char

```
SUBROUTINE error_vals_var_char(cval,varname,string)
CHARACTER (LEN=*), INTENT(IN) :: cval, string, varname
```

File
error_routines.F90

Type
Module subroutine

Purpose
Check whether the string cval (without trailing blanks) has an allowed value given by string.

Arguments

cval	Character string to be checked
varname	Variable name
string	Character string to which *cval* should be equal.

error_vals_var_int

```
SUBROUTINE error_vals_var_int(ival,varname,nlist,ifix)
CHARACTER (LEN=*), INTENT(IN) :: varname
INTEGER, INTENT(IN) :: ival, nlist
INTEGER, INTENT(IN), DIMENSION(nlist) :: ifix
```

File
error_routines.F90

Type
Module subroutine

Purpose
Check whether the integer parameter ival equals one of the values in the vector list ifix.

Arguments

ival	Integer parameter to be checked
varname	Variable name
nlist	Size of the vector list ifix
ifix	Vector list of integers which should contain ival

error_value_arr

```
SUBROUTINE error_value_arr(val,arrname,fixval,ndims,indx)
CHARACTER (LEN=*), INTENT(IN) :: arrname
INTEGER or LOGICAL, INTENT(IN) ::  fixval, val
INTEGER, INTENT(IN) :: ndims
INTEGER, INTENT(IN), DIMENSION(ndims) :: indx
```

File

error_routines.F90

Type

Generic module subroutine

Purpose

Check whether the integer or logical array element *val* equals the integer or logical value *fixval*.

Arguments

val	Integer or logical array element
arrname	Array name
fixval	Integer or logical value to which *val* should be equal
ndims	Array rank
indx	Vector index of the array element

Non-generic versions

error_value_arr_int val and fixval are of type INTEGER

error_value_arr_log val and fixval are of type LOGICAL

error_value_arr_struc

```
  SUBROUTINE error_value_arr_struc(val,arrname,compname,fixval,&
                                & ndims,indx)
CHARACTER (LEN=*), INTENT(IN) :: arrname, compname
INTEGER or LOGICAL, INTENT(IN) ::  fixval, val
INTEGER, INTENT(IN) :: ndims
INTEGER, INTENT(IN), DIMENSION(ndims) :: indx
```

File

error_routines.F90

Type

Generic module subroutine

Purpose

Check whether the value *val* of a component of a derived type array element equals *fixval*.

Arguments

val	Value of the component of the derived type array element
arrname	Name of derived type array
compname	Name of derived type component
fixval	Value to which *val* should be equal
ndims	Array rank
indx	Vector index of the array element

Non-generic versions

error_value_arr_struc_int val and fixval are of type INTEGER

error_value_arr_struc_log val and fixval are of type LOGICAL

error_value_var

```
SUBROUTINE error_value_var(val,varname,fixval)
CHARACTER (LEN=*), INTENT(IN) :: varname
INTEGER or LOGICAL, INTENT(IN) ::  fixval, val
```

File

error_routines.F90

Type

Generic module subroutine

Purpose

Check whether the integer or logical variable *val* equals the integer or logical value *fixval*.

Arguments

val	Integer or logical variable
varname	Variable name
fixval	Integer or logical value to which *val* should be equal

Non-generic versions

error_value_var_int val and fixval are of type INTEGER

error_value_var_log val and fixval are of type LOGICAL

warning_reset_arr

```
SUBROUTINE warning_reset_arr(val,arrname,set,ndims,indx)
CHARACTER (LEN=*), INTENT(IN) :: arrname
CHARACTER (LEN=*), INTEGER or LOGICAL, INTENT(IN) ::  set
CHARACTER (LEN=*), INTEGER or LOGICAL, INTENT(INOUT) ::  val
INTEGER, INTENT(IN) :: ndims
INTEGER, INTENT(IN), DIMENSION(ndims) :: indx
```

File
: *error_routines.F90*

Type
: Generic module subroutine

Purpose
: Reset the array element *val* to *set* with a warning message if needed.

Arguments

val	Character string, integer or logical array element
arrname	Array name
set	Value to which *val* should be reset
ndims	Array rank
indx	Vector index of the array element

Non-generic versions

warning_reset_arr_char	Resets a character string array element
warning_reset_arr_int	Resets an integer array element
warning_reset_arr_log	Resets a logical array element

warning_reset_arr_struc

```
SUBROUTINE warning_reset_arr_struc(val,arrname,compname,set,&
                                   & ndims,indx)
CHARACTER (LEN=*), INTENT(IN) :: arrname, compname
CHARACTER (LEN=*), INTEGER or LOGICAL, INTENT(IN) ::  set
CHARACTER (LEN=*), INTEGER or LOGICAL, INTENT(INOUT) ::  val
INTEGER, INTENT(IN) :: ndims
INTEGER, INTENT(IN), DIMENSION(ndims) :: indx
```

File
error_routines.F90

Type
Generic module subroutine

Purpose
Reset array element *val* to *set* with a warning message if needed.

Arguments

val	Character string, integer or logical component
arrname	Name of the derived type array
compname	Name of the derived type component
set	Value to which *val* should be reset
ndims	Rank of derived type array
indx	Vector index of the array element

Non-generic versions

warning_reset_arr_struc_char	Resets a character string component
warning_reset_arr_struc_int	Resets an integer component
warning_reset_arr_struc_log	Resets a logical component

warning_reset_var

```
SUBROUTINE warning_reset_var(val,varname,set)
CHARACTER (LEN=*), INTENT(IN) :: varname
CHARACTER (LEN=*), INTEGER, LOGICAL or REAL, INTENT(IN) ::  set
CHARACTER (LEN=*), INTEGER, LOGICAL or REAL, INTENT(INOUT) ::  val
```

File
error_routines.F90

Type
Generic module subroutine

Purpose
Reset *val* to *set* with a warning message if needed.

Arguments

val	Character string, integer, logical or real variable

varname	Variable name
set	Value to which *val* should be reset

Non-generic versions

warning_reset_var_char	Resets a character string variable
warning_reset_var_int	Resets an integer variable
warning_reset_var_log	Resets a logical variable
warning_reset_var_real	Resets a real variable

warning_ubound_var_real

```
SUBROUTINE warning_ubound_var_real(rval,varname,maxval,matchmax)
LOGICAL, INTENT(IN) :: matchmax
CHARACTER (LEN=*), INTENT(IN) :: varname
REAL, INTENT(IN) :: maxval, rval
```

File

error_routines.F90

Type

Module subroutine

Purpose

Writes a warning message if the real variable rval is not greater than maxval if matchmax is .TRUE. or lower than maxval if matchmax is .FALSE.

Arguments

rval	Real variable to be checked
varname	Variable name
maxval	Upper bound for rval
matchmax	If .TRUE., rval may be equal to maxval.

31.9 Grid interpolation from and to the model grid

The routines in this files deal with

- the relative coordinates of external (gridded or non-gridded) data points with respect to a rectangular model grid

- the relative coordinates of the model grid with respect to an external rectangular data grid
- interpolation of external data on the model grid, model grid data on an external grid or between external grids

data_to_data_vcoords_2d

```
SUBROUTINE data_to_data_vcoords_2d(zin,zout,vcoords,nhdat,
                                   & nzdatin,nzdatout,nzeff)
INTEGER, INTENT(IN) :: nhdat, nzdatin, nzdatout, nzeff
REAL, INTENT(IN), DIMENSION(nhdat,nzdatin) :: zin
REAL, INTENT(IN), DIMENSION(nhdat,nzdatout) :: zout
TYPE (VRelativeCoords), INTENT(OUT),&
                      & DIMENSION(nhdat,nzdatout) :: vcoords
```

File
: *grid_interp.F90*

Type
: Module subroutine

Purpose
: Returns the relative vertical coordinates of input data with respect to a series of output data locations.

Arguments

zin	Vertical coordinates (measured as the positive distance to the sea surface) of the input data
zout	Vertical coordinates (measured as the positive distance to the sea surface) of the points to which the data are to be interpolated. The interpolating and interpolated data have the same horizontal locations.
vcoords	Returned derived type array of relative coordinates
nhdat	Number of horizontal data locations.
nzdatin	Number of interpolating data along the vertical
nzdatout	Number of interpolated data along the vertical
nzeff	"Effective" number of interpolated data along the vertical. This means that interpolation is only performed for levels 1 to nzeff (equal or lower than nzdatout).

data_to_model_hcoords_glb

```
SUBROUTINE data_to_model_hcoords_glb(xdat,ydat,hcoords,n1dat,&
                                     & n2dat,cnode,intflag)
CHARACTER (LEN=1), INTENT(IN) :: cnode
INTEGER, INTENT(IN) :: intflag, n1dat, n2dat
REAL, INTENT(IN), DIMENSION(n1dat,n2dat) :: xdat, ydat
TYPE (HRelativeCoords), INTENT(OUT),&
                      & DIMENSION(n1dat*n2dat) :: hcoords
```

File

grid_interp.F90

Type

Module subroutine

Purpose

Returns the relative horizontal coordinates of a gridded array of external data points with respect to the global model grid.

Arguments

xdat	X-coordinates of the data points
ydat	Y-coordinates of the data points
hcoords	Returned array of relative coordinates
n1dat	X-dimension of the external data grid
n2dat	Y-dimension of the external data grid
cnode	Nodal type of the model grid points used for interpolation ('C', 'U', 'V')
intflag	Type of flagging for invalid (land) points 1: all locations are valid, interpolation to nearest wet points if necessary 2: locations are valid only if at least one neighbouring point is wet 3: locations are only valid if all neighbouring points are wet

data_to_model_vcoords_1d

```
SUBROUTINE data_to_model_vcoords_1d(zcoord,vcoords,i,j,nzdat,cnode)
CHARACTER (LEN=1), INTENT(IN) :: cnode
```

```
INTEGER, INTENT(IN) :: i, j, nzdat
REAL, INTENT(IN), DIMENSION(nzdat) :: zcoord
TYPE (VRelativeCoords), INTENT(OUT), DIMENSION(nzdat) :: vcoords
```

File
grid_interp.F90

Type
Module subroutine

Purpose
Returns the relative vertical coordinates of a 1-D vertical grid (with horizontal position on the model grid) with respect to the model grid.

Arguments

zcoord	Vertical coordinates (measured as the positive distance to the sea surface) of the data locations
vcoords	Returned array of vertical relative coordinates
i	X-index of the data locations on the model grid
j	Y-index of the data locations on the model grid
nzdat	Number of vertical levels of the 1-D data grid
cnode	Nodal type of the horizontal locations on the model grid ('C', 'U', 'V')

data_to_model_vcoords_2d

```
SUBROUTINE data_to_model_vcoords_2d(zcoord,vcoords,nzdat,cnode)
CHARACTER (LEN=1), INTENT(IN) :: cnode
INTEGER, INTENT(IN) :: nzdat
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nzdat) :: zcoord
TYPE (VRelativeCoords), INTENT(OUT), &
                      & DIMENSION(ncloc,nrloc,nzdat) :: vcoords
```

File
grid_interp.F90

Type
Module subroutine

Purpose
Returns the relative coordinates of a 3-D data grid with respect to the

model grid. The data grid is assumed to be positioned horizontally on the model grid. Dry cells are not taken into account.

Arguments

zcoord	Vertical coordinates (measured as the positive distance to the sea surface) of the data locations
vcoords	Returned array of vertical relative coordinates
nzdat	Number of vertical levels in the data array
cnode	Nodal type of the horizontal locations on the model grid ('C', 'U', 'V')

data_to_model_vcoords_3d

```
SUBROUTINE data_to_model_vcoords_3d(zcoord,hcoords,vcoords,&
                                    & n1dat,n2dat,n3dat,cnode)
CHARACTER (LEN=1), INTENT(IN) :: cnode
INTEGER, INTENT(IN) :: n1dat, n2dat, n3dat
REAL, INTENT(IN), DIMENSION(n1dat,n2dat,n3dat) :: zcoord
TYPE (HRelativeCoords), INTENT(IN), DIMENSION(n1dat*n2dat) :: hcoords
TYPE (VRelativeCoords), INTENT(OUT), &
                      & DIMENSION(2,2,n1dat*n2dat,n3dat) :: vcoords
```

File

grid_interp.F90

Type

Module subroutine

Purpose

Returns the relative coordinates of a 3-D data grid with respect to the model grid. The horizontal grids are different. Since horizontal interpolation is performed after vertical one and vertical model grid points with the same vertical index do not have the vertical Z-coordinate, the vertical relative coordinates are to be taken at the four model grid points surrounding the data point in the horizontal. This explains why the first two dimensions of vcoords have a size of 2.

Arguments

zcoord	Vertical coordinates (measured as the positive distance to the sea surface) of the data grid

hcoords	Horizontal relative coordinates of the data grid with respect to the model grid.
vcoords	Returned array of vertical relative coordinates. The first two dimensions refer to the four adjacent model grid locations in the X- and Y-directions.
n1dat	X-dimension of the data grid.
n2dat	Y-dimension of the data grid.
n3dat	Vertical dimension of the data grid.
cnode	Nodal type of the horizontal locations on the model grid ('C', 'U', 'V')

intpol_data_to_model_2d

```
SUBROUTINE intpol_data_to_model_2d(invals,outvals,ncin,nrin,&
                                   & nosize,surfgrid,datflag,land,&
                                   & inflag,outflag)
LOGICAL, INTENT(IN) :: land
INTEGER, INTENT(IN) :: datflag, ncin, nosize, nrin
REAL, INTENT(IN), OPTIONAL :: inflag, outflag
REAL, INTENT(IN), DIMENSION(ncin,nrin,nosize) :: invals
REAL, INTENT(OUT), DIMENSION(ncloc,nrloc,nosize) :: outvals
TYPE (HRelativeCoords), INTENT(IN),&
                      & DIMENSION(ncloc,nrloc) :: surfgrid
```

File

grid_interp.F90

Type

Module subroutine

Purpose

Interpolate a 2-D gridded horizontal data array on the model grid. Data and model grid are assumed to be different.

Arguments

invals	Array of data values
outvals	Returned data interpolated on the model grid
ncin	X-dimension of the data grid
nrin	Y-dimension of the data grid.

nosize	Number of data variables
surfgrid	Relative (horizontal) coordinates of the data grid with respect to the model grid
datflag	Enables flagging in case of invalid data points 0: flagging is disabled 1: flagging is enabled, no extrapolation is allowed 2: flagging is enabled, extrapolation is allowed
land	Land points on the model grid are included if set to .TRUE.
inflag	Flag for invalid input data which must be defined if datflag>0. If not present, its value is set to the minimum value real_min used for flagged data.
outflag	Replacement flag at model grid locations where no interpolation could be performed. If not present, its value is set to the undefined value real_fill.

intpol1d_model_to_dep

```
SUBROUTINE intpol1d_model_to_dep(invals,outval,i,j,dep,cnode,nzmin,nzmax,&
                                 & outflag)
CHARACTER (LEN=1), INTENT(IN) :: cnode
INTEGER, INTENT(IN) :: i, j
INTEGER, INTENT(IN), OPTIONAL :: nzmax, nzmin
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN) :: dep
REAL, INTENT(IN), DIMENSION(:) :: invals
REAL, INTENT(OUT) :: outval
```

File

grid_interp.F90

Type

Module subroutine

Purpose

Interpolate a vertical model profile at a given depth

Arguments

invals	Vertical input profile
outval	Returned value interpolated at the specified depth

i	(Local) X-index of the model grid location
j	(Local) Y-index of the model grid location
dep	Interpolation depth (positive distance from the free surface)
cnode	Vertical grid node where invals is defined ('C' or 'W')
nzmin	Lowest physical vertical level in case invals is defined at the W-nodes (1 otherwise)
nzmax	Highest physical vertical level in case invals is defined at the W-nodes (nz otherwise)
outflag	Replacement flag if dep is larger than the water depth at the given location

intpol_model_to_data_2d

```
SUBROUTINE intpol2d_model_to_data_2d(invals,outvals,hcoords,nhdims,&
                                     & nhdat,nosize,cnode,&
                                     & datvals,outflag)
CHARACTER (LEN=1), INTENT(IN) :: cnode
INTEGER, INTENT(IN) :: nhdat, nosize
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(4) :: nhdims
REAL, INTENT(INOUT), OPTIONAL, DIMENSION(nhdat) :: datvals
REAL, INTENT(OUT), DIMENSION(nhdat,nosize) :: outvals
REAL, INTENT(IN), DIMENSION(1-nhdims(1):ncloc+nhdims(2),&
                            & 1-nhdims(3):nrloc+nhdims(4),&
                            & nosize) :: invals
TYPE (HRelativeCoords), INTENT(IN), DIMENSION(nhdat) :: hcoords
```

File

grid_interp.F90

Type

Module subroutine

Purpose

Interpolate a 2-D gridded horizontal model array at a series of data locations.

Arguments

invals	2-D model grid array

outvals	Returned model data interpolated at the external data locations
hcoords	Relative (horizontal) coordinates of the data points with respect to the model grid
nhdims	Halo sizes of the model array (WESN directions)
nhdat	Number of horizontal data locations
nosize	Number of data variables for which interpolation is applied (added as an extra array dimension)
cnode	Nodal type of the variable(s) on the model grid ('C', 'U', 'V')
datvals	Optional array of data values. If present, its values are flagged at points where the relative coordinates are given as undefined.
outflag	Optional replacement flag at data locations where no interpolation could be performed. Otherwise the flag is set to the undefined value.

intpol3d_data_to_data_2d

```
SUBROUTINE intpol3d_data_to_data_2d(invals,outvals,vcoords,nhdat,&
                                    & nzdatin,nzdatout,nzeff,&
                                    & nosize,outflag)
INTEGER, INTENT(IN) :: nhdat, nosize, nzdatin, nzdatout, nzeff
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(nhdat,nzdatin,nosize) :: invals
REAL, INTENT(OUT), DIMENSION(nhdat,nzdatout,nosize) :: outvals
TYPE (VRelativeCoords), INTENT(IN),&
                      & DIMENSION(nhdat,nzdatout) :: vcoords
```

File

grid_interp.F90

Type

Module subroutine

Purpose

Interpolate data vertically from a 3-D data array to another 3-D output data array. The two grids share the same horizontal locations.

Arguments

invals	Interpolating data array, defined on the input grid
outvals	Returned interpolated data, defined on the output grid
vcoords	Vertical relative coordinates of the data points with respect to the output grid
nhdat	Number of horizontal locations on both grids
nzdatin	Vertical dimension of the input grid
nzdatout	Vertical dimension of the output grid
nzeff	"Effective" number of interpolated data along the vertical. This means that interpolation is only performed for levels 1 to nzeff (equal or lower than nzdatout).
nosize	Number of data variables for which interpolation is applied (added as an extra array dimension)
outflag	Optional replacement flag at data locations where no interpolation could be performed. Otherwise the flag is set to the undefined value.

intpol3d_model_to_data_1d

```
SUBROUTINE intpol3d_model_to_data_1d(invals,outvals,vcoords,nzdim,&
                                     & nzdat,nosize,datvals,outflag)
INTEGER, INTENT(IN) :: nosize, nzdat, nzdim
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(INOUT), OPTIONAL, DIMENSION(nzdat) :: datvals
REAL, INTENT(IN), DIMENSION(nzdim,nosize) :: invals
REAL, INTENT(OUT), DIMENSION(nzdat,nosize) :: outvals
TYPE (VRelativeCoords), INTENT(IN), DIMENSION(nzdat) :: vcoords
```

File

grid_interp.F90

Type

Module subroutine

Purpose

Interpolate model data vertically to an external (1-D) vertical grid. Model and external grid are assumed to be defined along the same horizontal grid, but the interpolation is taken at one horizontal location only.

Arguments

invals	Vertical profile on the model grid
outvals	Returned profile of model data interpolated along the external vertical grid
vcoords	Vertical relative coordinates of the model locations with respect to the vertical external grid
nzdim	Vertical dimension of the model grid
nzdat	Vertical dimension of the external grid
nosize	Number of data variables for which interpolation is applied (added as an extra array dimension)
datvals	Optional array of data values. If present, its values are flagged at points where the relative coordinates are given as undefined.
outflag	Optional replacement flag at data locations where no interpolation could be performed. Otherwise the flag is set to the undefined value real_fill.

intpol3d_model_to_data_2d

```
SUBROUTINE intpol3d_model_to_data_2d(invals,outvals,vcoords,nhdims,&
                                     & nzdim,nzdat,nosize,&
                                     & datvals,outflag)
INTEGER, INTENT(IN) :: nosize, nzdat, nzdim
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(4) :: nhdims
REAL, INTENT(IN), DIMENSION(1-nhdims(1):ncloc+nhdims(2),&
                           & 1-nhdims(3):nrloc+nhdims(4),&
                           & nzdim,nosize) :: invals
REAL, INTENT(OUT),&
    & DIMENSION(ncloc,nrloc,nzdat,nosize) :: outvals
REAL, INTENT(INOUT), OPTIONAL,&
    & DIMENSION(ncloc,nrloc,nzdat) :: datvals
TYPE (VRelativeCoords), INTENT(IN),&
    & DIMENSION(ncloc,nrloc,nzdat) :: vcoords
```

File

grid_interp.F90

Type

Module subroutine

Purpose

Interpolate model data (vertically) to an external 3-D grid. Model and external grid are assumed to be defined along the same horizontal grid.

Arguments

invals	Model grid array
outvals	Returned model data interpolated on the external grid
vcoords	Vertical relative coordinates of the model grid with respect to the vertical external grid
nhdims	Halo sizes of model array (WESN directions)
nzdim	Vertical dimension of the model grid
nzdat	Vertical dimension of the external grid
nosize	Number of data variables for which interpolation is applied (added as an extra array dimension)
datvals	Optional array of data values. If present, its values are flagged at points where the relative coordinates are given as undefined.
outflag	Optional replacement flag at data locations where no interpolation could be performed. Otherwise the flag is set to the undefined value.

intpol3d_model_to_data_3d

```
SUBROUTINE intpol3d_model_to_data_3d(invals,outvals,hcoords,&
                       & vcoords,nhdims,nzdim,nhdat,nzdat,&
                       & nosize,cnode,datvals,outflag)
CHARACTER (LEN=1), INTENT(IN) :: cnode
INTEGER, INTENT(IN) :: nhdat, nosize, nzdat, nzdim
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(4) :: nhdims
REAL, INTENT(INOUT), OPTIONAL, DIMENSION(nhdat,nzdat) :: datvals
REAL, INTENT(OUT), DIMENSION(nhdat,nzdat,nosize) :: outvals
REAL, INTENT(IN), DIMENSION(1-nhdims(1):ncloc+nhdims(2),&
                          & 1-nhdims(3):nrloc+nhdims(4),&
                          & nzdim,nosize) :: invals
TYPE (HRelativeCoords), INTENT(IN), DIMENSION(nhdat) :: hcoords
TYPE (VRelativeCoords), INTENT(IN),&
                      & DIMENSION(2,2,nhdat,nzdat) :: vcoords
```

File

grid_interp.F90

Type

Module subroutine

Purpose

Interpolate model data to an external 3-D grid. Model and external grids are assumed to be different in the horizontal and vertical.

Arguments

invals	Model grid array
outvals	Returned model data interpolated on the external grid
hcoords	Horizontal relative coordinates of the model grid with respect to the external grid
vcoords	Vertical relative coordinates of the model grid with respect to the external grid. Since horizontal interpolation is performed before the vertical one and vertical model grid points with the same vertical index do not have the same vertical Z-coordinate, the vertical relative coordinates are to be taken at the four model grid points surrounding the data point in the horizontal. The explains the two extra dimensions of size 2 used to store the values at these adjacent points.
nhdims	Halo sizes of the model array (WESN directions)
nzdim	Vertical dimension of the model grid
nhdat	Size of the external grid in the horizontal
nzdat	Vertical dimension of the external grid
nosize	Number of data variables for which interpolation is applied (added as an extra array dimension)
cnode	Nodal type of the variable(s) on the model grid ('C', 'U', 'V')
datvals	Optional array of data values. If present, its values are flagged at points where the relative coordinates are given as undefined.
outflag	Optional replacement flag at data locations where no interpolation could be performed. Otherwise the flag is set to the undefined value real_fill.

model_to_data_coords

```
SUBROUTINE model_to_data_coords(idgrd,ifil,surfgrid,xdat,ydat,&
                              & ncdat,nrdat,cnode)
CHARACTER (LEN=1), INTENT(IN) :: cnode
INTEGER, INTENT(IN) :: idgrd, ifil, ncdat, nrdat
REAL, INTENT(IN), DIMENSION(ncdat,nrdat) :: xdat, ydat
TYPE (HRelativeCoords), INTENT(OUT),&
                      & DIMENSION(ncloc,nrloc) :: surfgrid
```

File
: *grid_interp.F90*

Type
: Module subroutine

Purpose
: Returns the horizontal relative coordinates of the model grid with respect to an external (2-D) grid.

Arguments

idgrd	Key grid id of the external grid
ifil	File number of the external data grid (currently 1)
surfgrid	Returned horizontal relative coordinates of the model grid with respect to the external grid
xdat	X-coordinates of the external data grid
ydat	Y-coordinates of the external data grid
ncdat	X-dimension of the external data grid
nrdat	Y-dimension of the external data grid
cnode	Nodal type of the model grid locations ('C', 'U', 'V')

31.10 Routines performing operations on the model grid

construct_rectgrid

```
SUBROUTINE construct_rectgrid(xstart,ystart,delx,dely,xcoord,ycoord,&
                            & nxgrd,nygrd)
INTEGER, INTENT(IN) :: nxgrd, nygrd
```

```
REAL, INTENT(IN) :: xstart, ystart
REAL, INTENT(IN) [, DIMENSION(nxgrid)] ::  delx
REAL, INTENT(IN) [, DIMENSION(nygrid)] ::  dely
REAL, INTENT(OUT), DIMENSION(nxgrd,nygrd) :: xcoord, ycoord
```

File

grid_routines.f90

Type

Generic module subroutine

Purpose

Construct a uniform rectangular grid knowing its size, grid spacings and the position of the Southwest corner.

Arguments

xstart	X-coordinate of the Southwest corner at the (global) grid location (1,1) [m or degrees longitude]
ystart	Y-coordinate of the Southwest corner at the (global) grid location (1,1) [m or degrees latitude]
delx	Grid spacing(s) in the X-direction [m or degrees longitude]
dely	Grid spacing(s) in the Y-direction [m or degrees latitude]
nxgrd	Size of the grid in the X-direction
nygrd	Size of the grid in the Y-direction
xcoord	Returned X-coordinates [m or degrees longitude]
ycoord	Returned Y-coordinates [m or degrees latitude]

Non-generic versions

construct_rectgrid_nonunif	Non-uniform rectangular grid
construct_rectgrid_unif	Uniform rectangular grid

convert_loc_to_char

```
FUNCTION convert_loc_to_char(xcoord,ycoord) RESULT(charloc)
REAL, INTENT(IN) :: xcoord, ycoord
CHARACTER (LEN=21) :: charloc
```

File

grid_routines.f90

Type
Module function

Purpose
Returns a geographical location in string format (degrees, minutes, seconds).

Arguments

xcoord	X-coordinate of the geographical location [fractional degrees longitude]
ycoord	Y-coordinate of the geographical location [fractional degrees latitude]
charloc	Returned string format in degrees, minutes, seconds

distance_minloc

```
SUBROUTINE distance_minloc(x,y,xcoord,ycoord,nx,ny,iunit,distmin,locmin,mask)
INTEGER, INTENT(IN) :: iunit, nx, ny
INTEGER, INTENT(OUT), OPTIONAL, DIMENSION(2) :: locmin
REAL, INTENT(IN) :: x, y
LOGICAL, INTENT(IN), OPTIONAL, DIMENSION(nx,ny) :: mask
REAL, INTENT(IN), DIMENSION(nx,ny)  :: xcoord, ycoord
REAL, INTENT(OUT), OPTIONAL :: distmin
```

File
grid_routines.f90

Type
Module function

Purpose
Returns the index positions of the location on a horizontal grid which is closest to a given location.

Arguments

x	X-coordinate of the given location [m or fractional degrees longitude]
y	Y-coordinate of the given location [m or fractional degrees latitude]
xcoord	X-coordinates of the grid
ycoord	Y-coordinates of the grid

nx	Size of the grid in the X-direction
ny	Size of the grid in the Y-direction
iunit	Units of xcoord, ycoord in case of spherical coordinates 1: radians 2: degrees
distmin	If present, the returned minimum distance
locmin	If present, the location of the nearest distance
mask	Mask to exclude dry or invalid points if present and .TRUE.

distance_pts

```
FUNCTION distance_pts(x1,x2,y1,y2,inunit,outunit) RESULT(dist)
INTEGER, INTENT(IN) :: inunit, outunit
REAL, INTENT(IN) :: x1, x2, y1, y2
REAL :: dist
```

File

grid_routines.f90

Type

Module function

Purpose

Returns the distance between two given locations.

Arguments

x1	X-coordinate of the first location
x2	X-coordinate of the second location
y1	Y-coordinate of the first location
y2	Y-coordinate of the second location
inunit	Units of the locations in case of spherical coordinates 1: radians 2: degrees
outunit	Units of the result 1: radians 2: degrees 3: meters (always the case in case of a Cartesian grid)

find_obc_index_glb

```
FUNCTION find_obc_index_glb(i,j,cnode,ierr)
CHARACTER (LEN=lennode), INTENT(IN) :: cnode
INTEGER, INTENT(IN) :: i, j
INTEGER, INTENT(OUT), OPTIONAL :: ierr
INTEGER :: find_obc_index_glb
```

File
: *grid_routines.f90*

Type
: Module function

Purpose
: Returns the index of an open boundary point given its grid index location.

Arguments

i	X-index of the location
j	Y-index of the location
cnode	Type of open boundary node ('U', 'V', 'X', 'Y')
ierr	Error code in case no match has been found. If zero, no error occurred.

global_mask

```
SUBROUTINE global_mask(maskglb,cnode,wetonly)
CHARACTER (LEN=lennode), INTENT(IN) :: cnode
LOGICAL, INTENT(IN) :: wetonly
LOGICAL, INTENT(OUT), DIMENSION(nc,nr) :: maskglb
```

File
: *grid_routines.f90*

Type
: Module subroutine

Purpose
: Combine a local grid mask to a global one (used for parallel applications).

Arguments

maskglb	Returned global mask
cnode	Grid node to which the mask should apply ('C', 'U', 'V', 'X', 'Y')
wetonly	Considers open boundary points as land points if set to .TRUE..

grid_rotation_params

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

File

grid_routines.f90

Type

Module subroutine

Purpose

Determine parameters for making coordinate transforms between a rotated and the reference grid.

Arguments

iddesc	Grid file id

grid_rotation_transf

```
SUBROUTINE grid_rotation_transf(xcoord,ycoord,idir,iddesc)
INTEGER, INTENT(IN) :: iddesc, idir
REAL, INTENT(INOUT), DIMENSION(:,:) :: xcoord, ycoord
```

File

grid_routines.f90

Type

Module subroutine

Purpose

Performs coordinate transforms between a rotated and the reference grid or vice versa.

Arguments

xcoord	X-coordinates on old coordinate system on input, on new coordinate system on output
ycoord	Y-coordinates on old coordinate system on input, on new coordinate system on output
idir	Type of transformation
	1 from reference to rotated coordinates
	-1 from rotated to reference coordinates
iddesc	Grid file id

grid_spacings_curv

```
SUBROUTINE grid_spacings_curv(dist,x1,x2,y1,y2,inunit,outunit,hdel,cdir)
INTEGER, INTENT(IN) :: inunit, outunit
REAL, INTENT(IN), DIMENSION(:,:)  :: x1, x2, y1, y2
REAL, INTENT(OUT), DIMENSION(:,:) :: dist
```

File

grid_routines.f90

Type

Module subroutine

Purpose

Calculate the (horizontal) grid spacings on a given (Cartesian or spherical) curvilinear grid.

Arguments

dist	Returned grid spacings
x1	X-coordinates of the start locations
x2	X-coordinates of the end locations
y1	Y-coordinates of the start locations
y2	Y-coordinates of the end locations
inunit	Units of the grid locations in case of a spherical grid
	1: radians
	2: fractional degrees
outunit	Unit of the returned grid spacings in case of a spherical grid
	1: radians

2: fractional degrees

3: meters

grid_spacings_rectang

```
SUBROUTINE grid_spacings_rectang(dist,hdel,ycoord,inunit,outunit,cdir)
CHARACTER (LEN=1), INTENT(IN) :: cdir
INTEGER, INTENT(IN) :: inunit, outunit
REAL, INTENT(IN), DIMENSION(:) :: hdel, ycoord
REAL, INTENT(OUT), DIMENSION(:,:) :: dist
```

File

grid_routines.f90

Type

Module subroutine

Purpose

Calculate the (horizontal) grid spacings on a given (Cartesian or spherical) rectangular grid.

Arguments

dist	Returned grid spacings
hdel	Vector of grid spacings (uniform in case of a uniform rectangular grid
ycoord	Vector of Y-coordinates (latitude) in case of a spherical grid
inunit	Units of the grid locations in case of a spherical grid 1: radians 2: fractional degrees
outunit	Unit of the returned grid spacings in case of a spherical grid 1: radians 2: fractional degrees 3: meters
cdir	Coordinate direction with respect to which the spacings are taken 'X': X-direction 'Y': Y-direction

local_proc

```
FUNCTION local_proc(i,j,iproc)
INTEGER, INTENT(IN) :: i, j
INTEGER, INTENT(IN), OPTIONAL :: iproc
LOGICAL :: local_proc
```

File

grid_routines.f90

Type

Module function

Purpose

Returns .TRUE. if the point with local grid indices (i,j) belongs to the domain of the local process calling the function. If iproc is present, (i,j) are global indices and the function returns .TRUE. if the points belongs to the domain with process number iproc.

Arguments

i	X-index (local or global) of the grid point
j	Y-index (local or global) of the grid point
iproc	Process number in case (i,j) represent global indices. Must be present if i and j are global indices.

mask_array

```
FUNCTION mask_array(idim,jdim,lims,node)
CHARACTER (LEN=1), INTENT(IN) :: node
INTEGER, INTENT(IN) :: idim, jdim
INTEGER, INTENT(IN), DIMENSION(3,2) :: lims
LOGICAL, DIMENSION(idim,jdim) :: mask_array
```

File

grid_routines.f90

Type

Module function

Purpose

Define a local mask array on a sub-section of the model grid.

Arguments

idim	X-dimension of the mask array
jdim	Y-dimension of the mask array
lims	Vector selecting the sub-grid indices in the X- and Y-direction (start/end/step)
node	Grid node to which the mask applies ('C', 'U', 'V', 'W')

num_proc

```
FUNCTION num_proc(i,j)
INTEGER, INTENT(IN) :: i, j
INTEGER :: num_proc
```

File

grid_routines.f90

Type

Module function

Purpose

Returns the number of the local process domain containing the point with global indices (i,j).

Arguments

i	X-index of the location
j	Y-index of the location

rotate_vec

```
SUBROUTINE rotate_vec(vxin,vyin,vxout,vyout,angle,idir)
INTEGER, INTENT(IN) :: idir
REAL, INTENT(IN) [, DIMENSION(:,:)] ::  angle
REAL, INTENT(IN) [, DIMENSION(:[,:[,:[,:]]])] ::  vxin, vyin
REAL, INTENT(OUT) [, DIMENSION(:[,:[,:[,:]]])] ::  vxout, vyout
```

File

grid_routines.f90

Type

Generic module subroutine

Purpose

Rotate a scalar or array (upto rank 4) vector over an angle equal to *angle* if idir=1 or *-angle* if idir=-1

Arguments

vxin	X-component of input vector scalar or array
vyin	Y-component of input vector scalar or array
vxout	X-component of output vector scalar or array
vyout	Y-component of output vector scalar or array
idir	Rotation direction
angle	Rotation angle

Non-generic versions

rotate_vec_0d	Vector with scalar components
rotate_vec_1d	Vector with vector components
rotate_vec_2d	Vector with 2-D array components
rotate_vec_3d	Vector with 3-D array components
rotate_vec_4d	Vector with 4-D array components

Zcoord_arr

```
SUBROUTINE Zcoord_arr(zcoord,lbounds,ubounds,cnode,&
                    & meanlevel,outflag)
LOGICAL, INTENT(IN) :: meanlevel
CHARACTER (LEN=lennode), INTENT(IN) :: cnode
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(3) :: lbounds, ubounds
REAL, INTENT(OUT), DIMENSION(lbounds(1):ubounds(1),&
                           & lbounds(2):ubounds(2),&
                           & lbounds(3):ubounds(3)) :: zcoord
```

File

grid_routines.f90

Type

Module subroutine

Purpose

Return the z-coordinates on a section of the model grid.

Arguments

zcoord	Returned z-coordinates
lbounds	Lower bounds of the grid section
ubounds	Upper bounds of the grid section
cnode	Type of grid node ('C', 'U', 'V', 'W', 'UW', 'VW')
meanlevel	The z-coordinates are taken with respect to the mean or total water level if set to .TRUE. or .FALSE.
outflag	Output flag for dry points. If not present, a zero value is taken.

Zcoord_var

```
FUNCTION Zcoord_var(i,j,k,cnode,meanlevel)
LOGICAL, INTENT(IN) :: meanlevel
CHARACTER (LEN=lennode), INTENT(IN) :: cnode
INTEGER, INTENT(IN) :: i, j, k
REAL :: Zcoord_var
```

File

grid_routines.f90

Type

Module function

Purpose

Returns the z-coordinate of a model grid point.

Arguments

i	X-index of the grid point
j	Y-index of the grid point
k	Vertical index of the grid point
cnode	Type of grid node ('C', 'U', 'V', 'W', 'UW', 'VW')
meanlevel	The z-coordinate is taken with respect to the mean or total water level if set to .TRUE. or .FALSE.

31.11 Routines combining communication and I/O operations

The following combined operations are implemented

- Construction of a global array from local arrays by a combine communication. The global array is written to an output file by the master process.
- A global array is read by the master process and copied to all other processes.
- A global array is read and distributed to all local sub-domains.

combine_write_mod

```
SUBROUTINE combine_write_mod(values,filepars,varid,lbounds,&
                           & ubounds,rfill,varatts,vecids,nowet,&
                           & comm,commtype)
INTEGER, INTENT(IN) :: varid
INTEGER, INTENT(IN), OPTIONAL :: comm, commtype, nowet
REAL, INTENT(IN) :: rfill
TYPE(FileParams), INTENT(INOUT) :: filepars
INTEGER, INTENT(IN), DIMENSION(:) :: lbounds, ubounds
INTEGER, INTENT(IN), OPTIONAL, DIMENSION(:) :: vecids
REAL, INTENT(IN), DIMENSION(lbounds(1):,lbounds(2):[,lbounds(3):&
                       & [,lbounds(4):)]]) ::  values
TYPE (VariableAtts), INTENT(IN), OPTIONAL, DIMENSION(:) :: varatts
```

File

inout_paral.f90

Type

Generic module subroutine

Purpose

Combine local model grid arrays with real data into a global array and write the resulting array to a file in standard COHERENS format. The array has a rank between 2 and 4.

Arguments

values	Local model grid array
filepars	Derived type variable containing the attributes of the output file
varid	If positive, the id of the only variable written to the output file (between 1 and the number of variables in the file). If zero, the last dimension of *values* is considered as a variable dimension.

lbounds	Lower array bounds. The size must match the rank of the array.
ubounds	Upper array bounds. The size must match the rank of the array.
rfill	Fill value substituted at places in the global array where no values are obtained from the combine operation.
varatts	If present, the attributes of the data variable(s) (used only for log info, error checking and a header line in the output file)
vecids	If present, list of the variable ids in the output file (i.e. numbers between 1 and the number of variables in the file) and the size of the vector must match the last dimension of *values*. If not present, the last dimension is considered as a variable dimension and the ids are defined as 1, 2, up to the size of the last dimension. In case *values* is a 2-D array or varid is defined with a positive value, the argument is not allowed since only one data variable is present.
nowet	If present and positive, a land mask is applied. The number present then equals the number of dry points.
comm	MPI communicator. If not present, its value is comm_work.
commtype	Communication type for the combine operation (between 1 and 4). If not present, its value is set to iopt_MPI_comm_gath. 1: blocking, standard send 2: blocking, synchronous send 3: non-blocking, standard send 4: non-blocking, synchronous send

Non-generic versions

combine_write_mod_real_2d	Combine/write a 2-D real model grid array
combine_write_mod_real_3d	Combine/write a 3-D real model grid array
combine_write_mod_real_4d	Combine/write a 4-D real model grid array

combine_write_stats_glb

```
SUBROUTINE combine_write_stats_glb(realglb,filepars,varid,maxstats,&
                                   & nostatprocs,lstatprocs,varatts,&
                                   & vecids,comm,commtype)
```

```
INTEGER, INTENT(IN) :: maxstats, varid
INTEGER, INTENT(IN), OPTIONAL :: comm, commtype
TYPE(FileParams), INTENT(INOUT) :: filepars
INTEGER, INTENT(IN), DIMENSION(nprocs) :: nostatprocs
INTEGER, INTENT(IN), DIMENSION(maxstats,nprocs) :: lstatprocs
INTEGER, INTENT(IN), OPTIONAL, DIMENSION(:) :: vecids
REAL, INTENT(INOUT), DIMENSION(:[,:[,:[,:]]])] ::  realglb
TYPE (VariableAtts), INTENT(IN), OPTIONAL, DIMENSION(:) :: varatts
```

File

inout_paral.f90

Type

Generic module subroutine

Purpose

Perform a combine operation on a globally defined real array which is defined globally, but whose values are distributed among different processes and write the array to a file in standard COHERENS format. The array has a rank between 1 and 4. Array sections with the same first index belong to the same domain.

Arguments

realglb	Global array
filepars	Derived type variable containing the attributes of the output file
varid	If positive, variable id of the array within the output file.
maxstats	First dimension of array lstatprocs
nostatprocs	Number of "stations" in each local array
lstatprocs	Indices of local stations in the global array
varatts	If present, the attributes of the data variable(s) (used only for log info, error checking and a header line in the output file)
vecids	Arrays of variable ids if varid = 0. The argument is not present for vector arrays.
comm	MPI communicator. If not present, its value is comm_work.
commtype	Communication type for the combine operation (between 1 and 4). If not present, its value is set to iopt_MPI_comm_gath. 1: blocking, standard send

2: blocking, synchronous send

3: non-blocking, standard send

4: non-blocking, synchronous send

Non-generic versions

combine_write_stats_glb_real_1d Combine/write a 1-D array

combine_write_stats_glb_real_2d Combine/write a 2-D array

combine_write_stats_glb_real_3d Combine/write a 3-D array

combine_write_stats_glb_real_4d Combine/write a 4-D array

combine_write_stats_loc

```
SUBROUTINE combine_write_stats_loc(realloc,filepars,varid,maxstats,&
                                & nostatsglb,nostatprocs,lstatprocs,&
                                & reduced,varatts,vecids,comm,commtype)
LOGICAL, INTENT(IN), OPTIONAL :: reduced
INTEGER, INTENT(IN) :: maxstats, nostatsglb, varid
INTEGER, INTENT(IN), OPTIONAL :: comm, commtype
TYPE(FileParams), INTENT(INOUT) :: filepars
INTEGER, INTENT(IN), DIMENSION(nprocs) :: nostatprocs
INTEGER, INTENT(IN), DIMENSION(maxstats,nprocs) :: lstatprocs
INTEGER, INTENT(IN), OPTIONAL, DIMENSION(:) :: vecids
REAL, INTENT(IN), DIMENSION(:[,:[,:]]) ::  realloc
TYPE (VariableAtts), INTENT(IN), OPTIONAL, DIMENSION(:) :: varatts
```

File

inout_paral.f90

Type

Generic module subroutine

Purpose

Combine local real data at a number of stations on the model grid onto a global array and write the resulting array to a file in standard COHERENS format. The array has a rank between 1 and 4.

Arguments

realloc	Array with the local station data
filepars	Derived type variable containing the attributes of the output file

varid	If positive, the id of the only variable written to the output file (between 1 and the number of variables in the file). If zero, the last dimension of *realloc* is considered as a variable dimension.
maxstats	Second dimension of the array lstatprocs
nostatsglb	Total number (over all sub-domains) of stations within the global array
nostatprocs	Vector array with the number stations on each sub-domain
lstatprocs	Index mapping array of the local station indices to the ones in the global array
reduced	If .TRUE. or not present, the last array dimension is taken within the first one so that the rank of the output array is reduced by 1.
varatts	If present, the attributes of the data variable(s) (used only for log info, error checking and a header line in the output file)
vecids	If present, list of the variable ids in the output file (i.e. numbers between 1 and the number of variables in the file) and the size of the vector must match the last dimension of *realloc*. If not present, the last dimension is considered as a variable dimension and the ids are defined as 1, 2, up to the size of the last dimension. In case *realloc* is a vector or varid is defined with a positive value, the argument is not allowed since only one data variable is present. The argument is not present for vector arrays.
comm	MPI communicator. If not present, its value is comm_work.
commtype	Communication type for the combine operation (between 1 and 4). If not present, its value is set to iopt_MPI_comm_gath. 1: blocking, standard send 2: blocking, synchronous send 3: non-blocking, standard send 4: non-blocking, synchronous send

Non-generic versions

combine_write_stats_loc_real_1d	Combine/write a vector of station data
combine_write_stats_loc_real_2d	Combine/write a 2-D array of station data

combine_write_stats_loc_real_3d	Combine/write a 3-D array of station data
combine_write_stats_loc_real_4d	Combine/write a 3-D array of station data

combine_write_submod

```
SUBROUTINE combine_write_submod(realloc,filepars,varid,ndimsglb,&
                              & limprocs,rfill,varatts,vecids,&
                              & nowet,comm,commtype)
INTEGER, INTENT(IN) :: varid
INTEGER, INTENT(IN), OPTIONAL :: comm, commtype, nowet
REAL, INTENT(IN) :: rfill
TYPE(FileParams), INTENT(INOUT) :: filepars
INTEGER, INTENT(IN), DIMENSION(:) :: ndimsglb
INTEGER, INTENT(IN), OPTIONAL, DIMENSION(:) :: vecids
INTEGER, INTENT(IN), DIMENSION(2,2,nprocs) :: limprocs
REAL, INTENT(IN), DIMENSION(:,:[,:[,:)]]) ::  realloc
TYPE (VariableAtts), INTENT(IN), OPTIONAL, DIMENSION(:) :: varatts
```

File

inout_paral.f90

Type

Generic module subroutine

Purpose

Combine local real model sub-grid arrays into a global (sub-grid) array and write the resulting array to a file in standard COHERENS format. The array has a rank between 2 and 4.

Arguments

realloc	Local model sub-grid array
filepars	Derived type variable containing the attributes of the output file
varid	If positive, the id of the only variable written to the output file (between 1 and the number of variables in the file). If zero, the last dimension of *realloc* is considered as a variable dimension.
ndimsglb	Shape of the global array. Size of the vector must match the rank of *realloc*.

limprocs	Location of a local array section within the global array by means of lower and upper index bounds.
rfill	Fill value substituted at places in the global array where no values are obtained from the combine operation.
varatts	If present, the attributes of the data variable(s) (used only for log info, error checking and a header line in the output file)
vecids	If present, list of the variable ids in the output file (i.e. numbers between 1 and the number of variables in the file) and the size of the vector must match the last dimension of *realloc*. If not present, the last dimension is considered as a variable dimension and the ids are defined as 1, 2, up to the size of the last dimension. In case *realloc* is a 2-D array or varid is defined with a positive value, the argument is not allowed since only one data variable is present.
nowet	If present and positive, a land mask is applied. The number then equals the number of dry points.
comm	MPI communicator. If not present, its value is comm_work.
commtype	Communication type for the combine operation (between 1 and 4). If not present, its value is set to iopt_MPI_comm_gath. 1: blocking, standard send 2: blocking, synchronous send 3: non-blocking, standard send 4: non-blocking, synchronous send

Non-generic versions

combine_write_submod_real_2d	Combine/write a 2-D real sub-grid array
combine_write_submod_real_3d	Combine/write a 3-D real sub-grid array
combine_write_submod_real_4d	Combine/write a 4-D real sub-grid array

read_distribute_mod

```
SUBROUTINE read_distribute_mod(realloc,filepars,varid,lbounds,ubounds,&
                            & nhdist,fdist,land_mask,rfill,varatts,&
                            & vecids,maskvals,shared,comm,commtype)
LOGICAL, INTENT(IN) :: fdist, land_mask
LOGICAL, INTENT(IN), OPTIONAL :: shared
INTEGER, INTENT(IN) :: varid
```

```
INTEGER, INTENT(IN), OPTIONAL :: comm, commtype
REAL, INTENT(IN) :: rfill
TYPE(FileParams), INTENT(IN) :: filepars
INTEGER, INTENT(IN), DIMENSION(:) :: lbounds, ubounds
LOGICAL, INTENT(IN), OPTIONAL, DIMENSION(lbounds(1):,&
                                      & lbounds(2):) :: maskvals
INTEGER, INTENT(IN), DIMENSION(4) :: nhdist
INTEGER, INTENT(IN), OPTIONAL, DIMENSION(:) :: vecids
REAL, INTENT(INOUT), DIMENSION(lbounds(1):,lbounds(2):[,lbounds(3):&
                           & [,lbounds(4):)]]) ::  realloc
TYPE (VariableAtts), INTENT(IN), OPTIONAL, DIMENSION(:) :: varatts
```

File

inout_paral.f90

Type

Generic module subroutine

Purpose

Read and copy an integer or real scalar or array (upto rank 4) from a file in standard COHERENS format and copy to all other processes.

Arguments

realloc	Returned array
filepars	Derived type variable containing the attributes of the input file
varid	If positive, the id of the only variable read from the input file (between 1 and the number of variables in the file). If zero, the last dimension of *realloc* is considered as a variable dimension.
lbounds	Lower array bounds. The size must match the rank of the array.
ubounds	Upper array bounds. The size must match the rank of the array.
nhdist	Halo sizes used for the distribute operation (WESN directions)
fdist	Data are distributed only if this argument is set to .TRUE.
land_mask	A land mask is applied if set to .TRUE.
rfill	Fill value substituted at places in the local array where no values can be obtained from the distribute operation.

varatts	If present, the attributes of the data variable(s) (used only for log info and error checking)
vecids	If present, list of the variable ids in the input file (i.e. numbers between 1 and the number of variables in the file) and the size of the vector must match the last dimension of *realloc*. If not present, the last dimension is considered as a variable dimension and the ids are defined as 1, 2, up to the size of the last dimension. In case *realloc* is a 2-D array or varid is defined with a positive value, the argument is not allowed since only one data variable is present.
maskvals	Array of mask values used in case a land mask is applied
shared	The global array is known to (read by) all processes if .TRUE., to the master process only if .FALSE.. If not present, its value is set to shared_read.
comm	MPI communicator. If not present, its value is comm_work.
commtype	Communication type for the distribute operation (between 1 and 4). If not present, its value is set to iopt_MPI_comm_scat. 1: blocking, standard send 2: blocking, synchronous send 3: non-blocking, standard send 4: non-blocking, synchronous send

Non-generic versions

read_distribute_mod_real_2d	Read/distribute a real 2-D array
read_distribute_mod_real_3d	Read/distribute a real 3-D array
read_distribute_mod_real_4d	Read/distribute a real 4-D array

31.12 Input/output operations

Ensemble of routines for model input and output.

- open_file, open_filepars: open a file connection
- close_file, close_filepars: close a file connection
- read_vars: read data from a file in standard COHERENS format
- write_vars: write data to a file in standard COHERENS format

- read_glbatts_mod: read the global attributes from a forcing file in standard COHERENS format
- read_varatts_mod: read the variable attributes from a forcing file in standard COHERENS format
- read_glbatts_out: read the global attributes from a user output file in standard COHERENS format
- read_varatts_out: read the variable attributes from a user output file in standard COHERENS format
- write_atts_mod: write the global and variable attributes to a forcing file in standard COHERENS format
- inout_atts_out: write the global and variable attributes to a user output file in standard COHERENS format

close_file

```
SUBROUTINE close_file(iounit,ioform,filename,fildel)
LOGICAL, INTENT(IN), OPTIONAL :: fildel
CHARACTER (LEN=1), INTENT(IN) :: ioform
CHARACTER (LEN=*), INTENT(IN), OPTIONAL :: filename
INTEGER, INTENT(INOUT) :: iounit
```

File
: *inout_routines.f90*

Type
: Module subroutine

Purpose
: Close a file connection.

Arguments

iounit	File unit number
ioform	File format
	'A' ASCII
	'U' unformatted binary
	'D' direct access binary (currently not used in the program)

	'N' netCDF
filename	File name
fildel	If present and .TRUE., the file is deleted after closing.

close_filepars

```
SUBROUTINE close_filepars(filepars,fildel)
LOGICAL, INTENT(IN), OPTIONAL :: fildel
TYPE(FileParams), INTENT(INOUT) :: filepars
```

File
inout_routines.f90

Type
Module subroutine

Purpose
Closes a file connection and resets file attributes.

Arguments

filepars	File attributes
fildel	If present and .TRUE., the file is deleted after closing.

get_unit

```
FUNCTION get_unit()
INTEGER :: get_unit
```

File
inout_routines.f90

Type
Module function

Purpose
Returns the smallest available unit number not connected to a file.

inout_atts_out

```
SUBROUTINE inout_atts_out(file_type)
CHARACTER (LEN=2), INTENT(IN) :: file_type
```

File
inout_routines.f90

Type
Module subroutine

Purpose
Define and write the global, coordinate and variable attributes to a user-defined output file.

Arguments

file_type	Output type
	'TS' time series
	'TA' time averaged
	'HR' harmonic residuals
	'HA' harmonic amplitudes
	'HP' harmonic phases
	'HE' tidal ellipse parameters

monitor_files

```
SUBROUTINE monitor_files
```

File
inout_routines.f90

Type
Module subroutine

Purpose
Set some parameters for monitoring, open and close the log, error and warning files.

open_file

```
SUBROUTINE open_file(iounit,filename,iotype,ioform,lenrec)
CHARACTER (LEN=*), INTENT(IN) :: filename, ioform, iotype
INTEGER, INTENT(INOUT) :: iounit
INTEGER, INTENT(IN), OPTIONAL :: lenrec
```

File

inout_routines.f90

Type

Module subroutine

Purpose

Open a file connection.

Arguments

iounit	Returned file unit number
filename	File name
iotype	Type of file access
	'IN' for reading only
	'OUT' for writing only
	'INOUT' for reading and writing
ioform	File format
	'A' ASCII
	'U' unformatted binary
	'D' direct access binary (currently not used in the program)
	'N' netCDF
lenrec	Length of a data record in case the file is opened for direct access

open_filepars

```
SUBROUTINE open_filepars(filepars,iotype)
CHARACTER (LEN=*), INTENT(IN), OPTIONAL :: iotype
TYPE(FileParams), INTENT(INOUT) :: filepars
```

File

inout_routines.f90

Type
Module subroutine

Purpose
Open a file connection and set file attributes.

Arguments

filepars	File attributes
iotype	If present, the file access ('IN', 'OUT', 'INOUT'). Otherwise, the access is defined through the status attribute of filepars.

read_glbatts_mod

```
SUBROUTINE read_glbatts_mod(filepars)
TYPE(FileParams), INTENT(INOUT) :: filepars
```

File
inout_routines.f90

Type
Module subroutine

Purpose
Read the global attributes of a standard forcing file. In case of a sequential file, the file pointer must be located at the start of the data record.

Arguments

filepars	File attributes

read_glbatts_out

```
SUBROUTINE read_glbatts_out(filepars,gridpars,outgpars)
TYPE (FileParams), INTENT(INOUT) :: filepars
TYPE (FileParams), INTENT(INOUT) :: gridpars
TYPE (OutGridParams), INTENT(OUT) :: outgpars
```

File
inout_routines.f90

Type
Module subroutine

Purpose

Read the global attributes of a user output file. In case of a sequential file, the file pointer must be located at the start of the data record.

Arguments

filepars	Attributes of the user output file
gridpars	Attributes of the file containing the output grid (equal to filepars if outgpars%grid_file=.FALSE.)
outgpars	Attributes of the output grid

read_skip_out

```
SUBROUTINE read_skip_out(filepars)
TYPE (FileParams), INTENT(INOUT) :: filepars
```

File

inout_routines.f90

Type

Module subroutine

Purpose

Read the global and variable attributes and the coordinate data of a user output file without storing the data. In case the file is opened for sequential access, the file pointer is located at the first output time on exit. The routine should not be called in case of a netCDF file.

Arguments

filepars	Attributes of the user output file

read_station_names

```
SUBROUTINE read_station_names(filepars,station_names,nostats)
INTEGER, INTENT(IN) :: nostats
TYPE (FileParams), INTENT(INOUT) :: filepars
CHARACTER (LEN=lenname), INTENT(OUT),&
                      & DIMENSION(nostats) :: station_names
```

File

inout_routines.f90

Type
Module subroutine

Purpose
Read the names of data stations from a user output file. In case of sequential access, the file pointer must be located at the correct position.

Arguments

filepars	Attributes of the user output file
station_names	Returned station names
nostats	Number of stations

read_submod

```
SUBROUTINE read_submod(realsub,filepars,varid,&
                     & land_mask,varatts,vecids,maskvals)
LOGICAL, INTENT(IN) :: land_mask
INTEGER, INTENT(IN) :: varid
TYPE(FileParams), INTENT(IN) :: filepars
LOGICAL, INTENT(IN), OPTIONAL, DIMENSION(:,:) :: maskvals
INTEGER, INTENT(IN), OPTIONAL, DIMENSION(:) :: vecids
REAL, INTENT(INOUT), DIMENSION(:,:[,:[,:)]]) ::  realsub
TYPE (VariableAtts), INTENT(IN), OPTIONAL, DIMENSION(:) :: varatts
```

File
inout_routines.f90

Type
Generic module subroutine

Purpose
Read a subgrid section of a model array defined on the global grid from a file in COHERENS standard format. In case of a sequential file, the file pointer must be located at the appropriate position.

Arguments

realsub	Returned sub-grid array
filepars	Derived type variable containing the attributes of the input file

varid	If positive, the id of the only variable read from the input file (between 1 and the number of variables in the file). If zero, the last dimension of *realsub* is considered as a variable dimension.
land_mask	A (horizontal) land mask is applied if set to .TRUE..
varatts	If present, the attributes of the data variable(s) (used only for log info and error checking).
vecids	If present, list of the variable ids in the input file (i.e. numbers between 1 and the number of variables in the file) and the size of the vector must match the last dimension of *realsub*. If not present, the last dimension is considered as a variable dimension and the ids are defined as 1, 2, up to the size of the last dimension. In case *realsub* is a 2-D array or varid is defined with a positive value, the argument is not allowed since only one data variable is present.
maskvals	Array of mask values used in case a land mask is applied

Non-generic versions

read_submod_real_2d	Read a real 2-D sub-grid array
read_submod_real_3d	Read a real 3-D sub-grid array
read_submod_real_4d	Read a real 4-D sub-grid array

read_time

```
SUBROUTINE read_time(time,filepars,timerec)
CHARACTER (LEN=lentime) or REAL, INTENT(OUT)  ::  time
INTEGER, INTENT(IN), OPTIONAL :: timerec
TYPE(FileParams), INTENT(INOUT) :: filepars
```

File

inout_routines.f90

Type

Generic module subroutine

Purpose

Read a time record from a file in standard COHERENS format.

Arguments

time	Returned time either in the form of an absolute date/time in string format or a relative time in numerical format.

filepars Derived type variable containing the attributes of the input file

timerec If not present, the timerec global attribute is increased by one. The iostat global attribute is set to 1 or 3 for a sequential read or to 1, 2 or 3 in case of a non-sequential read.

Non-generic versions

read_time_char Read the time in string format.

read_time_real Read the time in (real) numeric format.

read_varatts_mod

```
SUBROUTINE read_varatts_mod(filepars,varatts,numvars)
INTEGER, INTENT(IN) :: numvars
TYPE (FileParams), INTENT(INOUT) :: filepars
TYPE (VariableAtts), INTENT(OUT), OPTIONAL,&
                  & DIMENSION(numvars) :: varatts
```

File

inout_routines.f90

Type

Module subroutine

Purpose

Read the variable attributes from a forcing file in standard COHERENS format. An information file is written if requested. In case of a sequential file, the file pointer must be located at the appropriate position.

Arguments

filepars Attributes of the input file

varatts If present, returned variable attributes. Otherwise the attributes are read but not stored.

numvars Number of variables

read_varatts_out

```
SUBROUTINE read_varatts_out(filepars,varatts,novars,bufid,vector)
LOGICAL, INTENT(IN) :: vector
INTEGER, INTENT(IN) :: bufid, novars
```

```
TYPE (FileParams), INTENT(INOUT) :: filepars
TYPE (VariableAtts), INTENT(OUT), DIMENSION(novars) :: varatts
```

File
inout_routines.f90

Type
Module subroutine

Purpose
Read the variable attributes from a user output file in standard COHERENS format. The routine can be called a first time to read the attributes of the coordinate variables in which case bufid must be 0 and vector is not present, and a second time to read the attributes of the data variables in which case bufid should be equal to the number of coordinate variables and vector must be present. In case of a sequential file, the file pointer must be located at the appropriate position.

Arguments

filepars	Attributes of the input file
varatts	Returned variable attributes
novars	Number of coordinate or data variables
bufid	In case of a netCDF file, the netCDF variable id of the first variable equals bufid+1. Otherwise, its value is irrelevant.
vector	Reads the vector attribute of the variables (if needed).

read_vars

```
SUBROUTINE read_vars(values,filepars,varid,varatts,vecids)
INTEGER, INTENT(IN) :: varid
TYPE (FileParams), INTENT(IN) :: filepars
INTEGER, INTENT(IN), OPTIONAL, DIMENSION(:) :: vecids
INTEGER or REAL, INTENT(OUT) [, DIMENSION(:[,:[,:[,:]]])] ::  values
TYPE (VariableAtts), INTENT(IN), OPTIONAL, DIMENSION(:) :: varatts
```

File
inout_routines.f90

Type
Generic module subroutine

Purpose

Read scalar or array data from a file in standard COHERENS format. In case of a sequential file, the file pointer must be located at the appropriate position.

Arguments

values	Returned scalar or array data
filepars	Derived type variable containing the attributes of the input file
varid	If positive, the id of the only variable read from the input file (between 1 and the number of variables in the file). If zero, the last dimension of *values* is considered as a variable dimension.
varatts	If present, the attributes of the data variable(s) (used only for log info and error checking)
vecids	If present, list of the variable ids in the input file (i.e. numbers between 1 and the number of variables in the file) and the size of the vector must match the last dimension of *values*. If not present, the last dimension is considered as a variable dimension and the ids are defined as 1, 2, up to the size of the last dimension. In case *values* is a scalar or varid is defined with a positive value, the argument is not allowed since only one data variable is present.

Non-generic versions

read_vars_int_0d	Integer scalar
read_vars_int_1d	Integer vector
read_vars_int_2d	Integer 2-D array
read_vars_int_3d	Integer 3-D array
read_vars_int_4d	Integer 4-D array
read_vars_real_0d	Real scalar
read_vars_real_1d	Real vector
read_vars_real_2d	Real 2-D array
read_vars_real_3d	Real 3-D array
read_vars_real_4d	Real 4-D array

write_atts_mod

```
SUBROUTINE write_atts_mod(filepars,varatts,numvars)
INTEGER, INTENT(IN) :: numvars
TYPE (FileParams), INTENT(INOUT) :: filepars
TYPE (VariableAtts), INTENT(INOUT), DIMENSION(numvars) :: varatts
```

File
: *inout_routines.f90*

Type
: Module subroutine

Purpose
: Write all global and variable (coordinate and data) attributes to a forcing file in standard COHERENS format.

Arguments

filepars	Attributes of the output file
varatts	Variable attributes
numvars	Number of coordinate plus data variables

write_info_mod

```
SUBROUTINE write_info_mod(filepars,varatts,numvars)
INTEGER, INTENT(IN) :: numvars
TYPE (FileParams), INTENT(IN) :: filepars
TYPE (VariableAtts), INTENT(IN), OPTIONAL,&
                  & DIMENSION(numvars) :: varatts
```

File
: *inout_routines.f90*

Type
: Module subroutine

Purpose
: Write the info file associated with a forcing file in standard COHERENS format.

Arguments

filepars	Attributes of the forcing file

varatts	Variable attributes
numvars	Number of (coordinate plus data) variables. Must be zero if varatts is not present.

write_time

```
SUBROUTINE write_time(time,filepars,timerec)
CHARACTER (LEN=lentime) or REAL, INTENT(IN)  ::  time
INTEGER, INTENT(IN), OPTIONAL :: timerec
TYPE(FileParams), INTENT(INOUT) :: filepars
```

File

inout_routines.f90

Type

Generic module subroutine

Purpose

Write a time record to a file in standard COHERENS format.

Arguments

time	Time record either in the form of an absolute date/time in string format or a relative time in numerical format.
filepars	Derived type variable containing the attributes of the input file
timerec	Time record for output to a netCDF file. If not present, the time record attribute is increased by one and used as the output time record.

Non-generic versions

write_time_char	Write the time in string format.
write_time_real	Write the time in (real) numeric format.

write_vars

```
SUBROUTINE write_vars(values,filepars,varid,varatts,vecids)
INTEGER, INTENT(IN) :: varid
TYPE (FileParams), INTENT(IN) :: filepars
INTEGER, INTENT(IN), OPTIONAL, DIMENSION(:) :: vecids
INTEGER or REAL, INTENT(IN)&
            & [, DIMENSION(:[,:[,:[,:]]])]  ::  values
TYPE (VariableAtts), INTENT(IN), OPTIONAL, DIMENSION(:) :: varatts
```

File

inout_routines.f90

Type

Generic module subroutine

Purpose

Write scalar or array data to a file in standard COHERENS format.

Arguments

values	Scalar or array data to be written.
filepars	Derived type variable containing the attributes of the output file
varid	If positive, the id of the only variable written to the output file (between 1 and the number of variables in the file). If zero, the last dimension of *values* is considered as a variable dimension.
varatts	If present, the attributes of the data variable(s) (used only for log info, error checking and a header line in the output file)
vecids	If present, list of the variable ids in the output file (i.e. numbers between 1 and the number of variables in the file) and the size of the vector must match the last dimension of *values*. If not present, the last dimension is considered as a variable dimension and the ids are defined as 1, 2, up to the size of the last dimension. In case that *values* is a scalar or varid is defined with a positive value, the argument is not allowed since only one data variable is present.

Non-generic versions

write_vars_int_0d	Integer scalar
write_vars_int_1d	Integer vector
write_vars_int_2d	Integer 2-D array
write_vars_int_3d	Integer 3-D array
write_vars_int_4d	Integer 4-D array
write_vars_real_0d	Real scalar
write_vars_real_1d	Real vector
write_vars_real_2d	Real 2-D array

write_vars_real_3d Real 3-D array

write_vars_real_4d Real 4-D array

31.13 Library of mathematical routines

brent_root

```
FUNCTION brent_root(func,varname,ipars,rpars,noint,noreal,x1,x2,&
                  & tol,fval,ierr)
CHARACTER (LEN=*), INTENT(IN) :: varname
INTEGER, INTENT(OUT) :: ierr
INTEGER, INTENT(IN) :: noint, noreal
REAL, INTENT(IN)  :: tol
REAL, INTENT(IN) :: x1, x2
REAL, INTENT(OUT) :: fval
INTEGER, INTENT(IN), DIMENSION(noint) :: ipars
REAL, INTENT(IN), DIMENSION(noreal) :: rpars
INTERFACE
   FUNCTION func(x,ipars,rpars,noint,noreal)
     INTEGER, INTENT(IN) :: noint, noreal
     INTEGER, INTENT(IN), DIMENSION(noint) :: ipars
     REAL, INTENT(IN), DIMENSION(noreal) :: rpars
     REAL,INTENT(IN) :: x
     REAL :: func
   END FUNCTION func
END INTERFACE
```

File

math_library.F90

Type

Module function

Purpose

Find the root of the function func using Brent's method. The initial guess is assumed to be in the interval [x1,x2].

Reference

Press *et al.* (1992)

Arguments

func	Input function
varname	Function name
ipars	List of integer parameters used in the function call.
rpars	List of real parameters used in the function call.
noint	Size of the vector ipars which may be zero
noreal	Size of the vector rpars which may be zero
x1	Lower bound of the initial interval
x2	Upper bound of the initial interval
tol	Required accuracy of the result
fval	Value of the function at the root.
ierr	Returned error code

0: No error occurred

1: x1 is equal to or larger than x2

2: Unable to adjust the interval [x1,x2] after maxcount=20 iterations

3: Unable to locate the root after maxiter=100 iterations

complex_polar

```
SUBROUTINE complex_polar(xreal,ximag,xamp,xpha,maskvals,outflag)
REAL, INTENT(IN), OPTIONAL :: outflag
LOGICAL, INTENT(IN), OPTIONAL, DIMENSION(:,:) :: maskvals
REAL, INTENT(IN) [, DIMENSION(:[,:])] ::  ximag, xreal
REAL, INTENT(OUT), OPTIONAL [, DIMENSION(:[,:])] ::  xamp, xpha
```

File

math_library.F90

Type

Generic module subroutine

Purpose

Returns the aplitude and phase of a complex scalar, vector or 2-D array.

Arguments

xreal	Real part of the complex scalar or array
ximag	Imaginary part of the complex scalar or array

xamp	If present, returned amplitude of the complex scalar or array
xpha	If present, returned phase of the complex scalar or array [radians]
maskvals	If present, 2-D mask array to exclude dry points. The argument is only allowed for 2-D arrays.
outflag	Output flag for invalid points, if present. Equal to real_min otherwise. The argument is only used when maskvals is present.

Non-generic versions

complex_polar_0d	Complex scalars
complex_polar_1d	Complex vector
complex_polar_2d	Complex 2-D array

complex_sqrt_arr

```
FUNCTION complex_sqrt_arr(z,ndims,mask,maskvals,outflag)
LOGICAL, INTENT(IN) :: mask
REAL, INTENT(IN), OPTIONAL :: outflag
INTEGER, INTENT(IN), DIMENSION(4) :: ndims
LOGICAL, INTENT(IN), OPTIONAL,&
                   & DIMENSION(ndims(1),ndims(2)) :: maskvals
COMPLEX, INTENT(IN), DIMENSION(ndims(1),ndims(2),&
                             & ndims(3),ndims(4)) :: z
COMPLEX, DIMENSION(ndims(1),ndims(2),&
                 & ndims(3),ndims(4)) :: complex_sqrt_arr
```

File
math_library.F90

Type
Module function

Purpose
Returns the square root of the elements of a complex (4-D) array.

Reference
Press *et al.* (1992)

Arguments

z	Complex input array
ndims	Shape of the complex array
mask	A mask is applied on the first two dimensions to exclude invalid (land) points if set to .TRUE.
maskvals	Values of the land mask. Must be present if mask = .TRUE.
outflag	Flag for land values. If not present, the flag is set to real_min

complex_sqrt_var

```
FUNCTION complex_sqrt_var(z)
COMPLEX, INTENT(IN) :: z
COMPLEX :: complex_sqrt_var
```

File
math_library.F90

Type
Module function

Purpose
Returns the square root of a complex scalar.

Reference
Press *et al.* (1992)

Arguments

z	Complex input scalar

gauss_quad

```
SUBROUTINE gauss_quad(nrpoints,weights,locations)
INTEGER, INTENT(IN) :: nrpoints
REAL (KIND=kndlong), INTENT(OUT), DIMENSION(nrpoints) :: locations, weights
```

File
math_library.F90

Type
Module subroutine

Purpose

Returns the locations and weights for numerical integration using the Gauss-Legendre quadrature method.

Arguments

nrpoints	Number of nodes used for applying Gauss-Lengendre integration
weights	Returned weight factors
m	Degree of the polynomial
ierr	Returned locations (between 0 and 1)

poly_all_roots

```
SUBROUTINE poly_all_roots(a,x,m,ierr)
INTEGER, INTENT(IN) :: m
INTEGER, INTENT(OUT) :: ierr
COMPLEX (KIND=kndlong), INTENT(OUT), DIMENSION(m)  :: x
REAL (KIND=kndlong), INTENT(IN), DIMENSION(m+1) :: a
```

File

math_library.F90

Type

Module subroutine

Purpose

Find all roots of a polynomial using Laguerre's method.

Reference

Press *et al.* (1992)

Arguments

a	Coefficients of the polynomial where a(k+1) represents the coefficient of the k^{th} power
x	Returned (complex) roots
m	Degree of the polynomial
ierr	Returned error code
	0: No error occurred
	1: No solution obtained after maxit=80 iterations

poly_div

```
SUBROUTINE poly_div(u,nu,v,nv,r)
INTEGER, INTENT(IN) :: nu, nv
REAL (KIND=kndlong), INTENT(OUT), DIMENSION(nu) :: r
REAL (KIND=kndlong), INTENT(INOUT), DIMENSION(nu) :: u
REAL (KIND=kndlong), INTENT(IN), DIMENSION(nv) :: v
```

File
math_library.F90

Type
Module subroutine

Purpose
Divide two polynomials

Reference
Press *et al.* (1992)

Arguments

u	Coefficients of the polynomial in the nominator where u(k+1) represents the coefficient of the k^{th} power
nu	Degree of the polynomial u plus one
v	Coefficients of the polynomial in the denominator where v(k+1) represents the coefficient of the k^{th} power
nv	Degree of the polynomial v plus one
r	Returned polynomial obtained after dividing u by v

poly_root

```
SUBROUTINE poly_root(a,x,m,ierr)
INTEGER, INTENT(IN) :: m
INTEGER, INTENT(OUT) :: ierr
COMPLEX, INTENT(INOUT) :: x
REAL, INTENT(IN), DIMENSION(m+1) :: a
```

File
math_library.F90

Type
Module subroutine

Purpose

Find the root of a polynomial with initial guess x using Laguerre's method.

Reference

Press *et al.* (1992)

Arguments

a	Coefficients of the polynomial where a(k+1) represents the coefficient of the k^{th} power
x	Initial guess on input, root on output.
m	Degree of the polynomial
ierr	Returned error code
	0: No error occurred
	1: No solution obtained after maxit=80 iterations

quadratic_root_arr

```
SUBROUTINE quadratic_root_arr(coef,root1,root2,ndims,outflag,&
                            & mask,maskvals)
LOGICAL, INTENT(IN) :: mask
REAL, INTENT(IN) :: outflag
INTEGER, INTENT(IN), DIMENSION(4) :: ndims
LOGICAL, INTENT(IN), OPTIONAL,&
                   & DIMENSION(ndims(1),ndims(2)) :: maskvals
REAL, INTENT(IN), DIMENSION(ndims(1),ndims(2),&
                            & ndims(3),ndims(4),3) :: coef
COMPLEX, INTENT(OUT), DIMENSION(ndims(1),ndims(2),&
                               & ndims(3),ndims(4)) :: root1, root2
```

File

math_library.F90

Type

Module subroutine

Purpose

Returns the roots of a series second degree equations.

Reference

Press *et al.* (1992)

Arguments

coef	Coefficients of the quadratic equation. The coefficients of the zero, first and second power are given by respectively the first, second and third index of the last dimension.
root1	First root
root2	Second root
ndims	Shape of the first four dimensions of the coefficient arrays
outflag	Flag for invalid (land) values.
mask	A mask is applied on the first two dimensions to exclude invalid (land) points if set to .TRUE.
maskvals	Values of the land mask. Must be present if mask = .TRUE.

quadratic_root_var

```
SUBROUTINE quadratic_root_var(coef,root1,root2,outflag)
REAL, INTENT(IN), DIMENSION(3) :: coef
REAL, INTENT(IN) :: outflag
COMPLEX, INTENT(OUT) :: root1, root2
```

File

math_library.F90

Type

Module subroutine

Purpose

Returns the roots of a second degree equation.

Reference

Press *et al.* (1992)

Arguments

coef	The coefficients of the zero, first and second power in the quadratic equation are given by respectively the first, second and third index.
root1	First root
root2	Second root
outflag	In the case that the solutions are complex, the roots are set to this value.

vector_mag_arr_atc

```
SUBROUTINE vector_mag_arr_atc(xcomp,ycomp,intsrce,intdest,nzdim,&
                            & nosize,ivarid,info,vecmag,vecpha,outflag)
LOGICAL, INTENT(IN) :: info
INTEGER, INTENT(IN) :: intdest, intsrce, ivarid, nosize, nzdim
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(ncloc+1,nrloc,nzdim,nosize) :: xcomp
REAL, INTENT(IN), DIMENSION(ncloc,nrloc+1,nzdim,nosize) :: ycomp
REAL, INTENT(OUT), OPTIONAL, DIMENSION(ncloc,nrloc,nzdim,nosize) :: vecmag, &
                                                                  & vecpha
```

File

math_library.F90

Type

Module subroutine

Purpose

Calculate the magnitude and phase of a vector, defined at the velocity nodes, at the C-nodes

Arguments

xcomp	Vector component in the X-direction at the U-nodes
ycomp	Vector component in the Y-direction at the V-nodes
intsrce	Selects valid points at the velocity nodes for interpolation from U/V- to C-nodes. 0: all points 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
intdest	Selects valid points at the C-nodes for interpolation from U/V- to C-nodes. 0: all points 1: wet points only
nzdim	Vertical dimension of input vectors

nosize	Variable (last) dimension of the input vector, equal to 1 for a single variable vector
ivarid	Variabler key id of the vector variable
info	Disables/enables writing of a log info.
vecmag	Returned vector magnitude (optional)
vecpha	Returned vector phase (optional) [radians]
outflag	Output flag for invalid points, if present. Zero otherwise.

vector_mag_arr_atu

```
SUBROUTINE vector_mag_arr_atu(xcomp,ycomp,intsrce,intdest,nzdim,&
                            & nosize,ivarid,info,vecmag,vecpha,outflag,hregula
LOGICAL, INTENT(IN) :: info
LOGICAL, INTENT(IN), OPTIONAL :: hregular
INTEGER, INTENT(IN) :: intdest, intsrce, ivarid, nosize, nzdim
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nzdim,nosize) :: xcomp
REAL, INTENT(IN), DIMENSION(0:ncloc,nrloc+1,nzdim,nosize) :: ycomp
REAL, INTENT(OUT), OPTIONAL, DIMENSION(ncloc,nrloc,nzdim,nosize) :: vecmag, &
                                                                  & vecpha
```

File

math_library.F90

Type

Module subroutine

Purpose

Calculate the magnitude and phase of a vector, defined at the velocity nodes, at the U-nodes

Arguments

xcomp	Vector component in the X-direction at the U-nodes
ycomp	Vector component in the Y-direction at the V-nodes
intsrce	Selects valid points at the V-nodes for interpolation to the U-node 0: all points 1: coastal boundaries, interior points and open boundaries only

2: interior points only

3: interior points and open boundaries only

4: coastal boundaries and interior points only

intdest	Selects valid points at the U-nodes for interpolation from the V-node 0: all points 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
nzdim	Vertical dimension of input vectors
nosize	Variable (last) dimension of the input vector, equal to 1 for a single variable vector
ivarid	Variabler key id of the vector variable
info	Disables/enables writing of a log info.
vecmag	Returned vector magnitude (optional)
vecpha	Returned vector phase (optional) [radians]
outflag	Output flag for invalid points, if present. Zero otherwise.
hregular	Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.

vector_mag_arr_atv

```
SUBROUTINE vector_mag_arr_atv(xcomp,ycomp,intsrce,intdest,nzdim,&
                           & nosize,ivarid,info,vecmag,vecpha,outflag,hregular)
LOGICAL, INTENT(IN) :: info
LOGICAL, INTENT(IN), OPTIONAL :: hregular
INTEGER, INTENT(IN) :: intdest, intsrce, ivarid, nosize, nzdim
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(ncloc+1,0:nrloc,nzdim,nosize) :: xcomp
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nzdim,nosize) :: ycomp
REAL, INTENT(OUT), OPTIONAL, DIMENSION(ncloc,nrloc,nzdim,nosize) :: vecmag, &
                                                                  & vecpha
```

File

math_library.F90

Type

Module subroutine

Purpose

Calculate the magnitude and phase of a vector, defined at the velocity nodes, at the V-nodes

Arguments

xcomp	Vector component in the X-direction at the U-nodes
ycomp	Vector component in the Y-direction at the V-nodes
intsrce	Selects valid points at the U-nodes for interpolation to the V-node 0: all points 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
intdest	Selects valid points at the V-nodes for interpolation from to the U-node 0: all points 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
nzdim	Vertical dimension of input vectors
nosize	Variable (last) dimension of the input vector, equal to 1 for a single variable vector
ivarid	Variabler key id of the vector variable
info	Disables/enables writing of a log info.
vecmag	Returned vector magnitude (optional)
vecpha	Returned vector phase (optional) [radians]

outflag — Output flag for invalid points, if present. Zero otherwise.

hregular — Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.

vector_mag_var_atc

```
SUBROUTINE vector_mag_var_atc(xcomp,ycomp,i,j,k,intsrce,intdest,&
                            & vecmag,vecpha,outflag)
INTEGER, INTENT(IN) :: i, intdest, intsrce, j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2) :: xcomp
REAL, INTENT(IN), DIMENSION(2) :: ycomp
REAL, INTENT(OUT), OPTIONAL :: vecmag, vecpha
```

File

math_library.F90

Type

Module subroutine

Purpose

Calculate the magnitude and phase of a vector, defined at velocity grid nodes, at a C-node grid point

Arguments

xcomp — Vector component in the X-direction at the U-nodes

ycomp — Vector component in the Y-direction at the V-nodes

intsrce — Selects valid points at the velocity nodes for interpolation from U/V- to C-nodes.

i — X-index of the input vector

j — Y-index of the input vector

k — Vertical index of the input vector

intsrce

intsrce — Selects valid points at the velocity nodes for interpolation from U/V- to C-nodes.

0: all points

1: coastal boundaries, interior points and open boundaries only

2: interior points only

3: interior points and open boundaries only

4: coastal boundaries and interior points only

intdest	Selects valid points at the C-nodes for interpolation from U/V- to C-nodes.
	0: all points
	1: wet points only
vecmag	Returned vector magnitude (optional)
vecpha	Returned vector phase (optional) [radians]
outflag	Output flag for invalid points, if present. Zero otherwise.

vector_mag_var_atu

```
SUBROUTINE vector_mag_var_atu(xcomp,ycomp,i,j,k,intsrce,intdest,&
                              & vecmag,vecpha,outflag,hregular)
LOGICAL, INTENT(IN), OPTIONAL :: hregular
INTEGER, INTENT(IN) :: i, intdest, intsrce, j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION :: xcomp
REAL, INTENT(IN), DIMENSION(2,2) :: ycomp
REAL, INTENT(OUT), OPTIONAL :: vecmag, vecpha
```

File

math_library.F90

Type

Module subroutine

Purpose

Calculate the magnitude and phase of a vector, defined at velocity grid nodes, at a U-node grid point

Arguments

xcomp	Vector component in the X-direction at the U-nodes
ycomp	Vector component in the Y-direction at the V-nodes
i	X-index of the input vector
j	Y-index of the input vector
k	Vertical index of the input vector

intsrce	Selects valid points at the V-nodes for interpolation to the U-node. 0: all points 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
intdest	Selects valid points at the U-node for interpolation from the V-nodes. 0: all points 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
vecmag	Returned vector magnitude (optional)
vecpha	Returned vector phase (optional) [radians]
outflag	Output flag for invalid points, if present. Zero otherwise.
hregular	Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.

vector_mag_var_atv

```
SUBROUTINE vector_mag_var_atv(xcomp,ycomp,i,j,k,intsrce,intdest,&
                            & vecmag,vecpha,outflag,hregular)
LOGICAL, INTENT(IN), OPTIONAL :: hregular
INTEGER, INTENT(IN) :: i, intdest, intsrce, j, k
REAL, INTENT(IN), OPTIONAL :: outflag
REAL, INTENT(IN), DIMENSION(2:2) :: xcomp
REAL, INTENT(IN), DIMENSION :: ycomp
REAL, INTENT(OUT), OPTIONAL :: vecmag, vecpha
```

File

math_library.F90

Type

Module subroutine

Purpose

Calculate the magnitude and phase of a vector, defined at velocity grid nodes, at a V-node grid point

Arguments

xcomp	Vector component in the X-direction at the U-nodes
ycomp	Vector component in the Y-direction at the V-nodes
i	X-index of the input vector
j	Y-index of the input vector
k	Vertical index of the input vector
intsrce	Selects valid points at the U-nodes for interpolation to the V-node. 0: all points 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
intdest	Selects valid points at the V-node for interpolation from the U-nodes. 0: all points 1: coastal boundaries, interior points and open boundaries only 2: interior points only 3: interior points and open boundaries only 4: coastal boundaries and interior points only
vecmag	Returned vector magnitude (optional)
vecpha	Returned vector phase (optional) [radians]
outflag	Output flag for invalid points, if present. Zero otherwise.
hregular	Flag to select uniform or area averaging in the horizontal, if present. Otherwise, the type of averaging is selected by iopt_arrint_hreg.

31.14 Output data for standard variables

Defines output for standard variables using their key ids.

- An output operator can optionally be provided: minimum, maximum or mean value, value at a given vertical level or specified depth. For details see Sections 20.1.1.1, 20.2.1.1, 20.3.2.1.
- The routines replace the user-defined usrdef_*vals routines if the key ids of the output variables have been supplied by the user as part of the setup.

define_out0d_vals

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

File
: *model_output.f90*

Type
: Module subroutine

Purpose
: Define 0-D output data.

Arguments

outdat	Returned output data
novars	Number of output variables
outvars	Attributes of the output variables (if ivarid is not defined)
ivarid	Variables's key ids (if outvars is not defined)
numvar	Variable numbers (e.g. sediment fraction number)
oopt	Optional output operator applied for each variable. Allowed values are:
	oopt_null no operator applied
	oopt_mean volume (3-D variable) or surface (2-D variable) averaged value

oopt_max	volume (3-D variable) or surface (2-D variable) maximum value
oopt_min	volume (3-D variable) or surface (2-D variable) minimum value

define_out2d_vals

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

File

model_output.f90

Type

Module subroutine

Purpose

Define 2-D output data at a given location.

Arguments

outdat	Returned output data
i	X-index of the output location
j	Y-index of the output location
novars	Number of output variables
outvars	Attributes of the output variables (if ivarid is not defined)
ivarid	Variables's key ids (if outvars is not defined)
numvar	Variable numbers (e.g. sediment fraction number)
oopt	Optional output operator applied for each variable. Allowed values are:
	oopt_null — no operator applied
	oopt_mean — vertically averaged value (in case ivarid represents a 3-D variable)

oopt_max	maximum value over the vertical (in case ivarid represents a 3-D variable)
oopt_min	minimum value over the vertical (in case ivarid represents a 3-D variable)
oopt_klev	value at a vertical level (in case ivarid represents a 3-D variable)
oopt_dep	value at a specified depth (in case ivarid represents a 3-D variable)
klev	vertical level(s) taken when oopt equals oopt_klev
dep	depth(s) below the sea surface taken when oopt equals oopt_dep
node	defines the vertical node ('C' (default if not present) or 'W' where a W-node (3-D) quantity is evaluated (see Section 20.1.1.1 for details)

define_out3d_vals

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

File

model_output.f90

Type

Module subroutine

Purpose

Define 2-D output data at a given location.

Arguments

outdat	Returned output data
i	X-index of the output location
j	Y-index of the output location
k	vertical index of the output location

novars	Number of output variables
outvars	Attributes of the output variables (if ivarid is not defined)
ivarid	Variables's key ids (if outvars is not defined)
numvar	Variable numbers (e.g. sediment fraction number)
node	defines the vertical node ('C' (default if not present) or 'W' where a W-node quantity is evaluated (see Section 20.1.1.1 for details)

31.15 Standard attributes for variables and forcing files

inquire_var

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

File
modvars_routines.f90

Type
Module subroutine

Purpose
Returns attributes of a model array variable given its variable key id.

Arguments

varid	Variable key id
f90_name	If present, returned FORTRAN 90 name attribute
long_name	If present, returned long name attribute
units	If present, returned units attribute

node	If present, returned grid node where the variable is defined on the model grid
vector_name	If present, returned vector name attribute
data_type	If present, returned data type attribute
nodim	If present, returned array rank
shape	If present, returned array shape
dom_dims	If present, returned array shape without halo
halo_size	If present, returned halo sizes
numvar	Variable number in case ivarid is a multi-variable key id
varatts	If present, the variable attributes are returned in a variable of type VariableAtts

set_modfiles_atts

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

File

modvars_routines.f90

Type

Module subroutine

Purpose

Define the global attributes of a forcing file.

Arguments

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

set_modfiles_name

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

File
modvars_routines.f90

Type
Module subroutine

Purpose
Define the default name of a forcing file. The result is stored in the filename attribute if no name has been defined by the user.

Arguments

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

set_modvars_atts

```
SUBROUTINE set_modvars_atts(iddesc,ifil,iotype,varatts,numvars,&
                          & noprofsd,numprofs,novars)
INTEGER, INTENT(IN) :: iddesc, ifil, iotype, numvars
INTEGER, INTENT(IN), OPTIONAL, DIMENSION(2:) :: noprofsd
INTEGER, INTENT(IN), OPTIONAL :: novars, numprofs
TYPE (VariableAtts), INTENT(INOUT), DIMENSION(:) :: varatts
```

File
modvars_routines.f90

Type
Module subroutine

Purpose
Define the variable attributes of a forcing file.

Arguments

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

2: Output file

varatts — Returned variable attributes

numvars — Number of data (excluding coordinate) variables in the data file

noprofsd — Number of open boundary data profiles in each data file. The parameter is only used if the forcing contains specifiers for 3-D scalar open boundary conditions and ifil = 1).

numprofs — Number of profiles in an open boundary data file. The parameter is only used if the forcing contains specifiers for 3-D scalar open boundary conditions and ifil > 1).

novars — Number of data variables in an open boundary data file. The parameter is only used if the forcing contains specifiers for 3-D scalar open boundary and ifil > 1). Value equals 1 for currents, temperature and salinity.

31.16 Library routines for linear algebra

cholesky_decomp

```
SUBROUTINE cholesky_decomp(a,n,ndim,varname,ierr,info)
LOGICAL, INTENT(IN) :: info
CHARACTER (LEN=*), INTENT(IN) :: varname
INTEGER, INTENT(IN) :: n, ndim
INTEGER, INTENT(OUT) :: ierr
REAL, INTENT(INOUT), DIMENSION(ndim,ndim) :: a
```

File
nla_library.F90

Type
Module subroutine

Purpose
Perform a Cholesky decomposition on a symmetric positive definite matrix.

Reference
Press *et al.* (1992)

Arguments

a	Symmetric, positive-definite matrix on input. On output, the Cholesky matrix is returned in the lower triangular and diagonal part.
n	Physical dimensions of the square matrix a
ndim	Array dimensions of the square matrix a
varname	FORTRAN name of the variable a
ierr	Returned error code
	0: No error occurred
	1: Error occurred
info	Writes info to the log file if .TRUE.

cholesky_solve

```
SUBROUTINE cholesky_solve(a,b,n,ndim,varname,info)
LOGICAL, INTENT(IN) :: info
CHARACTER (LEN=*), INTENT(IN) :: varname
INTEGER, INTENT(IN) :: n, ndim
REAL, INTENT(INOUT), DIMENSION(ndim) :: b
REAL, INTENT(IN), DIMENSION(ndim,ndim) :: a
```

File

nla_library.F90

Type

Module subroutine

Purpose

Solve a linear system of equations using Cholesky decomposition. The coefficient matrix is obtained from a previous call to cholesky_decomp.

Reference

Press *et al.* (1992)

Arguments

a	Cholesky coefficient matrix
b	Right hand side of the linear system on input. Solution vector on output.
n	Number of linear equations (physical dimensions of the square matric a and the vector b)

ndim	Array dimensions of the square matrix a and size of the vector b
varname	FORTRAN name of the variable a
info	Writes info to the log file if .TRUE.

LU_decomp

```
SUBROUTINE LU_decomp(a,indx,n,ndim,varname,ierr,info)
LOGICAL, INTENT(IN) :: info
CHARACTER (LEN=*), INTENT(IN) :: varname
INTEGER, INTENT(IN) :: n, ndim
INTEGER, INTENT(OUT) :: ierr
INTEGER, INTENT(OUT), DIMENSION(ndim) :: indx
REAL, INTENT(INOUT), DIMENSION(ndim,ndim) :: a
```

File

nla_library.F90

Type

Module subroutine

Purpose

Decompose the matrix a into a lower (L) and upper (U) triangular part.

Reference

Press *et al.* (1992)

Arguments

a	On input, the matrix to be decomposed. On output, L is substituted below the diagonal, U at and above the diagonal.
indx	Returned vector recording the row permutation effected by partial pivoting. The vector must be given as argument to a subsequent call to LU_solve
n	Physical dimensions of the square matrix a
ndim	Array dimensions of the square matrix a
varname	FORTRAN name of the variable a
ierr	Returned error code
	0: No error occurred.

1: A zero is found on the diagonal of the matrix a.

2: The matrix a is singular.

info — Writes info to the log file if .TRUE.

LU_solve

```
SUBROUTINE LU_decomp(a,indx,b,n,ndim,varname,info)
LOGICAL, INTENT(IN) :: info
CHARACTER (LEN=*), INTENT(IN) :: varname
INTEGER, INTENT(IN) :: n, ndim
INTEGER, INTENT(IN), DIMENSION(ndim) :: indx
REAL, INTENT(INOUT), DIMENSION(ndim) :: b
REAL, INTENT(IN), DIMENSION(ndim,ndim) :: a
```

File

nla_library.F90

Type

Module subroutine

Purpose

Solve a linear system of equations using LU decomposition

Reference

Press *et al.* (1992)

Arguments

a	LU-decomposed coefficient matrix returned from a previous call to LU_decomp
indx	Vector recording the row permutation effected by partial pivoting as returned from a previous call to LU_decomp
b	Right hand side of the linear system on input. Solution vector on output.
n	Number of linear equations (physical dimensions of the square matrix a and size of the vector b)
ndim	Array dimensions of the square matrix a and size of the vector b
varname	FORTRAN name of the variable a
info	Writes info to the log file if .TRUE.

svd_decomp

```
SUBROUTINE svd_decomp(a,w,v,m,n,varname,ierr,info)
LOGICAL, INTENT(IN) :: info
CHARACTER (LEN=*), INTENT(IN) :: varname
INTEGER, INTENT(IN) :: m,n
INTEGER, INTENT(OUT) :: ierr
REAL, INTENT(OUT), DIMENSION(n) :: w
REAL, INTENT(INOUT), DIMENSION(m,n) :: a
REAL, INTENT(OUT), DIMENSION(n,n) :: v
```

File

nla_library.F90

Type

Module subroutine

Purpose

Performs a singular value decomposition on the matrix a, i.e. $\mathsf{a} = \mathsf{u}\mathsf{d}^t\mathsf{v}$ where u, v are unitary matrices and d a diagonal matrix. The diagonal elements of w are the singular values of a.

Reference

Press *et al.* (1992)

Arguments

a	Matrix to be decomposed on input, unitary matrix u on output
w	Returned vector of singular values (diagonal elements of the diagonal matrix d). If n>m, the last n-m elements are zero.
v	Returned unitary matrix v
m	Number of rows in a
n	Number of columns in a
varname	FORTRAN name of the variable a
ierr	Returned error code
	0: No error occurred
	1: Error occurred
info	Writes info to the log file if .TRUE.

symm_eigen

```
SUBROUTINE symm_eigen(a,d,n,vectors,varname,ierr,info)
LOGICAL, INTENT(IN) :: info, vectors
CHARACTER (LEN=*), INTENT(IN) :: varname
INTEGER, INTENT(IN) :: n
INTEGER, INTENT(OUT) :: ierr
REAL, INTENT(OUT), DIMENSION(n) :: d
REAL, INTENT(INOUT), DIMENSION(n,n) :: a
```

File
nla_library.F90

Type
Module subroutine

Purpose
Returns the eigenvalues and eigenvectors of a real symmetric matrix.

Reference
Press *et al.* (1992)

Arguments

a	Real symmetric matrix on input. On output, the eigenvectors are stored in the columns of a if vectors is .TRUE.. If vectors is .FALSE., the matrix a is modified as well but no longer contains useful information.
d	Vector of eigenvalues in descending order
n	Dimensions of the square matrix a
vectors	Returns the eigenvectors if .TRUE.
varname	FORTRAN name of the variable a
ierr	Returned error code
	0: No error occurred
	1: Error occurred
info	Writes info to the log file if .TRUE.

tridiag_reduce

```
SUBROUTINE tridiag_reduce(a,d,e,n,vectors,varname,info)
LOGICAL, INTENT(IN) :: info
```

```
CHARACTER (LEN=*), INTENT(IN) :: varname
LOGICAL, INTENT(IN) :: vectors
INTEGER, INTENT(IN) :: n
REAL, INTENT(OUT), DIMENSION(n) :: d, e
REAL, INTENT(INOUT), DIMENSION(n,n) :: a
```

File
nla_library.F90

Type
Module subroutine

Purpose
Performs a Householder reduction to a tridiagonal form on a real symmetric matrix.

Reference
Press *et al.* (1992)

Arguments

a	Real symmetric matrix on input. On output, the orthogonal matrix effecting the transformation.
d	Diagonal elements of the output tridiagonal matrix
e	Off-diagonal elements of the output tridiagonal matrix (e(1)=0)
n	Dimensions of the square matrix a
vectors	No eigenvectors are needed if .FALSE.
varname	FORTRAN name of the variable a
info	Writes info to the log file if .TRUE.

tridiag_vert

```
SUBROUTINE tridiag_vert(tridcf,psi,nhdims,nzdim,novars,cnode)
CHARACTER (LEN=lennode), INTENT(IN) :: cnode
INTEGER, INTENT(IN) :: novars, nzdim
INTEGER, INTENT(IN), DIMENSION(4) :: nhdims
REAL, INTENT(IN), DIMENSION(ncloc,nrloc,nzdim,4,novars) :: tridcf
REAL, INTENT(INOUT), DIMENSION(1-nhdims(1):ncloc+nhdims(2),&
                               & 1-nhdims(3):nrloc+nhdims(4),&
                               & nzdim,novars) :: psi
```

File
nla_library.F90

Type
Module subroutine

Purpose
Update a 3-D variable or variables in time by solving a system of tridiagonal equations on each grid point in the horizontal.

Reference
Press *et al.* (1992)

Arguments

tridcf	The first three dimensions correspond to respectively the X-, Y- and vertical directions of the model grid. Index elements 1, 2, 3 and 4 in the last dimension represent respectively the coefficient matrices A, B, C, D in equations (5.315).
psi	Returned updated value of psi
nhdims	Halo sizes of the model grid array psi (WESN directions)
nzdim	Vertical array dimension
novars	Number of model variables stored in psi
cnode	Nodal type of the variables(s) psi

tridiag_vert_1d

```
SUBROUTINE tridiag_vert(tridcf,psi,nzdim,novars,info)
LOGICAL, INTENT(IN) :: info
INTEGER, INTENT(IN) :: novars, nzdim
REAL, INTENT(IN), DIMENSION(nzdim,4,novars) :: tridcf
REAL, INTENT(INOUT), DIMENSION(nzdim,novars) :: psi
```

File
nla_library.F90

Type
Module subroutine

Purpose
Update a variable or variables in time by solving a system of tridiagonal equations in the vertical.

Reference
Press *et al.* (1992)

Arguments

tridcf	The size of the first dimension represents the number of vertical points. Index elements 1, 2, 3 and 4 in the second dimension represents respectively the coefficient matrices A, B, C, D in equations (5.315). The size of the third dimension is the number of variables for which the linear system has to be solved.
psi	Returned updated value of psi
nzdim	Vertical array dimension
novars	Number of model variables stored in psi
info	Writes log info if set to .TRUE.

31.17 Parallel communications

collect_vars

```
SUBROUTINE collect_vars(locvals,procvals,noprocs,ivarid,&
                                          & comm,commtype)
INTEGER, LOGICAL or REAL, INTENT(IN) &
                       & [,DIMENSION(:[,:[,:[,:]]])] ::  locvals
INTEGER, INTENT(IN) :: ivarid, noprocs
INTEGER, INTENT(IN), OPTIONAL :: comm, commtype
INTEGER, LOGICAL or REAL, INTENT(INOUT),&
                         & DIMENSION (:[,:[,:[,:[,:]]]]) ::  procvals
```

File
paral_comms.f90

Type
Generic module subroutine

Purpose
Performs a collect operation, i.e. copies local arrays of the same shape into one globally defined array. The index of the last dimension corresponds to the process number of each local array.

Reference
Section 11.4.3

Arguments

locvals	Local scalar array. Can be of type INTEGER, LOGICAL, REAL. Rank is between 0 and 2 for LOGICAL logical arrays and between 0 and 4 otherwise.
procvals	Array which collects the contributions of all local arrays. The array is globally defined. Rank is between 1 and 3 for LOGICAL logical arrays and between 1 and 5 otherwise.
noprocs	Number of involved processes
ivarid	Variable key id (used for log info only, zero for undefined)
comm	MPI communicator. If not present, its value is comm_work.
commtype	Communication type for the collect operation (between 1 and 5). If not present, its value is set to iopt_MPI_comm_all if iopt_MPI_comm_coll=0 or to 5 otherwise. 1: blocking, standard send 2: blocking, synchronous send 3: non-blocking, standard send 4: non-blocking, synchronous send 5: collective communication

Non-generic versions

collect_vars_int_0d	Integer scalar
collect_vars_int_1d	Integer vector
collect_vars_int_2d	Integer 2-D array
collect_vars_int_3d	Integer 3-D array
collect_vars_int_4d	Integer 4-D array
collect_vars_log_0d	Logical scalar
collect_vars_log_1d	Logical vector
collect_vars_log_2d	Logical 2-D array
collect_vars_real_0d	Real scalar
collect_vars_real_1d	Real vector
collect_vars_real_2d	Real 2-D array
collect_vars_real_3d	Real 3-D array
collect_vars_real_4d	Real 4-D array

combine_mod

```
SUBROUTINE combine_mod(glbvals,locvals,lbounds,ivarid,&
                    & fill,idroot,comm,commtype,commall)
LOGICAL, INTENT(IN), OPTIONAL :: commall
INTEGER, LOGICAL or REAL, INTENT(IN) ::  fill
INTEGER, INTENT(IN) :: ivarid
INTEGER, INTENT(IN), OPTIONAL :: comm, commtype, idroot
INTEGER, INTENT(IN), DIMENSION(:) :: lbounds
INTEGER, LOGICAL or REAL, INTENT(OUT), &
       & DIMENSION(lbounds(1):,lbounds(2)&
       &         [,lbounds(3):[,lbounds(4):]]) ::  glbvals
INTEGER, LOGICAL or REAL, INTENT(IN), &
       & DIMENSION(lbounds(1):,lbounds(2)&
       &         [,lbounds(3):[,lbounds(4):]]) ::  locvals
```

File

paral_comms.f90

Type

Generic module subroutine

Purpose

Performs a combine or combine-all operation on local model grid arrays.

Reference

Section 11.4.2

Arguments

glbvals	Global array. Can be of type INTEGER, LOGICAL, REAL. Rank is 2 for LOGICAL arrays and between 2 and 4 otherwise.
locvals	Local array. Type and rank is the same as *glbvals*.
lbounds	Lower bounds of the local and global arrays. Size must be equal to the rank of *glbvals* and *locvals*.
ivarid	Variable key id (used for log info only, zero for undefined)
fill	Fill value for elements in the global array outside all local domain grids. Type must be the same as *glbvals* and *locvals*.

idroot — Root process where the global array is defined in case of a combine operation. If not present, the root equals the master process.

comm — MPI communicator. If not present, its value is comm_work.

commtype — Communication type for the combine operation (between 1 and 4). If not present, its value is set to iopt_MPI_comm_all for combine-all operations or iopt_MPI_comm_gath otherwise.

1: blocking, standard send

2: blocking, synchronous send

3: non-blocking, standard send

4: non-blocking, synchronous send

commall — A combine-all operation is performed only if present and set to .TRUE.

Non-generic versions

combine_mod_int_2d — Integer 2-D array

combine_mod_int_3d — Integer 3-D array

combine_mod_int_4d — Integer 4-D array

combine_mod_log_2d — Logical 2-D array

combine_mod_real_2d — Real 2-D array

combine_mod_real_3d — Real 3-D array

combine_mod_real_4d — Real 4-D array

combine_stats_glb

```
SUBROUTINE combine_stats_glb(glbvals,maxstats,nostatspocs,lstatsprocs,&
                          & ivarid,idroot,comm,commtype,commall)
LOGICAL, INTENT(IN), OPTIONAL :: commall
INTEGER, INTENT(IN) :: ivarid, maxstats
INTEGER, INTENT(IN), OPTIONAL :: comm, commtype, idroot
INTEGER, INTENT(IN), DIMENSION(nprocs) :: nostatprocs
INTEGER, INTENT(IN), DIMENSION(maxstats,nprocs) :: lstatprocs
INTEGER, LOGICAL or REAL, INTENT(INOUT),&
                        & DIMENSION(:[,:[,:]]) ::  glbvals
```

File

paral_comms.f90

Type

Generic module subroutine

Purpose

Perform a combine operation on a globally defined real array which is defined globally, but whose values are distributed among different processes. The array has a rank between 1 and 4. Array sections with the same first index belong to the same domain.

Arguments

glbvals	Global array. Can be of type INTEGER, LOGICAL, REAL. Rank is 1 for LOGICAL arrays and between 1 and 4 otherwise.
maxstats	First dimension of array lstatprocs
nostatprocs	Number of "stations" in each local array
lstatprocs	Indices of local stations in the global array
ivarid	Variable key id (used for log info only, zero for undefined)
idroot	Root process where the global array is defined in case of a combine operation. If not present, the root equals the master process.
comm	MPI communicator. If not present, its value is comm_work.
commtype	Communication type for the combine operation (between 1 and 4). If not present, its value is set to iopt_MPI_comm_all for combine-all operations or iopt_MPI_comm_gath otherwise. 1: blocking, standard send 2: blocking, synchronous send 3: non-blocking, standard send 4: non-blocking, synchronous send
commall	A combine-all operation is performed only if present and set to .TRUE.

Non-generic versions

combine_stats_glb_int_1d	Integer vector
combine_stats_glb_int_2d	Integer 2-D array
combine_stats_glb_int_3d	Integer 3-D array
combine_stats_glb_int_4d	Integer 4-D array

combine_stats_glb_log_1d Logical vector

combine_stats_glb_real_1d Real vector

combine_stats_glb_real_2d Real 2-D array

combine_stats_glb_real_3d Real 3-D array

combine_stats_glb_real_4d Real 4-D array

combine_stats_loc

```
SUBROUTINE combine_stats_loc(realglb,realloc,maxstats,nostatprocs,&
                      & lstatprocs,ivarid,idroot,comm,commtype)
INTEGER, INTENT(IN) :: ivarid, maxstats
INTEGER, INTENT(IN), OPTIONAL :: comm, commtype, idroot
INTEGER, INTENT(IN), DIMENSION(nprocs) :: nostatprocs
INTEGER, INTENT(IN), DIMENSION(nprocs,maxstats) :: lstatprocs
REAL, INTENT(OUT), DIMENSION(:[,:[,:]]) ::  realglb
REAL, INTENT(IN), DIMENSION(:[,:[,:]]) ::  realloc
```

File

paral_comms.f90

Type

Generic module subroutine

Purpose

Combine local station data into a global array on the root process.

Reference

Section 11.5

Arguments

realglb	Global array. Rank is between 1 and 4.
realloc	Local array. Rank is the same as *realglb*.
maxstats	Second dimension of array lstatprocs
nostatprocs	Number of local data stations
lstatprocs	Index mapping array
ivarid	Variable key id (used for log info only, zero for undefined)
idroot	Root process where the global array is defined. If not present, the root equals the master process.
comm	MPI communicator. If not present, its value is comm_work.

commtype Communication type for the combine operation (between 1 and 4). If not present, its value is set to iopt_MPI_comm_gath.

1: blocking, standard send

2: blocking, synchronous send

3: non-blocking, standard send

4: non-blocking, synchronous send

Non-generic versions

combine_stats_loc_real_1d Real vector

combine_stats_loc_real_2d Real 2-D array

combine_stats_loc_real_3d Real 3-D array

combine_stats_loc_real_4d Real 4-D array

combine_submod

```
SUBROUTINE combine_submod(realglb,realloc,limprocs,ivarid,rfill,&
                          & idroot,comm,commtype)
INTEGER, INTENT(IN) :: ivarid
INTEGER, INTENT(IN), DIMENSION(2,2,nprocs) :: limprocs
INTEGER, INTENT(IN), OPTIONAL :: comm, commtype, idroot
REAL, INTENT(IN) :: rfill
REAL, INTENT(OUT), DIMENSION(:,:[,:[,:]]) ::  realglb
REAL, INTENT(IN), DIMENSION(:,:[,:[,:]]) ::  realloc
```

File

paral_comms.f90

Type

Generic module subroutine

Purpose

Combine local sub-grid model arrays into a global array on the root process.

Reference

Section 11.4.2

Arguments

realglb Global array. Rank is between 2 and 4.

realloc Local array. Rank is the same as *realglb*.

limprocs	Start/end indices in the X- and Y-direction of the local array section within the global array
ivarid	Variable key id (used for log info only, zero for undefined)
rfill	Fill values for elements in the global array with no corresponding element in a local array
idroot	Root process where the global array is defined in case of a combine operation. If not present, the root equals the master process.
comm	MPI communicator. If not present, its value is comm_work.
commtype	Communication type for the combine operation (between 1 and 4). If not present, its value is set to iopt_MPI_comm_gath. 1: blocking, standard send 2: blocking, synchronous send 3: non-blocking, standard send 4: non-blocking, synchronous send

Non-generic versions

combine_submod_real_2d	Real 2-D array
combine_submod_real_3d	Real 3-D array
combine_submod_real_4d	Real 4-D array

copy_chars

```
SUBROUTINE copy_chars(chardat,lenstr,ivarid,idroot,comm,commtype)
INTEGER, INTENT(IN) :: ivarid, lenstr
INTEGER, INTENT(IN), OPTIONAL :: comm, commtype, idroot
CHARACTER (LEN=lenstr), INTENT(INOUT) &
                  & [, DIMENSION(:[,:[,:[,:]]])] ::  chardat
```

File

paral_comms.f90

Type

Generic module subroutine

Purpose

Copies a scalar string or an array of strings from the root to all other processes.

Reference

Section 11.4.2

Arguments

chardat	String scalar or array to be copied. Rank is between 0 and 4.
lenstr	Length of the string(s)
ivarid	Variable key id (used for log info only, zero for undefined)
idroot	Root process where the global array is defined. If not present, the root equals the master process.
comm	MPI communicator. If not present, its value is comm_work.
commtype	Communication type for the copy operation (between 1 and 5). If not present, its value is set to iopt_MPI_comm_scat if iopt_MPI_comm_coll=0 or to 5 otherwise. 1: blocking, standard send 2: blocking, synchronous send 3: non-blocking, standard send 4: non-blocking, synchronous send 5: collective communication

Non-generic versions

copy_chars_0d	Character string
copy_chars_1d	Character string vector
copy_chars_2d	Character string 2-D array
copy_chars_3d	Character string 3-D array
copy_chars_4d	Character string 4-D array

copy_vars

```
SUBROUTINE copy_vars(values,ivarid,idroot,comm,commtype)
INTEGER, INTENT(IN) :: ivarid
INTEGER, INTENT(IN), OPTIONAL :: comm, commtype, idroot
INTEGER, LOGICAL or REAL, INTENT(INOUT) &
                    &  [, DIMENSION(:[,:[,:[,:]]])] ::  values
```

File

paral_comms.f90

Type
Generic module subroutine

Purpose
Copies an integer, logical or real scalar or array data from the root to all other processes.

Reference
Section 11.4.2

Arguments

values	Scalar or array data to be copied. Rank is between 0 and 4.
ivarid	Variable key id (used for log info only, zero for undefined)
idroot	Root process where the global array is defined. If not present, the root equals the master process.
comm	MPI communicator. If not present, its value is comm_work.
commtype	Communication type for the copy operation (between 1 and 5). If not present, its value is set to iopt_MPI_comm_scat if iopt_MPI_comm_coll=0 or to 5 otherwise. 1: blocking, standard send 2: blocking, synchronous send 3: non-blocking, standard send 4: non-blocking, synchronous send 5: collective communication

Non-generic versions

copy_vars_int_0d	Integer scalar
copy_vars_int_1d	Integer vector
copy_vars_int_2d	Integer 2-D array
copy_vars_int_3d	Integer 3-D array
copy_vars_int_4d	Integer 4-D array
copy_vars_log_0d	Integer scalar
copy_vars_log_1d	Logical vector
copy_vars_log_2d	Logical 2-D array
copy_vars_log_3d	Logical 3-D array

copy_vars_log_4d Logical 4-D array

copy_vars_real_0d Real scalar

copy_vars_real_1d Real vector

copy_vars_real_2d Real 2-D array

copy_vars_real_3d Real 3-D array

copy_vars_real_4d Real 4-D array

distribute_mod

```
SUBROUTINE distribute_mod(glbvals,locvals,lbounds,nhdist,&
                        & ivarid,fill,shared,idroot,comm,commtype)
LOGICAL, INTENT(IN), OPTIONAL :: shared
INTEGER, LOGICAL or REAL, INTENT(IN) ::  fill
INTEGER, INTENT(IN) :: ivarid
INTEGER, INTENT(IN), OPTIONAL :: comm, commtype, idroot
INTEGER, INTENT(IN), DIMENSION(:) :: lbounds
INTEGER, INTENT(IN), DIMENSION(4) :: nhdist
INTEGER, LOGICAL or REAL, INTENT(IN), DIMENSION(lbounds(1):,lbounds(2):&
                         & [,lbounds(3):[,lbounds(4):]]) ::  glbvals
INTEGER, LOGICAL or REAL, INTENT(IN), DIMENSION(lbounds(1):,lbounds(2):&
                         & [,lbounds(3):[,lbounds(4):]]) ::  locvals
```

File

paral_comms.f90

Type

Generic module subroutine

Purpose

Distributes an integer, logical or real model grid array from the root to all other processes.

Reference

Section 11.4.2

Arguments

glbvals	Global integer, logical or real model grid array to be distributed. Rank is 2 for logical arrays and between 2 and 4 otherwise.
locvals	Returned distributed array. Type and rank is the same as *glbvals*.

lbounds	Lower bounds of the local and global arrays. Size must be equal to the rank of *glbvals* and *locvals*.
nhdist	Halo sizes used in the distribution operation (WESN directions)
ivarid	Variable key id (used for log info only, zero for undefined)
fill	Fill values for the local array. Type must be the same as *glbvals, locvals*.
shared	The global array is defined on all processes if .TRUE. (in-which case no communication is performed), on the root process only if .FALSE.
idroot	Root process where the global array is defined, if shared is .FALSE. If not present, the root equals the master process.
comm	MPI communicator. If not present, its value is comm_work.
commtype	Communication type for the distribute operation (between 1 and 4). If not present, its value is set to iopt_MPI_comm_scat. 1: blocking, standard send 2: blocking, synchronous send 3: non-blocking, standard send 4: non-blocking, synchronous send

Non-generic versions

distribute_mod_int_2d	Integer 2-D array
distribute_mod_int_3d	Integer 3-D array
distribute_mod_int_4d	Integer 4-D array
distribute_mod_log_2d	Logical 2-D array
distribute_mod_real_2d	Real 2-D array
distribute_mod_real_3d	Real 3-D array
distribute_mod_real_4d	Real 4-D array

distribute_mod_hgrid_2d

```
SUBROUTINE distribute_mod_hgrid_2d(surfglb,surfloc,lbounds,&
                                   & nhdist,ivarid,ifill,rfill,&
                                   & shared,idroot,comm,commtype)
LOGICAL, INTENT(IN), OPTIONAL :: shared
INTEGER, INTENT(IN) :: ifill, ivarid
```

```
INTEGER, INTENT(IN), OPTIONAL :: comm, commtype, idroot
REAL, INTENT(IN) :: rfill
INTEGER, INTENT(IN), DIMENSION(2) :: lbounds
INTEGER, INTENT(IN), DIMENSION(4) :: nhdist
TYPE (HRelativeCoords), INTENT(IN),&
                      & DIMENSION(lbounds(1):,lbounds(2):) :: surfglb
TYPE (HRelativeCoords), INTENT(INOUT),&
                      & DIMENSION(lbounds(1):,lbounds(2):) :: surfloc
```

File

paral_comms.f90

Type

Module subroutine

Purpose

Distributes a "global" derived type 2-D model grid array of type HRelativeCoords from the root to all other processes.

Reference

Section 11.4.2

Arguments

surfglb	Global derived type array to be distributed
surfloc	Local distributed derived type array
lbounds	Lower bounds of the local and global arrays.
nhdist	Halo sizes used in the distribution operation (WESN directions)
ivarid	Variable key id (used for log info only, zero for undefined)
ifill	Fill value for integer components
rfill	Fill value for real components
shared	The global array is defined on all processes if .TRUE. (in which case no communication is performed), on the root process only if .FALSE.
idroot	Root process where the global array is defined if shared is .FALSE. If not present, the root equals the master process.
comm	MPI communicator. If not present, its value is comm_work.
commtype	Communication type for the distribute operation (between 1 and 4). If not present, its value is set to iopt_MPI_comm_scat.

1: blocking, standard send

2: blocking, synchronous send

3: non-blocking, standard send

4: non-blocking, synchronous send

exchange_mod

```
SUBROUTINE exchange_mod(values,lbounds,nhexch,ivarid,comm,corners,commtype)
LOGICAL, INTENT(IN), OPTIONAL :: corners
INTEGER, INTENT(IN) :: ivarid
INTEGER, INTENT(IN), OPTIONAL :: comm, commtype
INTEGER, INTENT(IN), DIMENSION(:) :: lbounds
INTEGER, INTENT(IN), DIMENSION(4) :: nhexch
INTEGER, LOGICAL or REAL, INTENT(INOUT), &
            & DIMENSION(lbounds(1):,lbounds(2):&
            &         [,lbounds(3):[,lbounds(4):]]) ::  values
```

File

paral_comms.f90

Type

Generic module subroutine

Purpose

Perform exchange communications between a local process and its neighbours.

Reference

Section 11.4.3

Arguments

values	Local integer, logical or real model grid array. Rank is 2 for logical arrays and between 2 and 4 otherwise.
lbounds	Lower bounds of the local array. Size must be equal to the rank of *values*.
nhexch	Halo sizes used in the exchange operations (WESN directions). A zero value means that no exchange is made in the corresponding direction.
ivarid	Variable key id (used for log info only, zero for undefined)
comm	MPI communicator. If not present, its value is comm_work.

corners Corner communications are disabled if present and set to .FALSE.

commtype Communication type for the exchange operation (between 1 and 5). If not present, its value is set to iopt_MPI_comm_exch.

1: blocking, standard send

2: blocking, synchronous send

3: non-blocking, standard send

4: non-blocking, synchronous send

5: combined send-receive communication

Non-generic versions

exchange_mod_int_2d Integer 2-D array

exchange_mod_int_3d Integer 3-D array

exchange_mod_int_4d Integer 4-D array

exchange_mod_log_2d Logical 2-D array

exchange_mod_real_2d Real 2-D array

exchange_mod_real_3d Real 3-D array

exchange_mod_real_4d Real 4-D array

halo_exch_comms

```
SUBROUTINE halo_exch_comms(arrcomms,nhexch,corners)
LOGICAL, INTENT(IN) :: corners
INTEGER, INTENT(IN), DIMENSION(4) :: nhexch
TYPE(ExchComms), INTENT(OUT), DIMENSION(MaxHaloComms) :: arrcomms
```

File

paral_comms.f90

Type

Module subroutine

Purpose

Set parameters for exchange communications. The routine is called by exchange_mod.

Arguments

arrcomms Properties of the communications performing the exchanges

nhexch — Halo sizes used in the exchange operations (WESN directions). A zero value means that no exchange is made in the corresponding direction.

corners — Corner communications are disabled if set to .TRUE.

31.18 Utility routines for parallel application

max_vars

```
SUBROUTINE max_vars(values,maxval,ivarid,idroot,comm,commtype,&
                  & commall,mask)
LOGICAL, INTENT(IN), OPTIONAL :: commall
INTEGER, INTENT(IN) :: ivarid
INTEGER or REAL, INTENT(OUT) ::  maxval
INTEGER, INTENT(IN), OPTIONAL :: comm, commtype, idroot
LOGICAL, INTENT(IN), OPTIONAL, DIMENSION(:,:) :: mask
INTEGER or REAL, INTENT(IN) [, DIMENSION(:,:[,:])] ::  values
```

File

paral_utilities.f90

Type

Generic module subroutine

Purpose

Compute the global maximum of a scalar or array integer or real variable.

Arguments

values — Local integer or real scalar, 2-D or 3-D model grid array

maxval — Returned global maximum. Type must be the same as *values*.

ivarid — Variable key id (used for log info only, zero for undefined)

idroot — Root process which returns the result in case of a all-to-one communication. If not present, the root equals the master process.

comm — MPI communicator. If not present, its value is comm_work.

commtype Communication type (between 1 and 5). If not present, its value is set to iopt_MPI_comm_gath or iopt_MPI_comm_all if iopt_MPI_comm_coll=0 or to 5 (collective call) otherwise.

1: blocking, standard send

2: blocking, synchronous send

3: non-blocking, standard send

4: non-blocking, synchronous send

5: collective communication

commall Result is returned by all processes if present and .TRUE.

mask If present, points where mask is .FALSE. are excluded. Otherwise all points are included. The argument is not allowed for scalar input data.

Non-generic versions

max_vars_int_0d Integer scalar

max_vars_int_2d Integer 2-D array

max_vars_int_3d Integer 3-D array

max_vars_real_0d Real scalar

max_vars_real_2d Real 2-D array

max_vars_real_3d Real 3-D array

maxloc_vars

```
SUBROUTINE maxloc_vars(values,maxval,maxpos,ivarid,&
                     & idroot,comm,commtype,commall,mask)
LOGICAL, INTENT(IN), OPTIONAL :: commall
INTEGER, INTENT(IN) :: ivarid
INTEGER or REAL, INTENT(OUT) ::  maxval
INTEGER, INTENT(IN), OPTIONAL :: comm, commtype, idroot
INTEGER, INTENT(OUT), DIMENSION(:) :: maxpos
LOGICAL, INTENT(IN), OPTIONAL, DIMENSION(:,:) :: mask
INTEGER or REAL, INTENT(IN), DIMENSION(:,:[,:])] ::  values
```

File

paral_utilities.f90

Type

Generic module subroutine

Purpose

Compute the value and index location of a global maximum of an integer or real 2-D or 3-D model grid array.

Arguments

values	Local integer or real 2-D or 3-D model grid array
maxval	Returned global maximum. Type must be the same as *values*.
maxpos	Global indices of the location of the maximum. Size equals the rank of *values*.
ivarid	Variable key id (used for log info only, zero for undefined)
idroot	Root process which returns the result in case of a all-to-one communication. If not present, the root equals the master process.
comm	MPI communicator. If not present, its value is comm_work.
commtype	Communication type (between 1 and 4). If not present, its value is set to iopt_MPI_comm_gath or iopt_MPI_comm_all. 1: blocking, standard send 2: blocking, synchronous send 3: non-blocking, standard send 4: non-blocking, synchronous send
commall	Result is returned by all processes if present and .TRUE.
mask	If present, points where mask is .FALSE. are excluded. Otherwise all points are included. The argument is not allowed for scalar input data.

Non-generic versions

maxloc_vars_int_2d	Integer 2-D array
maxloc_vars_int_3d	Integer 3-D array
maxloc_vars_real_2d	Real 2-D array
maxloc_vars_real_3d	Real 3-D array

min_vars

```
SUBROUTINE min_vars(values,minval,ivarid,idroot,comm,commtype,&
                  & commall,mask)
LOGICAL, INTENT(IN), OPTIONAL :: commall
```

```
INTEGER, INTENT(IN) :: ivarid
INTEGER or REAL, INTENT(OUT) :: minval
INTEGER, INTENT(IN), OPTIONAL :: comm, commtype, idroot
LOGICAL, INTENT(IN), OPTIONAL, DIMENSION(:,:) :: mask
INTEGER or REAL, INTENT(IN) [, DIMENSION(:,:[,:])] :: values
```

File

paral_utilities.f90

Type

Generic module subroutine

Purpose

Compute the global minimum of a scalar or array integer or real variable.

Arguments

values	Local integer or real scalar, 2-D or 3-D model grid array
minval	Returned global minimum. Type must be the same as *values*.
ivarid	Variable key id (used for log info only, zero for undefined)
idroot	Root process which returns the result in case of a all-to-one communication. If not present, the root equals the master process.
comm	MPI communicator. If not present, its value is comm_work.
commtype	Communication type (between 1 and 5). If not present, its value is set to iopt_MPI_comm_gath or iopt_MPI_comm_all if iopt_MPI_comm_coll=0 or to 5 (collective call) otherwise. 1: blocking, standard send 2: blocking, synchronous send 3: non-blocking, standard send 4: non-blocking, synchronous send 5: collective communication
commall	Result is returned by all processes if present and .TRUE.
mask	If present, points where mask is .FALSE. are excluded. Otherwise all points are included. The argument is not allowed for scalar input data.

Non-generic versions

- min_vars_int_0d Integer scalar
- min_vars_int_2d Integer 2-D array
- min_vars_int_3d Integer 3-D array
- min_vars_real_0d Real scalar
- min_vars_real_2d Real 2-D array
- min_vars_real_3d Real 3-D array

minloc_vars

```
SUBROUTINE minloc_vars(values,minval,minpos,ivarid,&
                     & idroot,comm,commtype,commall,mask)
LOGICAL, INTENT(IN), OPTIONAL :: commall
INTEGER, INTENT(IN) :: ivarid
INTEGER or REAL, INTENT(OUT) ::  minval
INTEGER, INTENT(IN), OPTIONAL :: comm, commtype, idroot
INTEGER, INTENT(OUT), DIMENSION(:) :: minpos
LOGICAL, INTENT(IN), OPTIONAL, DIMENSION(:,:) :: mask
INTEGER or REAL, INTENT(IN), DIMENSION(:,:[,:])] ::  values
```

File

paral_utilities.f90

Type

Generic module subroutine

Purpose

Compute the value and index location of a global minimum of an integer or real 2-D or 3-D model grid array.

Arguments

values	Local integer or real 2-D or 3-D model grid array
minval	Returned global minimum. Type must be the same as *values*.
minpos	Global indices of the location of the minimum. Size equals the rank of *values*.
ivarid	Variable key id (used for log info only, zero for undefined)
idroot	Root process which returns the result in case of a all-to-one communication. If not present, the root equals the master process.

comm — MPI communicator. If not present, its value is comm_work.

commtype — Communication type (between 1 and 4). If not present, its value is set to iopt_MPI_comm_gath or iopt_MPI_comm_all.

1: blocking, standard send

2: blocking, synchronous send

3: non-blocking, standard send

4: non-blocking, synchronous send

commall — Result is returned by all processes if present and .TRUE.

mask — If present, points where mask is .FALSE. are excluded. Otherwise all points are included. The argument is not allowed for scalar input data.

Non-generic versions

minloc_vars_int_2d — Integer 2-D array

minloc_vars_int_3d — Integer 3-D array

minloc_vars_real_2d — Real 2-D array

minloc_vars_real_3d — Real 3-D array

sum_vars

```
SUBROUTINE sum_vars(values,sumval,ivarid,idroot,comm,commtype,&
                  & commall,mask)
LOGICAL, INTENT(IN), OPTIONAL :: commall
INTEGER, INTENT(IN) :: ivarid
INTEGER or REAL, INTENT(OUT) ::  sumval
INTEGER, INTENT(IN), OPTIONAL :: comm, commtype, idroot
LOGICAL, INTENT(IN), OPTIONAL, DIMENSION(:,:) :: mask
INTEGER or REAL, INTENT(IN) [, DIMENSION(:,:[,:])] ::  values
```

File

paral_utilities.f90

Type

Generic module subroutine

Purpose

Compute the global sum of a scalar or array integer or real variable. The result depends on the number of processes, but it more efficient than sum2_vars (see below).

Arguments

values	Local integer or real scalar, 2-D or 3-D model grid array
sumval	Returned global sum. Type must be the same as *values*.
ivarid	Variable key id (used for log info only, zero for undefined)
idroot	Root process which returns the result in case of a all-to-one communication. If not present, the root equals the master process.
comm	MPI communicator. If not present, its value is comm_work.
commtype	Communication type (between 1 and 5). If not present, its value is set to iopt_MPI_comm_gath or iopt_MPI_comm_all if iopt_MPI_comm_coll=0 or to 5 (collective call) otherwise. 1: blocking, standard send 2: blocking, synchronous send 3: non-blocking, standard send 4: non-blocking, synchronous send 5: collective communication
commall	Result is returned by all processes if present and .TRUE.
mask	If present, points where mask is .FALSE. are excluded. Otherwise all points are included. The argument is not allowed for scalar input data.

Non-generic versions

sum_vars_int_0d	Integer scalar
sum_vars_int_2d	Integer 2-D array
sum_vars_int_3d	Integer 3-D array
sum_vars_real_0d	Real scalar
sum_vars_real_2d	Real 2-D array
sum_vars_real_3d	Real 3-D array

sum2_vars

```
SUBROUTINE sum2_vars(realdat,realsum,nhdims,cnode,ivarid,idroot,&
                   & comm,commtype,commall,mask)
LOGICAL, INTENT(IN), OPTIONAL :: commall
CHARACTER (LEN=lennode), INTENT(IN) :: cnode
```

```
INTEGER, INTENT(IN) :: ivarid
INTEGER, INTENT(IN), OPTIONAL :: comm, commtype, idroot
REAL, INTENT(OUT) :: realsum
INTEGER, INTENT(IN), DIMENSION(4) :: nhdims
LOGICAL, INTENT(IN), OPTIONAL, DIMENSION(:,:) :: mask
REAL, INTENT(IN), DIMENSION(:,:[,:[,:]])] ::  realdat
```

File

paral_utilities.f90

Type

Generic module subroutine

Purpose

Compute the global sum of a real model grid array. The process does not depend on the domain decomposition, but is less efficient than sum_vars. A land mask is always applied.

Arguments

realdat	Local real scalar, 2-D, 3-D or 4-D model grid array
realsum	Returned global sum.
nhdims	Halo sizes of the model grid array (WESN directions)
cnode	Grid node where the model array is defined: 'C', 'U', 'V', 'E' (corner node)
ivarid	Variable key id (used for log info only, zero for undefined)
idroot	Root process which returns the result in case of a all-to-one communication. If not present, the root equals the master process.
comm	MPI communicator. If not present, its value is comm_work.
commtype	Communication type (between 1 and 4). If not present, its value is set to iopt_MPI_comm_gath or iopt_MPI_comm_all. 1: blocking, standard send 2: blocking, synchronous send 3: non-blocking, standard send 4: non-blocking, synchronous send
commall	Result is returned by all processes if present and .TRUE.

mask — If present, points where mask is .FALSE. are excluded. Otherwise, the dry points are obtained from the grid pointers arrays. The argument has primarily been introduced to make allowance for either permanently (land) or temporarily dry grid points at C-nodes.

Non-generic versions

sum2_vars_real_2d Real 2-D array

sum2_vars_real_3d Real 3-D array

sum2_vars_real_4d Real 4-D array

31.19 Reset of setup parameters

When a usrdef_ routines is called (or parameters are read from a CIF), some parameters may need to be reset by calling appropriate "reset" routines. For example, reset_mod_params is called after usrdef_mod_params and reset_initial_conditions after usrdef_phsics.

reset_initial_conditions

```
SUBROUTINE reset_initial_conditions
```

File
reset_model.F90

Type
Module subroutine

Purpose
Reset initial conditions.

reset_mod_params

```
SUBROUTINE reset_mod_params
```

File
reset_model.F90

Type
Module subroutine

Purpose

Reset model parameters, defined either by default or by the user. A warning message is written in most cases.

reset_mod_vars

```
SUBROUTINE reset_mod_vars(varatts)
TYPE (VariableAtts), INTENT(INOUT) :: varatts
```

File

reset_model.F90

Type

Module subroutine

Purpose

Define the attributes of variables in model forcing files.

Arguments

varatts	Returned variable attributes

reset_out_files

```
SUBROUTINE rest_out_files(filepars,status)
TYPE (FileParams), INTENT(INOUT), DIMENSION(:[,:]) ::  filepars
CHARACTER (LEN=1), INTENT(IN),&
                 & DIMENSION(SIZE(filepars,DIM=1)) :: status
```

File

reset_model.F90

Type

Generic module subroutine

Purpose

Reset the status and iunit attributes of user output files.

Arguments

filepars	Attributes of the user output files
status	Values of the new status attributes

Non-generic versions

reset_out_files_1d	Vector array of files
reset_out_files_2d	2-D array of files

reset_out_gpars

```
SUBROUTINE rest_out_gpars(outgpars,arrname,tseries,end_reset)
LOGICAL, INTENT(IN) :: end_reset, tseries
CHARACTER (LEN=*), INTENT(IN) :: arrname
TYPE (OutGridParams), INTENT(INOUT), DIMENSION(:) :: outgpars
```

File
reset_model.F90

Type
Module subroutine

Purpose
Reset the attributes of a user defined output grid.

Arguments

outgpars	Attributes of the user output grid
arrname	Name of the attributes array
tseries	.TRUE. for time series output, .FALSE. otherwise
end_reset	If .TRUE., resets the last output time (defined by the tlims(2) attribute) to the latest possible value before the end of the simulation.

reset_out_stats

```
SUBROUTINE rest_out_stats(statlocs)
TYPE (StationLocs), INTENT(INOUT), DIMENSION(:) :: statlocs
```

File
reset_model.F90

Type
Module subroutine

Purpose
Reset the attributes of user defined output station attributes.

Arguments

statlocs	Attributes of the user defined output stations

reset_out_vars

```
SUBROUTINE reset_out_vars(varatts)
TYPE (VariableAtts), INTENT(INOUT), DIMENSION(:) :: varatts
```

File
reset_model.F90

Type
Module subroutine

Purpose
Define the attributes of user output variables in case they are not defined by the user.

Arguments

varatts	Returned variable attributes

31.20 Random generators

A library for the generation of random numbers is implemented. The routines are based on the algorithm developed by L'Ecuyer & Côté (1991). The random numbers are either taken between a given interval or from a normalised distribution with a given mean and standard deviation. The latter routines are taken form Press *et al.* (1992). The library is used as follows

1. All random generators are initialised in the program by calling rng_init. In the current implementation all generators use the default initial seeed.

2. A random generator is opened by rng_open. The routine returns a "generator" id number which is used in all subsequent calls. Up to 32 independent generators can be open at the same time.

3. Random number(s) are generated by the following routines

   - within a given interval: rng_uniform_var, rng_uniform_arr
   - with a given mean and standard deviation: rng_normal_var, rng_normal_arr

4. A random generator can be re-set for a new independent series of random numbers by calling rng_reset. The routine can be used for parallel applications to prevent that the same numbers are generated on different sub-domains.

5. A random generator is closed by calling rng_close. After this call the generator number id can no longer be used.

6. All generators are closed and parameters are reset to their initial default conditions by a call in the program of rng_finalize at the end of a simulation.

rng_close

```
SUBROUTINE rng_close(numgen)
INTEGER, INTENT(IN), OPTIONAL :: numgen
```

File
rng_library.f90

Type
Module subroutine

Purpose
Disable ("close") a random generator.

Arguments

numgen	Generator id

rng_finalize

```
SUBROUTINE rng_finalize
```

File
rng_library.f90

Type
Module subroutine

Purpose
Close all generators and reset parameters to the their default values. The routine is called by the program at the end of a simulation.

rng_init

```
SUBROUTINE rng_init
```

File
rng_library.f90

Type
Module subroutine

Purpose
Set the random seed for all generators and initialise other parameters.

rng_multmod_decompos

```
FUNCTION rng_multmod_decompos(a,s,m)
INTEGER, INTENT(IN) :: a, m, s
INTEGER :: rng_multmod_decompos
```

File
rng_library.f90

Type
Module function

Purpose
Returns $(as)\mathrm{mod}(m)$. The function is used internally in the library.

Reference
L'Ecuyer & Côté (1991)

rng_normal_arr

```
SUBROUTINE rng_normal_arr(xrand,ncount,numgen,xmean,xstd)
INTEGER, INTENT(IN) :: ncount, numgen
REAL, INTENT(IN) :: xmean, xstd
REAL, INTENT(OUT), DIMENSION(ncount) :: xrand
```

File
rng_library.f90

Type
Module subroutine

Purpose

Generate a vector of random numbers with mean xmean and standard deviation xstd.

Arguments

xrand	Returned random numbers
ncount	Size of the returned vector
numgen	Generator id
xmean	Mean value
xstd	Standard deviation

rng_normal_var

```
SUBROUTINE rng_normal_var(xrand,numgen,xmean,xstd)
INTEGER, INTENT(IN) :: numgen
REAL, INTENT(OUT) :: xrand
REAL, INTENT(IN) :: xmean, xstd
```

File

rng_library.f90

Type

Module subroutine

Purpose

Generate a random number with mean xmean and standard deviation xstd.

Arguments

xrand	Returned random number
numgen	Generator id
xmean	Mean value
xstd	Standard deviation

rng_open

```
SUBROUTINE rng_open(numgen)
INTEGER, INTENT(OUT) :: numgen
```

File

rng_library.f90

Type
Module subroutine

Purpose
Enable ("open") a random generator.

Arguments

numgen	Returned generator id

rng_opened

```
SUBROUTINE rng_opened(numgen,flag)
LOGICAL, INTENT(OUT) :: flag
INTEGER, INTENT(IN) :: numgen
```

File
rng_library.f90

Type
Module subroutine

Purpose
Returns flag as .TRUE. if the generator with id numgen has been opened.

Arguments

numgen	Generator id
flag	Returned flag

rng_reset

```
SUBROUTINE rng_reset(numgen,seedtype)
CHARACTER (LEN=1), INTENT(IN)  :: seedtype
INTEGER, INTENT(IN) :: numgen
```

File
rng_library.f90

Type
Module subroutine

Purpose
Reset an open generator according to the value of seedtype. The routine is called in parallel mode to enable independent series of random numbers on different sub-domains with the same random generator.

Reference
L'Ecuyer & Côté (1991)

Arguments

numgen	Generator id
seedtype	Type of reset
	'I' reset to the initial state
	'L' no reset
	'N' a new set a random numbers will be generated which are different from the initial state

rng_standard_normal

```
SUBROUTINE rng_standard_normal(xran,numgen)
INTEGER, INTENT(IN) :: numgen
REAL, INTENT(OUT) :: xran
```

File
rng_library.f90

Type
Module subroutine

Purpose
Generate a random number with zero mean and unit variance.

Reference
Routine gasdev from Press *et al.* (1992)

Arguments

xran	Returned random number
numgen	Generator id

rng_standard_uniform

```
SUBROUTINE rng_standard_normal_uniform(xran,numgen)
INTEGER, INTENT(IN) :: numgen
REAL, INTENT(OUT) :: xran
```

File
rng_library.f90

Type
Module subroutine

Purpose
Generate a random number between 0 and 1.

Reference
L'Ecuyer & Côté (1991)

Arguments

xran	Returned random number
numgen	Generator id

rng_uniform_arr

```
SUBROUTINE rng_uniform_arr(xrand,nosize,numgen,xlo,xhi)
INTEGER, INTENT(IN) :: nosize, numgen
REAL, INTENT(IN) :: xhi, xlo
REAL, INTENT(OUT), DIMENSION(nosize) :: xrand
```

File
rng_library.f90

Type
Module subroutine

Purpose
Generate a vector of random numbers between xlo and xhi.

Arguments

xrand	Returned vector of random numbers
nosize	Size of the random vector
numgen	Generator id
xlo	Lower limit
xhi	Upper limit

rng_uniform_var

```
SUBROUTINE rng_uniform_var(xrand,numgen,xlo,xhi)
INTEGER, INTENT(IN) :: numgen
REAL, INTENT(OUT) :: xrand
REAL, INTENT(IN) :: xhi, xlo
```

File
rng_library.f90

Type
Module subroutine

Purpose
Generate a random number between xlo and xhi.

Arguments

xrand	Returned vector of random numbers
numgen	Generator id
xlo	Lower limit
xhi	Upper limit

31.21 Time and calendar date routines

add_secs_to_date

```
SUBROUTINE add_secs_to_date(datetime1,datetime2,numsteps,dsecs)
CHARACTER (LEN=lentime) or INTEGER, DIMENSION(7),&
                                  &  INTENT(IN) ::  datetime1
CHARACTER (LEN=lentime) or INTEGER, DIMENSION(7),&
                                   & INTENT(OUT) ::  datetime2
INTEGER, INTENT(IN) :: numsteps
REAL, INTENT(IN) :: dsecs
```

File
time_routines.f90

Type
Generic module subroutine

Purpose
Add a number of number of time steps to a given date.

Arguments

datetime1	Initial date and time
datetime2	Returned new date and time. Must have the same type as *datetime1*.
numsteps	Number of time steps

dsecs	Number of seconds in one time step	[s]

Non-generic versions

add_secs_to_date_char	date and time in string format
add_secs_to_date_int	date and time in integer vector format

add_secs_to_phase

```
SUBROUTINE add_secs_to_phase(phasein,phaseout,numsteps,dsecs,freq)
INTEGER, INTENT(IN) :: numsteps
REAL, INTENT(IN) :: dsecs, freq, phasein
REAL, INTENT(OUT) :: phaseout
```

File

time_routines.f90

Type

Module subroutine

Purpose

Update an harmonic phase after a number of time steps. The result is returned modulo 2π.

Arguments

phasein	Initial phase	[rad]
phaseout	Returned updated phase	[rad]
numsteps	Number of time steps	
dsecs	Number of seconds in one time step	[s]
freq	Harmonic frequency	[rad/s]

check_date

```
SUBROUTINE check_date(datetime,varname)
CHARACTER (LEN=lentime) or INTEGER, DIMENSION(7),&
                                    & INTENT(IN) ::  datetime
CHARACTER (LEN=*), INTENT(IN) :: arrname
```

File

time_routines.f90

Type
Generic module subroutine

Purpose
Check a calendar time and time in string or integer format.

Arguments

datetime	Calendar date and time
varname	Name of the date/time variable

Non-generic versions

check_date_char	calendar date and time in string format
check_date_int	calendar date and time in integer vector format

clock_date_time

```
SUBROUTINE clock_date_time(charclock)
CHARACTER (LEN=lentime), INTENT(OUT) :: charclock
```

File
time_routines.f90

Type
Module subroutine

Purpose
Returns the calendar date and time from the machine's internal real-time clock.

Arguments

charclock	Returned internal calendar date and time in COHERENS string format

convert_date

```
FUNCTION convert_date(datetime1) RESULT(datetime2)
CHARACTER (LEN=lentime) or INTEGER, DIMENSION(7),&
                                    & INTENT(IN) ::  datetime1
INTEGER, DIMENSION(7) or CHARACTER (LEN=lentime) :  datetime2
```

File
time_routines.f90

Type
: Generic module subroutine

Purpose
: Convert a calendar date and time from string to integer format or vice versa.

Arguments

datetime1	Calendar date and time on input
datetime2	Returned calendar date and time in the opposite format of *datetime1*

Non-generic versions

check_date_to_char	convert to integer format
check_date_to_int	convert to string format

date_to_year

```
SUBROUTINE date_to_year(datetime,yearnum)
CHARACTER (LEN=lentime) or INTEGER, DIMENSION(7),&
                                  &  INTENT(IN) ::  datetime
REAL, INTENT(OUT) :: yearnum
```

File
: *time_routines.f90*

Type
: Generic module subroutine

Purpose
: Convert a calendar date and time in string or integer format to a year number with a decimal part.

Arguments

datetime	Calendar date and time
yearnum	Returned year number

Non-generic versions

date_to_year_char	calendar date and time is in string format
date_to_year_int	calendar date and time is in integer format

date_number

```
SUBROUTINE date_to_year(datetime,daynum)
CHARACTER (LEN=lentime) or INTEGER, DIMENSION(7),&
                                    & INTENT(IN) :: datetime
REAL, INTENT(OUT) :: daynum
```

File
time_routines.f90

Type
Generic module subroutine

Purpose
Returns the day number of the year. Result is between 0 and 365 (or 366 for a leap year).

Arguments

datetime	Calendar date and time
daynum	Returned day number

Non-generic versions

day_number_char	calendar date and time is in string format
day_number_int	calendar date and time is in integer format

diff_clock

```
FUNCTION diff_clock(npccold)
INTEGER, INTENT(IN) :: npccold
INTEGER :: diff_clock
```

File
time_routines.f90

Type
Module function

Purpose
Returns the number of clock counts since a previous call to read_clock.

Arguments

npccold	Number of previous clock counts

diff_dates

```
SUBROUTINE diff_dates(datetime1,datetime2,tunit,nosecs,&
                    & millisecs,rtime)
CHARACTER (LEN=lentime) or INTEGER, DIMENSION(7),&
                             & INTENT(IN) ::  datetime1, datetime2
INTEGER, INTENT(IN) :: tunit
INTEGER, INTENT(OUT), OPTIONAL :: millisecs
INTEGER (KIND=kndilong), INTENT(OUT), OPTIONAL :: nosecs
REAL, INTENT(OUT), OPTIONAL :: rtime
```

File

time_routines.f90

Type

Generic module subroutine

Purpose

Returns the time between two given calendar dates. Result is negative if the second date is earlier than the first one.

Arguments

datetime1	First calendar date and time
datetime2	Second calendar date and time in the same format as *datetime1*
tunit	Time unit for the result
	0: seconds and (optionally) milliseconds
	1: seconds
	2: minutes
	3: hours
	4: days
	5: months
	6: years
nosecs	If present and tunit=0, the result is in integer seconds.
millisecs	If present and tunit=0, the residual number of milliseconds (between 0 and 999).
rtime	If present and tunit>0, the result is in real format (with milliseconds set to zero).

Non-generic versions

diff_dates_char calendar dates are in string format

diff_dates_int calendar dates are in integer format

.earlier.

```
Usage: datetime1.earlier.datetime2
CHARACTER (LEN=lentime) or INTEGER(7),&
                          & INTENT(IN) ::  datetime1, datetime2
LOGICAL operator :: earlier
```

File

time_routines.f90

Type

Generic operator

Purpose

Returns .TRUE. is *datetime1* is earlier than *datetime2*.

Arguments

datetime1 First calendar date and time

datetime2 Second calendar date and time in the same format as *datetime1*

Non-generic versions

earlier_char compares calendar dates in string format

earlier_int compares calendar dates in integer format

error_lbound_var_date

```
SUBROUTINE error_lbound_var_date(cdate,varname,cdatemin,matchmin)
LOGICAL, INTENT(IN) :: matchmin
CHARACTER (LEN=*), INTENT(IN) :: varname
CHARACTER (LEN=lentime), INTENT(IN) :: cdate, cdatemin
```

File

time_routines.f90

Type

Module subroutine

Purpose

Checks whether cdate is not earlier than datemin if matchmin is .TRUE. or is later than cdatemin if matchmin is .FALSE.

Arguments

cdate	Calendar date and time to be checked
varname	Name of the calendar date variable
cdatemin	Earliest allowed calendar date
machtmin	If .FALSE. (.TRUE.), cdate must be later (not earlier) than cdatemin.

error_ubound_var_date

```
SUBROUTINE error_ubound_var_date(cdate,varname,cdatemax,matchmax)
LOGICAL, INTENT(IN) :: matchmax
CHARACTER (LEN=*), INTENT(IN) :: varname
CHARACTER (LEN=lentime), INTENT(IN) :: cdate, cdatemax
```

File

time_routines.f90

Type

Module subroutine

Purpose

Checks whether cdate is not later than cdatemax if matchmax is .TRUE. or is earlier than cdatemax if matchmax is .FALSE.

Arguments

cdate	Calendar date and time to be checked
varname	Name of the calendar date variable
cdatemax	Latest allowed calendar date
matchmin	If .FALSE. (.TRUE.), cdate must be earlier (not later) than cdatemax.

initialise_time

```
SUBROUTINE initialise_time
```

File

time_routines.f90

Type
Module subroutine

Purpose
Routine called by the program at the initial time to initialise a series of parameters related to date and time.

.later.

```
Usage: datetime1.later.datetime2
CHARACTER (LEN=lentime) or INTEGER(7),&
                          & INTENT(IN) ::  datetime1, datetime2
LOGICAL operator :: .later.
```

File
time_routines.f90

Type
Generic operator

Purpose
Returns .TRUE. is *datetime1* is later than *datetime2*.

Arguments

datetime1	First calendar date and time
datetime2	Second calendar date and time in the same format as *datetime1*

Non-generic versions

later_char	compares calendar dates in string format
later_int	compares calendar dates in integer format

leap_year

```
FUNCTION leap_year(iyear)
INTEGER, INTENT(IN) :: iyear
INTEGER :: leap_year
```

File
time_routines.f90

Type
Module function

Purpose
Returns the number of leap days (0 or 1) in a given year.

Arguments

iyear	Given year

log_timer_in

```
SUBROUTINE log_timer_in(npcc,ivarid,logname,numvar,info)
LOGICAL, INTENT(IN), OPTIONAL :: info
CHARACTER (LEN=*), INTENT(IN), OPTIONAL :: logname
INTEGER, INTENT(IN), OPTIONAL :: ivarid, numvar
INTEGER, INTENT(OUT), OPTIONAL :: npcc
```

File
time_routines.f90

Type
Module subroutine

Purpose
Writes the tracing info to a log file when a routine is called and reads (optionally) the current machine's clock count.

Arguments

npcc	If present, the returned clock count.
ivarid	If present, the key id of variable whose name is added in the log message
logname	If not present, the log message writes the name of the calling routine (i.e. the one at the highest program level). Otherwise, this name is replaced by the string logname.
numvar	Variable number in case of a multi-variable key id
info	No log message will be written if this argument is present and .FALSE.

log_timer_out

```
SUBROUTINE log_timer_out(npcc,itm_type,info)
LOGICAL, INTENT(IN), OPTIONAL :: info
INTEGER, INTENT(IN), OPTIONAL :: itm_type, npcc
```

File
: *time_routines.f90*

Type
: Module subroutine

Purpose
: Writes the tracing info to a log file before a routine is exited and update the number of clock counts for a given timer.

Arguments

npcc	If present, the clock count from a previous call to log_timer_in.
itm_type	If present, the key id of the timer which needs to be updated.
info	No log message will be written if this argument is present and .FALSE.

.noearlier.

```
Usage: datetime1.noearlier.datetime2
CHARACTER (LEN=lentime) or INTEGER(7),&
                          & INTENT(IN) ::  datetime1, datetime2
LOGICAL operator :: noearlier
```

File
: *time_routines.f90*

Type
: Generic operator

Purpose
: Returns .TRUE. is *datetime1* is not earlier than *datetime2*.

Arguments

datetime1	First calendar date and time

datetime2 Second calendar date and time in the same format as *datetime1*

Non-generic versions

noearlier_char compares calendar dates in string format

noearlier_int compares calendar dates in integer format

.nolater.

```
Usage: datetime1.nolater.datetime2
CHARACTER (LEN=lentime) or INTEGER(7),&
                          & INTENT(IN) ::  datetime1, datetime2
LOGICAL operator :: .nolater.
```

File

time_routines.f90

Type

Generic operator

Purpose

Returns .TRUE. is *datetime1* is not later than *datetime2*.

Arguments

datetime1 First calendar date and time

datetime2 Second calendar date and time in the same format as *datetime1*

Non-generic versions

nolater_char compares calendar dates in string format

nolater_int compares calendar dates in integer format

num_time_steps

```
SUBROUTINE num_time_steps(datetime1,datetime2,dsecs,numsteps)
CHARACTER (LEN=lentime) or INTEGER, DIMENSION(7),&
                                & INTENT(IN) ::  datetime1, datetime2
INTEGER, INTENT(OUT) :: numsteps
REAL, INTENT(IN) :: dsecs
```

File

time_routines.f90

Type
Generic module subroutine

Purpose
Returns the number of time steps between two given calendar dates. Result is negative if the second date is earlier than the first one and rounded from below if the number of seconds between the two dates is not an integer multiple of dsecs.

Arguments

datetime1	First calendar date and time	
datetime2	Second calendar date and time in the same format as *datetime1*	
descs	Number of seconds in one time step	[s]
numsteps	Returned number of time steps	

Non-generic versions

num_time_steps_char	calendar dates are given in string format
num_time_steps_int	calendar dates are given in integer format

read_clock

```
FUNCTION read_clock()
```

File
time_routines.f90

Type
Module function

Purpose
Returns the machine's internal clock count.

suspend_proc

```
SUBROUTINE suspend_proc(nosecswait)
INTEGER, INTENT(IN) :: nosecswait
```

File
time_routines.f90

Type
Module function

Purpose
Suspends program execution of the calling process for the given number of seconds.

Arguments

nosecswait	Wait time	[s]

update_time

```
SUBROUTINE update_time
```

File
time_routines.f90

Type
Module subroutine

Purpose
Updates the calendar date/time and a series of parameters related to date and time (e.g. the number of seconds since the start of the simulation) at the start of a new iteration in the time loop.

year_to_date

```
SUBROUTINE year_to_date(yearnum,datetime)
CHARACTER (LEN=lentime) or INTEGER, DIMENSION(7),&
                                    & INTENT(OUT) ::  datetime
REAL, INTENT(IN) :: yearnum
```

File
time_routines.f90

Type
Generic module subroutine

Purpose
Convert an absolute year number to a calendar date and time in a string or integer vector format.

Arguments

yearnum	Absolute date in years
datetime	Returned calendar date

Non-generic versions

year_to_date_char	result is returned in string format
year_to_date_int	result is returned in integer format

31.22 Routines used by the turbulence model

mom_strat_MA

```
FUNCTION mom_strat_MA(ricnum,mask)
LOGICAL, INTENT(IN), DIMENSION(ncloc,nrloc) :: mask
REAL, INTENT(IN), DIMENSION(ncloc,nrloc) :: ricnum
REAL, DIMENSION(ncloc,nrloc) :: mom_strat_MA
```

File

turbulence_routines.F90

Type

Module function

Purpose

Returns the Munk-Anderson stratification function for momentum $f_m(Ri)$ using (4.138). Result is returned at wet points only.

Arguments

ricnum	Richardson number
mask	Mask to exclude land areas

richardson_number

```
FUNCTION richardson_number(k,mask)
INTEGER, INTENT(IN) :: k
LOGICAL, INTENT(IN), DIMENSION(:,:) :: mask
REAL, DIMENSION(LBOUND(mask,1):UBOUND(mask,1),&
              & LBOUND(mask,2):UBOUND(mask,2)) :: richardson_number
```

File

turbulence_routines.F90

Type
Module function

Purpose
Returns the Richardson number Ri defined by (4.129) at the vertical level k. Result is returned at wet points only.

Arguments

k	Index of vertical level
mask	Mask to exclude land areas

richardson_number_0d

```
FUNCTION richardson_number_0d(i,j,k)
INTEGER, INTENT(IN) :: i, j, k
REAL :: richardson_number_0d
```

File
turbulence_routines.F90

Type
Module function

Purpose
Returns the Richardson number Ri defined by (4.129) at a location on the model grid.

Arguments

i	X-location on the model grid
j	Y-location on the model grid
k	Vertical location on the model grid

scal_strat_MA

```
FUNCTION scal_strat_MA(ricnum,mask)
LOGICAL, INTENT(IN), DIMENSION(ncloc,nrloc) :: mask
REAL, INTENT(IN), DIMENSION(ncloc,nrloc) :: ricnum
REAL, DIMENSION(ncloc,nrloc) :: scal_strat_MA
```

File
turbulence_routines.F90

Type
Module function

Purpose
Returns the Munk-Anderson stratification function for scalars $g_m(Ri)$ using (4.139). Result is returned at wet points only.

Arguments

ricnum	Richardson number
mask	Mask to exclude land areas

31.23 Miscellaneous utility routines

cfl_orlan

```
FUNCTION cfl_orlan(x1,x2,x3)
REAL, INTENT(IN) :: x1, x2, x3
REAL :: cfl_orlan
```

File
utility_routines.f90

Type
Module function

Purpose
Evaluate the Orlanski function $O_R(r_1, r_2, r_3)$ defined by equation (5.269).

diff_vals

```
FUNCTION diff_vals(ilist)
INTEGER, INTENT(IN), DIMENSION(:) :: ilist
LOGICAL :: diff_vals
```

File
utility_routines.f90

Type
Module function

Purpose
Returns .TRUE. if the elements of the vector array ilist are all different.

Arguments

ilist Input list

digit_number_int

```
FUNCTION digit_number_int(ival)
INTEGER, INTENT(IN) :: ival
INTEGER :: digit_number_int
```

File

utility_routines.f90

Type

Module function

Purpose

Returns the number of significant (decimal) digits needed to represent the integer number ival.

digit_number_longint

```
FUNCTION digit_number_longint(ival)
INTEGER (KIND=kndilong), INTENT(IN) :: ival
INTEGER :: digit_number_longint
```

File

utility_routines.f90

Type

Module function

Purpose

Returns the number of significant (decimal) digits needed to represent the long integer number ival.

index_position

```
FUNCTION index_position(ival,ilist)
INTEGER, INTENT(IN) :: ival
INTEGER, INTENT(IN), DIMENSION(:) :: ilist
INTEGER :: index_position
```

File
utility_routines.f90

Type
Module function

Purpose
Returns the index of the first occurrence of ival in the vector list ilist or zero if ival is not included.

least_squares_fit

```
SUBROUTINE least_squares_fit(x,y,nodat,afit,bfit,corrcoef)
INTEGER, INTENT(IN) :: nodat
REAL, INTENT(OUT) :: afit, bfit, corrcoef
REAL, INTENT(IN), DIMENSION(nodat) :: x, y
```

File
utility_routines.f90

Type
Module subroutine

Purpose
Perform a least squares on the input data (a,b).

Arguments

x	Data values on the X-axis
y	Data values on the Y-axis
nodat	Number of data values
afit	Returned slope parameter of the interpolated straight line
bfit	Returned X-value at the intersection of the straight line with the X-axis
corrcoef	Returned correlation coefficient r^2

lim_dims

```
FUNCTION lim_dims(lims)
INTEGER, INTENT(IN), DIMENSION(3) :: lims
INTEGER :: lim_dims
```

File
utility_routines.f90

Type
Module function

Purpose
Returns the number of iterations within the loop whose first, end and step values are given by respectively the first, second and third element of lims.

loop_index

```
FUNCTION loop_index(lims,iloop)
INTEGER, INTENT(IN) :: iloop
INTEGER, INTENT(IN), DIMENSION(3) :: lims
LOGICAL :: loop_index
```

File
utility_routines.f90

Type
Module function

Purpose
Returns .TRUE. if iloop is within the loop whose first, end and step values are given by respectively the first, second and third element of lims.

mult_index

```
FUNCTION mult_index(ix,iy)
INTEGER, INTENT(IN) :: ix, iy
LOGICAL :: mult_index
```

File
utility_routines.f90

Type
Module function

Purpose
Returns .TRUE. if iy is non-zero and ix an integer multiple of iy.

num_halo

```
FUNCTION num_halo(iopt_adv)
INTEGER, INTENT(IN) :: iopt_adv
INTEGER :: num_halo
```

File
utility_routines.f90

Type
Module function

Purpose
Returns the halo size needed for advection.

Arguments

iopt_adv	Type of advection scheme
	0: no advection
	1: upwind
	2: central or Lax-Wendroff
	3: TVD

outer_product

```
FUNCTION outer_product(a,b)
REAL, DIMENSION(:), INTENT(IN) :: a, b
REAL, DIMENSION(SIZE(a), SIZE(b)) :: outer_product
```

File
utility_routines.f90

Type
Module function

Purpose
Returns the outer product of the vectors a, b. The elements of the resulting matrix A are $A_{ij} = a_i b_j$.

prime_factoring

```
SUBROUTINE prime_factoring(nx,nfacs,ifacs,maxfacs,maxprime)
INTEGER, INTENT(IN) :: maxfacs, maxprime, nx
INTEGER, INTENT(OUT) :: nfacs
INTEGER, INTENT(OUT), DIMENSION(maxfacs) :: ifacs
```

File
utility_routines.f90

Type
Module subroutine

Purpose
Returns the prime number factors of an integer number.

Arguments

nx	Integer number
nfacs	Returned number of prime factors (lower or equal than maxfacs)
ifacs	Returned vector of prime numbers in descending order (lower or equal than maxprime)
maxfacs	Maximum number of prime numbers
maxprime	Largest allowed prime factor

qsort_index

```
SUBROUTINE qsort_index(arr,indx,iorder)
INTEGER, INTENT(IN) :: iorder
REAL, INTENT(IN), DIMENSION(:) :: arr
INTEGER, INTENT(OUT), DIMENSION(SIZE(arr)) :: indx
```

File
utility_routines.f90

Type
Module subroutine

Purpose
Uses the "quick-sort" algorithm to create the index array indx such that the values of arr(indx(j)) are sorted in ascending (descending) order if iorder = 1 (-1)

Reference
: Routine indexx from Press *et al.* (1992)

relax_factor

```
FUNCTION relax_factor(idist,ityp,width)
INTEGER, INTENT(IN) :: idist, ityp
REAL, INTENT(IN) :: width
REAL :: relax_factor
```

File
: *utility_routines.f90*

Type
: Module function

Purpose
: Returns the weighting factor for the relaxation scheme.

Arguments

idist	Distance of the grid point from the open boundary in grid indices
ityp	Type of weight function 1: linear form (4.389) 2: quadratic form (4.390) 3: hyperbolic form (4.391)
width	Width of the relaxation zone in grid indices

string_replace

```
SUBROUTINE string_replace(string,cin,cout)
CHARACTER (LEN=*), INTENT(INOUT) :: string
CHARACTER (LEN=1), INTENT(IN) :: cin, cout
```

File
: *utility_routines.f90*

Type
: Module subroutine

Purpose
: Replaces each occurence of character cin by the character cout in string. If the string contains blanks only, the string is replaced by cout.

swap_data

```
SUBROUTINE swap_data(var1,var2)
INTEGER, REAL or COMPLEX &
            & [, DIMENSION(:[,:])], INTENT(INOUT) ::  var1, var2
```

File

utility_routines.f90

Type

Generic module subroutine

Purpose

Exchange ("swap") the two integer, real or complex arguments. In case of integers, only scalars allowed. In case of real or complex data the arguments are either scalars, vectors or 2-D arrays.

Non-generic versions

swap_data_var_int	integer scalars
swap_data_var_real	real scalars
swap_data_var_cmplx	complex scalars
swap_data_1d_real	real vectors
swap_data_1d_cmplx	complex vectors
swap_data_2d_real	real 2-D arrays
swap_data_2d_cmplx	complex 2-D array

tvd_limiter

```
FUNCTION tvd_limiter(x,mask)
REAL, INTENT(IN), DIMENSION(:,:) :: x
LOGICAL, INTENT(IN), OPTIONAL,&
                   & DIMENSION(LBOUND(x,1):UBOUND(x,1),&
                             & LBOUND(x,2):UBOUND(x,2)[,&
                             & LBOUND(x,3):UBOUND(x,3)]) :: mask
REAL[, DIMENSION(LBOUND(x,1):UBOUND(x,1),&
               & LBOUND(x,2):UBOUND(x,2)[,&
               & LBOUND(x,3):UBOUND(x,3)])] :: tvd_limiter
```

File

utility_routines.f90

Type
Generic module function

Purpose
Evaluate the TVD limiting function $\Omega(r)$ on the model grid using either the superbee (5.52) or the monotonic (5.53) limiter. No result is returned on land areas.

Arguments

x	Argument r of the limiting function
mask	Mask to exclude land areas. Shape must be the same as the result.

Non-generic versions

tvd_limiter_0d Scalar result

tvd_limiter_2d 2-D result

tvd_limiter_3d 3-D result

two_power

```
FUNCTION two_power(n)
INTEGER, INTENT(IN) :: n
INTEGER :: two_power
```

File
utility_routines.f90

Type
Module function

Purpose
Returns the smallest number p for which $2^p \geq n$.

upper_case

```
SUBROUTINE upper_case(string)
CHARACTER (LEN=*), INTENT(INOUT) :: string
```

File
utility_routines.f90

Type
: Module routine

Purpose
: Converts a string to upper case