# Chapter 10

# Model grid and spatial interpolation

## 10.1 Model grid arrays

### 10.1.1 Array shapes

The layout of the model grid and the grid indexing system have been presented in Section 5.2.1. The shape of a model array defined on the model grid (further denoted as "model grid arrays") depends on its nodal location. If the model code would have been designed for serial (non-parallel) applications only, the shape of a model grid array could be obtained from Table 5.1. For example, the shapes of a C-node, U-node or W-node array would be (nc,nr,nz), (nc,nr,nz) or (nc,nr,nz+1), where it is recalled that nc, nr, nz are the (computational) grid dimensions in the X-, Y- and Z-direction.

Since the model can be set up either in serial or in parallel mode, two additional complexities arise:

1. The parallel code is based upon non-shared memory between the processes. This means that model grid arrays are only defined locally. The local array shape then depends on the dimensions of the local process domains. The local dimensions in the X- and Y-direction are given by the process-dependent parameters ncloc and nrloc, whereas the vertical dimension is still given by nz.

2. The solution of the discretised equations on a parallel grid requires that the domain on which an array is defined, must extend into the neighbouring domains. This extended area, called the halo area, is defined by adding extra rows and columns. The maximum size of the

halo in each of the four directions (West/East/South/North) is given by the parameter nhalo = 2. The actual size is array-dependent.

Examples of array shapes declarations are illustrated below.

```
REAL, DIMENSION(1-nhalo:ncloc+nhalo,1-nhalo:nrloc+nhalo,nz) :: sal
REAL, DIMENSION(0:ncloc+1,0:nrloc+1) :: zeta
REAL, DIMENSION(0:ncloc,0:nrloc) :: atmpres
REAL, DIMENSION(ncloc,nrloc,nz+1) :: vdifcoefscal
```

Example 10.1: examples of model grid array bounds.

The first array has a full size halo of two rows and columns in each direction, the second a halo of one row and column in each direction, the third a one column halo attached on its western boundary, whereas the last one is defined without halo.

In Section 5.2.1 it was explained that the last array column at the eastern boundary and the last row at the northern boundary of the computational grid consist of dummy land points. Important to note that this is mostly not the case for local array definitions, i.e. the horizontal grid points with index (ncloc,j) and i,nrloc are physical points unless one of the edges of the sub-domain extends into one of these two dummy boundary areas.

Local arrays usually have no global equivalent in the program. However, in case that a global array has to be defined, then it must be declared, for consistency, with the same halo sizes as its local equivalent. For example, the arrays

```
gxcoord(0:ncloc+1,0:nrloc+1), gxcoordglb(0:nc+1,0:nr+1)
```

are the local and global representations of the same array.

A more detailed account of parallel grids is presented in Chapter 11.

## 10.1.2 Parameters and arrays related to the model grid

### 10.1.2.1 definition of the model grid

The following parameters and arrays are required for the definition of the model grid:

1. The number of grid cells in the X-, Y- and vertical direction: parameters nc, nr, nz.

2. Horizontal grid

2.1 Rectangular uniform. The following attributes of the derived variable surfacegrids(igrd_model,1) are needed:

| | |
|---|---|
| x0dat | reference coordinate $x_r$ or longitude $\lambda_r$ |
| y0dat | reference coordinate $y_r$ or latitude $\phi_r$ |
| delxdat | uniform grid spacing $\Delta x$ or $\Delta\lambda$ in the X-direction |
| delydat | uniform grid spacing $\Delta y$ or $\Delta\phi$ in the Y-direction |
| rotated | .TRUE in case a rotated grid is selected |
| gridangle | grid rotation angle $\alpha$ in case of rotated grids |
| y0rot | reference latitude $\phi'_r$ in case of a rotated spherical grid |

2.2 Rectangular non-uniform.

- The following attributes of the derived variable surfacegrids(igrd_model,1) are required:

| | |
|---|---|
| x0dat | reference coordinate $x_r$ or longitude $\lambda_r$ |
| y0dat | reference coordinate $y_r$ or latitude $\phi_r$ |
| rotated | .TRUE in case a rotated grid is selected |
| gridangle | grid rotation angle $\alpha$ in case of rotated grids |
| y0rot | reference latitude $\phi'_r$ in case of a rotated spherical grid |

- The grid spacings are stored in the arrays gdelxglb(1:nc) and gdelyglb(1:nr) for respectively the X- and Y-direction.

2.3 Fully curvilinear.

- The coordinates are obtained with respect to a reference location whose position is determined by attributes of the derived variable surfacegrids(igrd_model,1)

  x0dat reference x-coordinate $x_r$ or longitude $\lambda_r$

  y0dat reference y-coordinate $y_r$ or latitude $\phi_r$
- The geographical locations of the cell corners (solid circles in Figure 5.1) with respect to the reference point are stored firstly in the arrays gxcoordglb(1:nc,1:nr) and gycoordglb(1:nc,1:nr), after which the reference coordinates are added to the arrays. The reason for this procedure is to avoid rounding errors for fine-resolution grids.

3. Vertical grid

3.1 Vertically uniform grid. This is the default case. The vertical grid, i.e. $\sigma$-coordinates of the cells corners are stored in the 3-D array gscoord(1:nc-1,1:nr-1,1:nz+1) by the program.

3.2 Vertically non-uniform, horizontally uniform.

- If the switch iopt_grid_vtype_transf is not set, the vertical array gsigcoord(1:nz+1) is supplied and stored by the program in gscoord(1:nc-1,1:nr-1,1:nz+1) at each horizontal location. Note that gscoord must be equal to 0 at the lowest and 1 at the highest cell.
- If iopt_grid_vtype_transf is set to 11, 12 or 13, the vertical grid is obtained by the program using the transformation (4.23), (4.24) or (4.26).

3.3 Vertically and horizontally non-uniform.

- If the switch iopt_grid_vtype_transf is not set, the full 3-D array gscoord(1:nc-1,1:nr-1,1:nz+1) must be given at each horizontal and vertical position. Note that the array must be equal to 0 at the lowest and 1 at the highest cell.
- If iopt_grid_vtype_transf is set to 21, the vertical grid is obtained by the program using the Song & Haidvogel (1994) transformation given by equations (4.33) and (4.35).

A list of the grid parameters and arrays defining a model grid in COHERENS is given in Table 10.1.

- Local (global) means that the parameter or array is defined on the local sub-domain (global computational grid). In case of a serial application, local and global definitions coincide (e.g. ncloc=nc)
- The bounds of the global array are obtained from the local one by replacing (nc,nr) with (ncloc,nrloc).

#### 10.1.2.2 definition of the open boundaries

The following information has to be given by the user

- the number of open sea (nosbu,nosbv) and river (nrvbu,nrvbv) boundaries at U- and V-nodes
- the location of the open boundaries at U- and V-nodes: arrays iobu(nobu), jobu(nobu), iobv(nobv), jobv(nobv) where nobu, nobv are the total (open sea and river) boundaries at U- and V-nodes

Table 10.1: Model grid parameters and arrays.

| Local name | Global name | Local array bounds | Purpose |
|---|---|---|---|
| ncloc | nc | – | number of grid cells in the X-direction |
| nrloc | nr | – | number of grid cells in the Y-direction |
| nz | nz | – | number of grid cells in the Z-direction |
| gxcoord | gxcoordglb | (0:ncloc+1,0:nrloc+1) | X-coordinates of the cell corners (UV-nodes) in meters or degrees longitude (between -$180^0$ and $180^0$) |
| gycoord | gycoordglb | (0:ncloc+1,0:nrloc+1) | Y-coordinates of the cell corners (UV-nodes) in meters or degrees latitude (between -$90^0$ and $90^0$) |
| gscoord | gscoordglb | (0:ncloc+1,0:nrloc+1, nz+1) | $\sigma$-level coordinate at the W-nodes. |

The following remarks apply:

- At a U- or V-node open boundary one of the two adjacent cells must be wet (sea) while the other one must be either a land cell or located outside the physical domain. This allows to define linear as well as "ragged" (stair-case) open boundaries.

- Open sea boundaries at U-nodes must be stored in iobu(1:nosbu), jobu(1:nosbu), river boundaries in iobu(nosbu+1:nosbu+nrvbu), jobu(nosbu+1:nosbu+nrvbu). A similar procedure is taken at V-nodes.

- Taking account of the previous restriction, the order of storage is irrelevant. However, the ordering is of importance for the definition of the open boundary conditions (see Chapter 16) since it introduces an indexing system, i.e. ii is (e.g.) the index of the U-open boundary located at (iobu(ii), jobu(ii)).

Besides U- and V-open boundaries the program also uses internally the concept of X- and Y-open boundaries which are defined as follows:

- A corner node is a X-open boundary if one of the adjacent U-nodes is an open boundary and the other one a land/coastal or open sea boundary.

Table 10.2: Open boundary parameters and arrays.

| Local name | Global name | Local array bounds | Purpose |
|---|---|---|---|
| nosbuloc | nosbu | – | number of West/East open sea boundaries at U-nodes |
| nrvbuloc | nrvbu | – | number of West/East river open boundaries at U-nodes |
| nobuloc | nobu | – | total number of West/East open boundaries at U-nodes (nobuloc = nosbuloc+nrvbuloc) |
| nosbvloc | nosbv | – | number of South/North open sea boundaries at V-nodes |
| nrvbvloc | nrvbv | – | number of South/North river open boundaries at V-nodes |
| nobvloc | nobv | – | total number of South/North open boundaries at V-nodes (nobvloc = nosbvloc+nrvbvloc) |
| nobxloc | nobx | – | number of X-open boundaries at corner nodes |
| nobyloc | noby | – | number of Y-open boundaries at corner nodes |
| iobuloc | iobu | nobuloc | X-indices of the West/East open boundaries at U-nodes |
| jobuloc | jobu | nobuloc | Y-indices of the West/East open boundaries at U-nodes |
| iobvloc | iobv | nobvloc | X-indices of the South/North open boundaries at V-nodes |
| jobvloc | jobv | nobvloc | Y-indices of the South/North open boundaries at V-nodes |
| iobxloc | iobx | nobxloc | X-indices of the open boundaries at X-nodes |
| jobxloc | jobx | nobxloc | Y-indices of the open boundaries at X-nodes |
| iobyloc | ioby | nobyloc | X-indices of the open boundaries at Y-nodes |
| jobyloc | joby | nobyloc | Y-indices of the open boundaries at Y-nodes |

- A corner node is a Y-open boundary if one of the adjacent V-nodes is an open boundary and the other one a land/coastal or open sea boundary.

The locations of the X- and Y-open boundaries are defined within the program and do not need to be supplied by the user.

A list of all grid parameters and arrays is given in Table 10.2.

- Local (global) means that the parameter or array is defined on the local sub-domain (global computational grid). In case of a serial application, local and global definitions coincide (e.g. nosbuloc=nosbu).
- The bounds of the global array are obtained from the local one by replacing the local dimension by the global one (e.g. nobuloc by nobu).

#### 10.1.2.3 grid spacings

Grid spacings in the horizontal direction are calculated by the standard formulae

$$\begin{aligned} \Delta X^v_{ij} &= h^v_{1;ij} = \sqrt{(x_{i+1,j} - x_{ij})^2 + (y_{i+1,j} - y_{ij})^2} \\ \Delta Y^u_{ij} &= h^u_{2;ij} = \sqrt{(x_{i,j+1} - x_{ij})^2 + (y_{i,j+1} - y_{ij})^2} \end{aligned} \tag{10.1}$$

or

$$\begin{aligned} \Delta X^v_{ij} &= h^v_{1;ij} = R\sqrt{(\lambda_{i+1,j} - \lambda_{ij})^2 + \cos\frac{1}{2}(\phi_{ij} + \phi_{i+1,j})(\phi_{i+1,j} - \phi_{ij})^2} \\ \Delta Y^u_{ij} &= h^u_{2;ij} = R\sqrt{(\lambda_{i,j+1} - \lambda_{ij})^2 + \cos\frac{1}{2}(\phi_{ij} + \phi_{i,j+1})(\phi_{i,j+1} - \phi_{ij})^2} \end{aligned} \tag{10.2}$$

depending on whether Cartesian or spherical coordinates are used. Similar expressions are used to evaluate the grid spacings at other nodes. Grid spacings in the vertical are calculated using $h_3 = H\Delta\sigma$ where $H$ is the total water depth and $\Delta\sigma$ the $\sigma$-difference between two neighbouring levels.

The following grid spacing arrays are defined:

- grid spacing $h_1$ in meters

  `delxatc, delxatu, delxatv, delxatuv`

- grid spacing $h_2$ in meters

  `delyatc, delyatu, delyatv, delyatuv`

- $\sigma$-grid spacing $\Delta\sigma = h_3/H$

`delsatc, delsatu, delsatv, delsatw, delsatuw, delsatvw`

The last letter(s) in the variable name denote(s) the grid node where the array is defined.

#### 10.1.2.4 pointer arrays

Pointer arrays denote the status of a grid node (dry, wet, open boundary). A specific array is defined for each horizontal nodal type

nodeatc status at C-nodes

- 0: inactive cell which can either by a permanently land cell or a sea cell temporarily set to dry by a mask criterium in case a flooding scheme is applied (see Section 5.4.2)
- 1: active (wet) sea cell

nodeatu Pointers at U-node cell faces

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

nodeatv Pointers at V-node cell faces

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

nodeatuv Pointer at corner (UV) nodes

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

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

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

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

nodeatuw Pointer at UW-node cell faces

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

1: coastal boundary

2: interior wet UW-node

3: open sea boundary

4: river open boundary

nodeatvw Pointer at VW-node cell faces

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

1: coastal boundary

2: interior wet VW-node

3: open sea boundary

4: river open boundary

- The values of the pointer arrays are obtained from the bathymetry (zero water depths at land, positive depths at sea) and the positions of the open boundaries as given by the user.
- In the current implementation the arrays are fixed in time. Dynamic masks are foreseen in future versions.
- Local bounds in the horizontal are given by (1-nhalo:ncloc+nhalo, 1-nhalo:nrloc+nhalo).
- All arrays, except nodeatc, have a vertical dimension of nz at U-, V- or UV-nodes and nz+1 at UW- or VW-nodes. The vertical dimension has been added for the future implementation of structures.
- The bottom and surface values of nodeatuw, nodeatvw are, by definition, always equal to 1.
- In the current implementation all arrays must be vertically uniform (with exception of the previous restriction for UW- and VW-nodes).

## 10.2 Interpolation of model arrays at a different node

Since the model uses an Arakawa C-grid, arrays often have to be interpolated from one grid node to another one. The simplest way is to consider uniform interpolation between neighbouring points. The following issues need to be taken into consideration:

- Uniform averaging is no longer applicable (or at least recommended) in case of a curvilinear grid.
- Model grid arrays are undefined at land points. Land mask can be used to eliminate these points from the interpolation.

These points will be addressed in the discussion below.

### 10.2.1 Interpolation without land flags

The general formula for interpolation a quantity X defined at node "a" to node "b" is

$$X^b(x,y,z) = \frac{\sum_{n=1}^{N} w_n X^a(x_n,y_n,z_n)}{\sum_{n=1}^{N} w_n} \tag{10.3}$$

where $w_n$ are grid-dependent weight factors. In case of a uniform interpolation all $w_n = 1/N$ so that

$$X^b = \frac{1}{N}\sum_{n=1}^{N} X^a(x_n,y_n,z_n) \tag{10.4}$$

The parameter $N$ depends on the number of grid dimensions involved in the interpolation:

- N=2: interpolation along a coordinate line (1 direction). In case of non-uniform interpolation the weights factors have dimensions of lengths.
- N=4: interpolation within a coordinate surface (2 directions). In case of non-uniform interpolation the weight factors have dimensions of areas.
- N=8: 3-D interpolation (3 directions). In case of non-uniform interpolation the weight factors have dimensions of volumes.

General rules are

- Interpolations within a grid cell are always uniform.

- By default, interpolation involving values at neighbouring cells are uniform. This can be overruled with the switch iopt_array_interp or by calling the interpolation routines with the optional hregular and vregular argument set to .FALSE.. The second option always takes precedence over the first.

- The interpolation routines in the program provide an option to exclude land points or open sea boundaries from the interpolation.

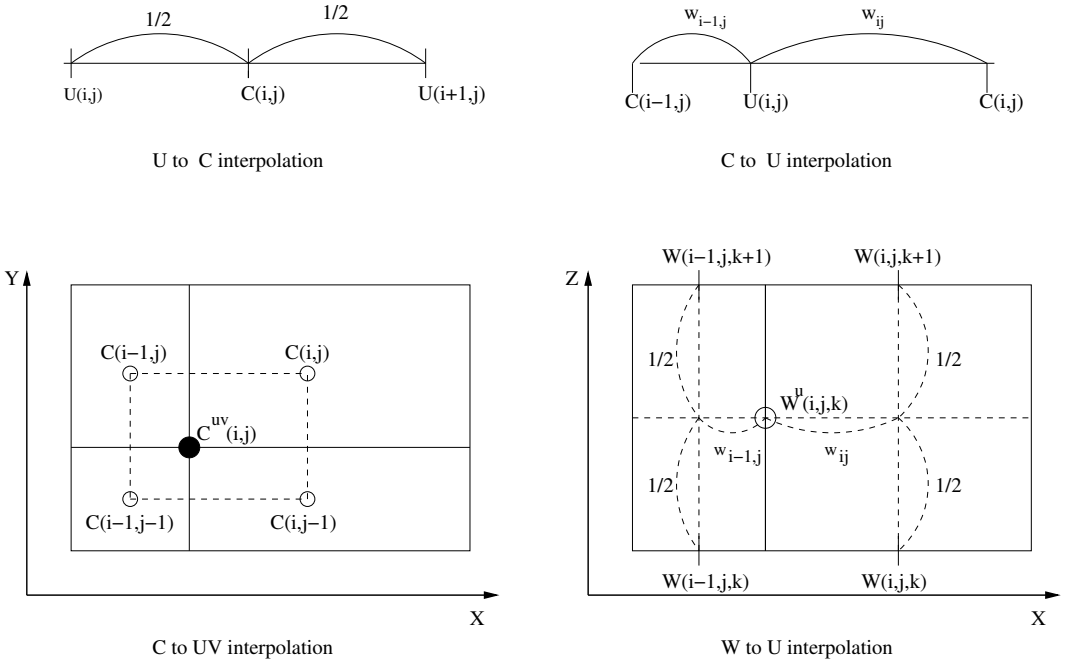


Figure 10.1: Interpolation of model grid arrays at a different node.

Examples

1. interpolation from U- to C-node:

$$X^c(i,j) = \frac{1}{2}(X^u(i,j) + X^u(i+1,j)) \tag{10.5}$$

2. (non-uniform) interpolation from C- to U-node:

$$X^u(i,j) \quad = \quad \frac{w(i,j)X^c(i-1,j) + w(i-1,j)X^c(i,j)}{w(i-1,j) + w(i,j)}$$

$$w(i,j) = \frac{1}{2}h_1^c(i,j) \tag{10.6}$$

3. (non-uniform) interpolation from C- to UV-node:

$$\begin{aligned} X^{uv}(i,j) &= \Big(w(i,j)X^c(i-1,j-1) + w(i-1,j)X^c(i,j-1) \\ &\quad + w(i,j-1)X^c(i-1,j) + w(i-1,j-1)X^c(i,j)\Big) \\ &\quad \Big/\Big(w(i,j) + w(i-1,j) + w(i,j-1) + w(i-1,j-1)\Big) \\ w(i,j) &= \frac{1}{4}h_1^c(i,j)h_2^c(i,j) \end{aligned} \tag{10.7}$$

4. (non-uniform) interpolation from W- to U- node:

$$\begin{aligned} X^u(i,j,k) &= \frac{1}{2}\Big(w(i,j)X^w(i-1,j,k) + \\ &\quad w(i-1,j)X^w(i,j,k) + w(i,j)X^w(i-1,j,k+1) \\ &\quad + w(i-1,j)X^w(i,j,k+1)\Big)\Big/\Big(w(i,j) + w(i-1,j)\Big) \\ w(i,j) &= \frac{1}{2}h_1^c(i,j) \end{aligned} \tag{10.8}$$

What type of interpolation needs to be taken ?

- uniform Cartesian grids: uniform (allways)
- uniform spherical: uniform (recommended)
- non-uniform rectangular and curvilinear: user decision (depending on the variation of grid size between neighbouring cells)

### 10.2.2 Interpolation with land flags

Land flags can be used to eliminate land areas or temporarily inactive cells (in case a mask function is applied as decribed in Section 5.4.2) from the interpolation. The general interpolation formula with land flags becomes

$$X^b(x,y,z) = \frac{\sum_{n=1}^{N} s_n w_n X^a(x_n,y_n,z_n)}{\sum_{n=1}^{N} s_n w_n} \tag{10.9}$$

where $s_n$ equals 0 for flagged and 1 for non-flagged grid points. The program foresees the following options

0: No flags ($s_n$=1 everywhere).

1: Land cells and cell faces at or near coastal boundaries (except open sea boundaries) are flagged.

2: Land cells and cell faces at or near coastal boundaries (including open boundaries) are flagged.

3: Only cell faces at open sea boundaries are flagged (allowing the inclusion of coastal boundaries in the interpolation).

As an example the formula for array interpolation from C- to UV-nodes becomes

$$\begin{aligned} X^{uv}(i,j) &= \Big(s(i-1,j-1)w(i,j)X^c(i-1,j-1) \\ &\quad +s(i,j-1)w(i-1,j)X^c(i,j-1) \\ &\quad +s(i-1,j)w(i,j-1)X^c(i-1,j) \\ &\quad +s(i,j)w(i-1,j-1)X^c(i,j)\Big) \Big/ \\ &\quad \Big(s(i-1,j-1)w(i,j)+s(i,j-1)w(i-1,j) \\ &\quad +s(i-1,j)w(i,j-1)+s(i,j)w(i-1,j-1)\Big) \\ w(i,j) &= \frac{1}{4}h_1^c(i,j)h_2^c(i,j) \end{aligned} \tag{10.10}$$

Note that the flags can be applied at the source node "a" (as given in the equation above) as well as at the destination node "b".

## 10.3 Curvilinear, index and relative coordinates

Before proceeding to discuss the interpolation towards or from an external data grid, some more detailed discussion is required how the model's curvilinear coordinates are related to the grix index system introduced in Section 5.2.1.

Consider the grid layout as shown in Figure 10.2. The index (i,j) of a UV-nodal grid point can be interpreted as the curvilinear coordinates ($\xi_1$, $\xi_2$) with respect to axes in the X- and Y-direction and origin at corner index (0,0). Within the same frame of reference, the corresponding curvilinear coordinates of the C-point with index (i,j) are (i+1/2,j+1/2). For U- and V-nodal points this becomes (i,j+1/2) and (i+1/2,j).

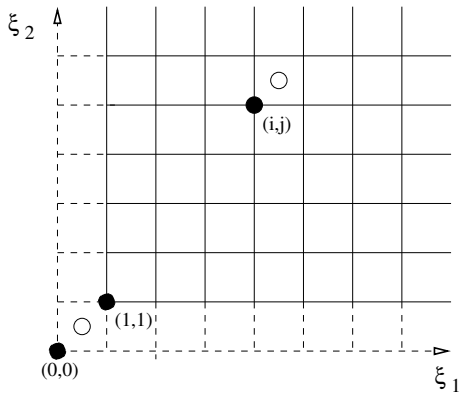


Figure 10.2: Curvilinear versus index coordinates.

Instead of taking the origin at the corner point with index (0,0), the origin can be moved to the C-point with index (0,0). In that case, the curvilinear coordinates of a corner point at (i,j) are (i-1/2,j-1/2), while the curvilinear coordinates of a C-grid point are the same as its indices. For U- and V-nodal points this becomes (i-1/2,j) and (i,j-1/2). Similarly, if the origin is taken at the U-node index point (0,0), the curvilinear coordinates of a C-point, U-point, V-point and corner point are respectively (i+1/2,j), (i,j), (i+1/2,j-1/2) and (i,j-1/2). Finally, taking the origin at the point with V-node index (0,0), the curvilinear coordinates becomes (i,j+1/2) for a C-node, (i-1/2,j+1/2) for a U-node,(i,j) for a V-node and (i-1/2,j) for a corner point.

Relative coordinates are a convenient way to perform interpolation from one to an other. Let ($\xi_1$,$\xi_2$) be the (normalised) curvilinear coordinates of an arbitrary point, the corresponding relative coordinates are then (i,j,x,y) where (i,j) are the integer and (x,y) the decimal parts of ($\xi_1$,$\xi_2$). From the previous discussion, the integer coordinates are taken with respect to a certain node (C, U, V, UV). For example, if a corner index system is taken, the relative coordinates of a C-point with index (i,j) with respect to this corner grid are (i,j,1/2,1/2).

## 10.4 Interpolation of a 2-D external data grid at the model grid

### 10.4.1 General description of the procedure

Assume that the model grid is embedded within an external 2-D data grid as shown in Figure 10.3. The relative coordinates of the C-grid point X

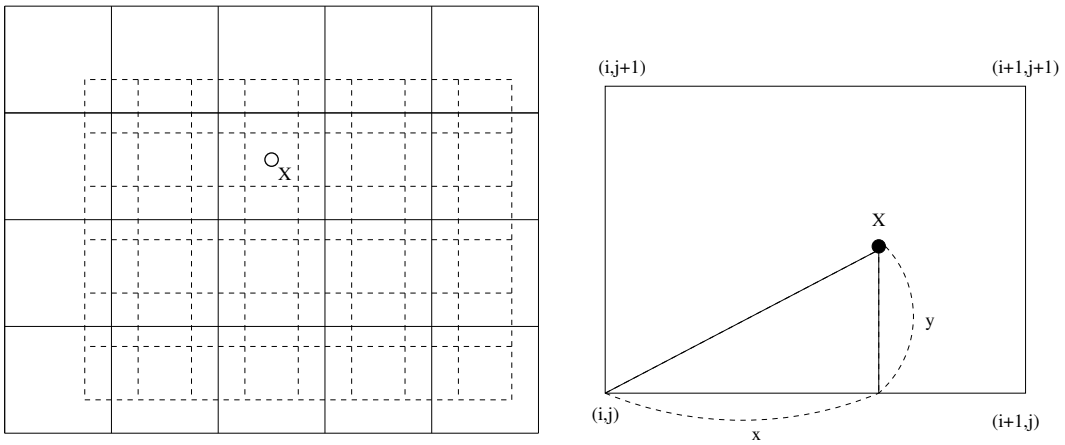


Figure 10.3: Interpolation of external data to the model grid. The solid (dashed) lines represent the external (model) grid.

with respect to the data grid at its cell corners are (i,j,x,y) where (x,y) are expressed as normalised coordinates (i.e. between 0 and 1). The interpolated value of data 'd' at X is given by

$$\begin{aligned} d(X) &= (1-x)(1-y)d(i,j) + x(1-y)d(i+1,j) \\ &\quad +(1-x)yd(i,j+1) + xyd(i+1,j+1) \end{aligned} \tag{10.11}$$

where i, j are the indices defined on the data grid.

The following remarks apply

- The model points are principally taken at C-nodes, although the method can be applied to other nodes as well.
- If the external grid is rectangular (uniform or non-uniform), the relative coordinates can be calculated by the program. No algorithm is currently implemented for the determination of the relative coordinates in case of a curvilinear or unstructured data grid.
- The weight factors in (10.11) only depend on the locations of the external grid and do not take account of possible land points, which should be excluded from the interpolation. The following options are available:
  1. All land points are taken into account.
  2. Interpolation is only performed if at least one of the surrounding four points is located at sea.

3. Interpolation is only performed if all the four surrounding points are at sea.

If the data grid represents meteorological data, no disctinction is made between land and sea points.

### 10.4.2 Implementation

The method can currently be applied for interpolation of surface meteorological data or (satellite) surface temperature data. Future extensions are planned for surface wave data.

As a first step the type of the 2-D external data grid is defined via the following derived type definition:

```
TYPE :: GridParams
   INTEGER :: nhtype, n1dat, n2dat
   REAL :: delxdat, delydat, x0dat, y0dat
END TYPE GridParams
```

where

nhtype Type of the surface data grid.

- 0: single grid point
- 1: uniform rectangular grid
- 2: non-uniform rectangular grid
- 3: non-rectangular (curvilinear or non-structured) grid
- 4: the same as the model grid

n1dat number of grid cells in the X-direction

n2dat number of grid cells in the Y-direction

delxdat grid spacings in the X-direction (m or fractional degrees longitude) when nhtype=1

delydat grid spacings in the Y-direction (m or fractional degrees latitude) when nhtype=1

x0dat X-coordinate (Cartesian or longitude) of the lower left corner when nhtype=1

y0dat Y-coordinate (Cartesian or latitude) of the lower left corner when nhtype=1

Note that nhtype has to be supplied always if the value is different from its default one (0). The n1dat, n2dat attributes have to be given if nhtype$>$ 0.

All surface grid attributes are stored into the array surfacegrids

```
TYPE (GridParams), DIMENSION(MaxGridTypes,2) :: surfacegrids
```

where MaxGridTypes is the maximum number of external grids. A key id of the form igrd_* is used as index for the first dimension. The following values are currently implemented

- igrd_meteo: surface meteorological grid
- igrd_sst: sea surface temperature (SST) grid
- igrd_waves: surface wave grid
- igrd_model: model grid

The last id seems to imply that the model grid itself is considered as an external data grid which is obviously not the case. It has only been implemented as an utility in case the user wants to define a rectangular uniform model grid (see Section 14.6.1).

The second index can currently only take the value of 1 which means interpolation to the model grid. The value 2 is intended for interpolation of model data to the data grid which is currently not implemented.

The next step consists in determining the relative coordinates of all model grid points located at sea. Use is made of the derived type definition

```
TYPE :: HRelativeCoords
   INTEGER :: icoord, jcoord
   REAL :: xcoord, ycoord
END TYPE HRelativeCoords
```

For example, in case of surface meteorological data, all relative coordinates are stored in the array

```
TYPE (HRelativeCoords), DIMENSION(ncloc,nrloc) :: meteogrid
```

The relative coordinates itself can be determined by one of the following procedures depending on the value of the nhtype attribute

0 : The external grid reduces to one data point. No interpolation is required.

1 : The external data grid is rectangular and uniform. The relative coordinates are determined by the program.

2 : The external data grid is rectangular and non-uniform. The geographical (Cartesian or spherical) coordinates of the data grid are supplied by the user. The relative coordinates are calculated by the program.

3 : The external data grid is curvilinear or unstructured. No algorithm is provided by the program. The relative coordinates have to be supplied by the user.

4 : The data grid coincides with the model grid. No interpolation is required.

The user procedure for defining the input of 2-D external forcing data, is then the following:

1. The following parameters are defined by the user in usrdef_mod_params:
   - Set the appropriate switch which is iopt_meteo for meteo input or iopt_temp_sbc for SST input.
   - Define the attributes of the external grid(s) in surfacegrids.
   - The attributes of the following files have or may be defined:

     modfiles(io_metgrd,1,1): grid locations of the meteorological grid in case nhtype>1

     modfiles(io_sstgrd,1,1) : grid locations of the SST grid in case nhtype>1

     modfiles(io_wavgrd,1,1): grid locations of the surface wave grid in case nhtype>1

     modfiles(io_metsur,1,1): meteorological data file

     modfiles(io_sstsur,1,1) : SST data file

     modfiles(io_wavsur,1,1): wave data file

2. If nhtype>0, define the locations of the external grid. Two options are available depending on the value of nhtype:
   - The grid locations are obtained in absolute (geograhical) coordinates by calling the user defined routines usrdef_surface_absgrd. The program converts the absolute coordinates into relative ones.
   - The grid locations are obtained in relative coordinates by calling the user defined routines usrdef_surface_relgrd.

3. Define data input by calling the user-defined routines usrdef_surface_data.

Note that a usrdef_* routine is not called if the corresponding status attribute of the associated file is set to 'R' in which case a corrsponding read_* routine is called where the data are read from a file in standard COHERENS format.

Detailed descriptions of the procedures are given in Chapter 14 and Section 17.2.

## 10.5 Interpolation of model data at external locations

### 10.5.1 General description of the procedure

The procedure is similar to the one discussed in the previous section, except that the roles of the external grid and model grid are interchanged. This type of interpolation is currently only implemented for nesting procedures, but may be used in future versions for the development of one- or two-way coupling with meteorological and surface wave models.

Advantage now is that the external data points no longer need to be located on some external (structured or unstructured ) "data" grid. The only information needed by the program is the location, i.e. the relative coordinates, of the external locations with respect to the model grid.

However, there are additional complexities which need to be taken into account:

- Since the model uses a staggered C-grid, currents and scalar quantities are calculated at different locations (nodes). This means that the relative coordinates need to be defined in general with respect to different curvilinear coordinate origins (C(0,0),U(0,0),V(0,0)), whereby the type of origin depends on the type of variable to be interpolated (C(0,0) for C-node quantities and U(0,0), V(0,0) for vector quantitities).

- Model data located on land should be eliminated from the interpolation.

- Interpolation of 3-D quantities requires that the relative coordinates must contain a vertical dimension as well.

- The program allows to define multiple nested sub-grids within the same main grid.

A horizontal layout of the main and sub-grid with the positions of all points used in the interpolation on the two grids is displayed in Figure 10.4.

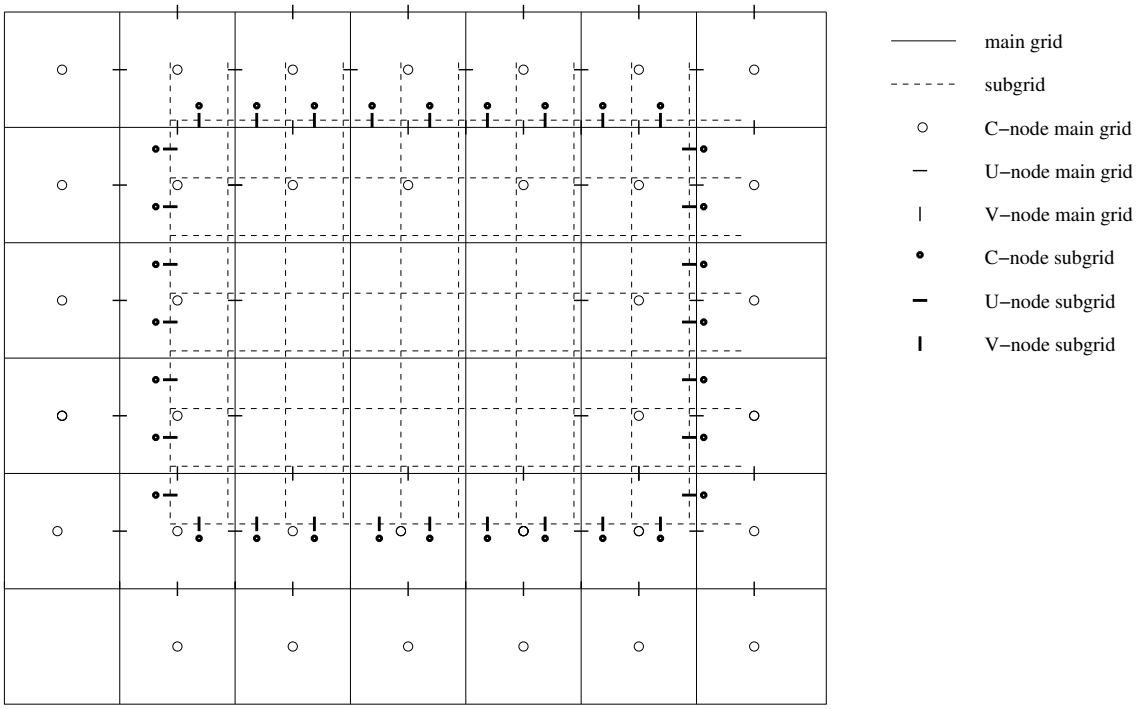


Figure 10.4: Illustration of the nesting procedure in the horizontal plane. Solid lines mark the main (coarser) grid, dashed lines the (finer) sub-grids. The empty circles and (light colour) line segments mark the points on the main grid used in the interpolation, the filled circles and (thick) line segments the points on the sub-grid to which interpolation needs to be performed.

Interpolation can be performed on the 2-D variables $U$, $V$, $\zeta$ and the 3-D variables $\delta u$, $\delta v$, $T$, $S$. The number and type of data depends on the type of open boundary conditions used by the sub-grid and the dimensions (2-D or 3-D) of the two grids. The only restriction, imposed by the program, is that, although the model permits to use elevation data either at external C-nodes (the solid circles in Figure 10.4 or at the velocity nodes on the open boundaries itself (shown by the horizontal and vertical thick line segments in the figure), only the latter form is allowed when the open boundary data are derived from interpolation. The following five kinds of interpolation then have to be considered in general.

1. Interpolation of U-node model values on the main grid to U-node open boundary points on the sub-grid (U-to-U interpolation applied for $U$ and $\delta u$)

2. Interpolation of V-node model values on the main grid to V-node open

boundary points on the sub-grid (V-to-V interpolation applied for $V$ and $\delta v$)

3. Interpolation of C-node model values on the main grid to U-node open boundary points on the sub-grid (C-to-U interpolation applied for $\zeta$ only )

4. Interpolation of C-node model values on the main grid to V-node open boundary points on the sub-grid (C-to-V interpolation applied for $\zeta$ only)

5. Interpolation of C-node model values on the main grid to C-node open boundary points (C-to-C interpolation applied for $T$ and $S$).

For each type of interpolation relative coordinates have to be defined with different nodal type and origin.

In case of a 3-D quantity ($\delta u$, $\delta v$, $T$, $S$), the C-to-C, U-to-U and V-to-V interpolation must extend to the vertical direction as well. this is achieved by adding the vertical relative coordinates k, z to the four horizontal ones (i, j, x, y) where k is the vertical index level of the W-, UW- or VW-node point just below the data point at the which interpolation has to be performed and z is (here) the normalised vertical distance (between 0 and 1) to the W-, UW-, VW-node level. The definition is illustrated in Figure 10.5.

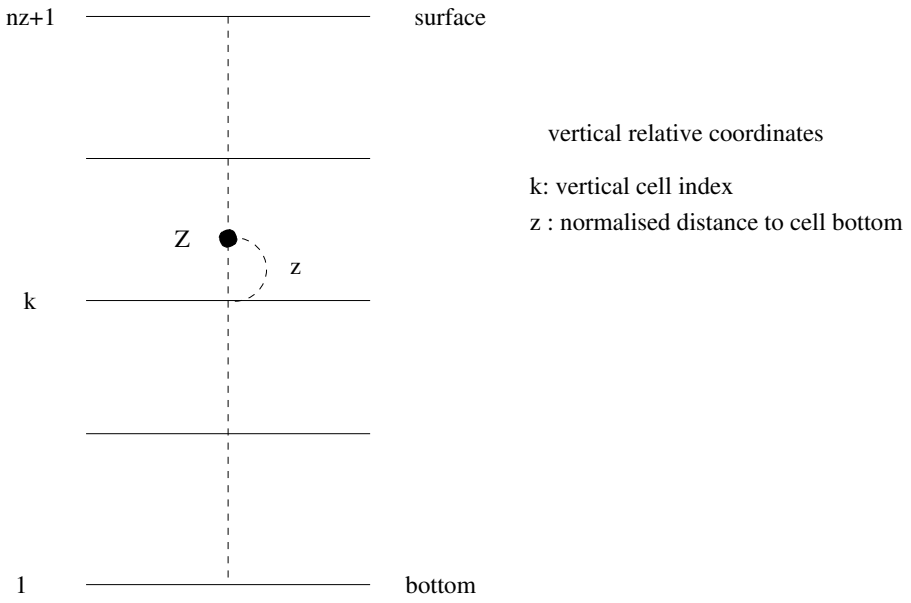


Figure 10.5: Definition of vertical relative coordinates.

The following remarks are to be given

- Interpolation does not take account of variations in water depth, i.e. the vertical position at which interpolation is to performed, is assumed to be independent in time.
- Vertical interpolation is performed prior to horizontal interpolation. This means that the model data at the four surrounding main grid points are firstly interpolated vertically at the same level as the destination point. If a depth point is below or above the $z$-level of the lowest or highest grid level, the interpolated value is set to the model value at the lowest or highest level. The profile at the sub-grid location is next obtained by horizontal interpolation of the four profiles using (10.11).

### 10.5.2 Implementation

The program provides the following derive type definition for storing vertical relative coordinates

```
TYPE :: VRelativeCoords
   INTEGER :: kcoord
   REAL :: zcoord
END TYPE VRelativeCoords
```

Details of the code implementation are more complex than the ones in Section 10.4.2. To simplify the discussion below, it is assumed that the program is set up in serial mode.

1. The following parameters are defined by the user in usrdef_mod_params:
   - The program switch iopt_nests is set to 1.
   - The parameter nonestsets is set to the number of nested sub-grids.
   - The attributes of the following files have or may be defined:

     modfiles(io_nstspc,1,1) parameters to be defined in usrdef_nstgrd_spec

     modfiles(io_nstgrd,1:nonestsets,1) locations of the sub-grid open boundary points in absolute or relative coordinates for each sub-grid

     modfiles(io_2uvnst,1:nonestsets,2) output data files with interpolated values of $U$, $V$, $\zeta$ for each sub-grid

     modfiles(io_3uvnst,1:nonestsets,2) output data files with interpolated values of $\delta u$, $\delta v$

modfiles(io_salnst,1:nonestsets,2) output data files with interpolated values of $S$

modfiles(io_tmpnst,1:nonestsets,2) output data files with interpolated values of $T$

modfiles(io_sednst,1:nonestsets,2) output data files with interpolated values of sediments

Note that output will be written only when the status attribute of the respective output data file is set to 'W'.

2. The following arrays are defined in usrdef_nstgrd_spec

```
INTEGER, DIMENSION(nonestsets) :: nohnstglbc, nohnstglbu, &
                                  & nohnstglbv, novnst, inst2dtype
INTEGER, DIMENSION(nonestsets) :: nosednst
INTEGER, DIMENSION(maxsedvars,nonestsets) :: instsed
                                  & nohnstglbv, novnst, inst2dtype
```

The first three arrays represent, for each sub-grid, the number of sub-grid open boundary points at C-, U- and V-nodes. The fourth gives the number of vertical levels for each sub-grid and inst2dtype selects the type of data for 2-D nesting. The arrays nosednst, instsed inform the program how many and which fractions are used for the nesting of sediment concentrations. For details see Sections 17.3.1 and 19.4.

3. The locations of the open boundary locations are determined for each sub-grid. The relative coordinates for each type of interpolation are stored in the following derived type arrays

```
TYPE (HRelativeCoords), DIMENSION(numhnstc) :: hnstctoc
TYPE (HRelativeCoords), DIMENSION(numhnstu) :: hnstutou, hnstctou
TYPE (HRelativeCoords), DIMENSION(numhnstv) :: hnstvtov, hnstctov
TYPE (VRelativeCoords), DIMENSION(2,2,numhnstc,numvnst) :: vnstctoc
TYPE (VRelativeCoords), DIMENSION(2,2,numhnstu,numvnst) :: vnstutou
TYPE (VRelativeCoords), DIMENSION(2,2,numhnstv,numvnst) :: vnstvtov
```

where

```
numhnstc = SUM(nohnstglbc); numhnstu = SUM(nohnstglbu)
numhnstv = SUM(nohnstglbv); numvnst = MAXVAL(novnst)
```

These locations are to be defined in the user. Two options are available:

- The user supplies, within usrdef_nstgrd_abs, the horizontal positions in absolute (geographical) and the vertical positions, taken as the positive distance from the mean sea level. The relative coordinate arrays hnstctoc, ..., vnstvtov are determined by the program.
- The user supplies, within usrdef_nstgrd_rel, the horizontal positions in relative coordinates and the vertical positions, taken as the positive distance from the mean sea level. The horizontal coordinates are stored in hnstctoc, hnstctou, hnstctov, hnstutou, hnstvtov, the vertical relative coordinates are calculated by the program and stored in vnstctoc, vnstutou, vnstvtov.

4. The following "index mapping" arrays are defined by the program

```
INTEGER, DIMENSION(noprocs,numhnstc,nonestsets) :: indexnstc
INTEGER, DIMENSION(noprocs,numhnstu+numhnstv,nonestsets) :: indexnstuv
```

   where

   | | |
   |---|---|
   | noprocs | number of processes (1 in serial mode) |
   | indexnstc | projects the local index of a C-node data point onto a "global" index over all sub-domains |
   | indexnstuv | projects the local index of a U-node or V-node data point onto a "global" index over all sub-domains |

5. Model data are interpolated and written to the appropriate output file. Time resolution is determined by the tlims file attribute.

Note that a usrdef_* routine is not called if the corresponding status attribute of the associated file is set to 'R' in which case a corrsponding read_* routine is called where the data are read from a file in standard COHERENS format.

Detailed descriptions of the procedures are given in Chapter 14 and Section 17.3.