



Estuary, Lake and Coastal Ocean Model: ELCOM

v2.2 User Manual

Ben Hodges and Chris Dallimore.

Centre for Water Research
University of Western Australia.
February 8, 2007

Contents

List of Figures	vii
List of Tables	ix
1 Introduction	1
1.1 The ELCOM model	1
1.2 Implementation language	1
1.3 Outline of a simulation run	1
1.4 Conventions and definitions	2
2 Pre-Processor	3
2.1 Introduction	3
2.2 Setting up the bathymetry file (bathymetry.dat)	3
2.2.1 Introduction	3
2.2.2 Physical and computational space coordinates	4
2.3 Setting up the boundary cell set file (bc.dat)	14
2.3.1 Introduction	14
2.3.2 Boundary cell set categories	15
2.3.3 Boundary cell set face direction	15
2.3.4 Boundary cell set keywords	15
2.3.5 Organization of the boundary cell set file (bc.dat)	16
2.3.6 flow_multi_sides	18
2.4 Setting up the levee file (levee.dat) (Optional)	19
2.4.1 Introduction	19
2.4.2 Levee set categories	19
2.4.3 Organization of the levee set file (levee.dat)	19
2.5 Setting up the update file (update.dat) (Optional)	19
2.5.1 Introduction	19
2.5.2 Update set categories	20
2.5.3 Organization of the update set file (update.dat)	20
2.6 Running the pre-processor with <code>run_pre.dat</code>	20
3 ELCOM Operation	21
3.1 Introduction	21

3.2	Files for ELCOM	21
3.3	Simulation modules	21
3.3.1	Stratification	21
3.3.2	Tracers	21
3.3.3	Surface thermodynamics	21
3.3.4	Rain	22
3.3.5	Wind	22
3.3.6	Inflow/outflow	22
3.3.7	Conservation of Background Potential Energy	22
3.3.8	Bubble Plume Destratification	23
3.3.9	Underflow	23
3.3.10	Water Age Modelling	23
3.3.11	Drifters / Particles	23
3.3.12	Jets/Pumps	23
3.3.13	Non-hydrostatic code	23
3.4	Initialization	24
3.4.1	Uniform initial conditions	24
3.4.2	Vertical Profile	24
3.4.3	Multiple Vertical Profiles	24
3.4.4	Horizontal Variation	25
3.4.5	User subroutines	25
3.5	Configuration controls	25
3.5.1	Time step	25
3.5.2	Number of steps	26
3.5.3	Start date	26
3.5.4	CGM tolerance	26
3.5.5	Turbulence	26
3.5.6	Output frequency	26
3.6	Configuration using file <code>run_elcom.dat</code>	26
3.7	Input files	32
3.7.1	<code>sparsedata.unf</code> and <code>usedata.unf</code>	32
3.7.2	Temporal boundary condition (1D) files	32
3.7.3	Temporal profile boundary condition (2D) files	34

- 3.7.4 Update boundary condition files 34
- 3.7.5 Temporal filter constant files 34
- 3.7.6 Drifter files 34
- 3.7.7 Output definition file (Datablock.db) 38
- 3.8 Running ELCOM 42
- 3.9 Post-Processing 42
 - 3.9.1 Conversion of datablock file to NETCDF 42
 - 3.9.2 Save and Restart output files 43
- 3.10 ELCOM Valid Scalars 43

List of Figures

2.1	ELCOM Coordinate System	5
2.2	Plan view of Cartesian mesh	5
2.3	Plan view of bathymetry data	6
2.4	Elevation view of bathymetry data	7
2.5	Calculation of the north vector	9
2.6	Vertical grid spacings	13

List of Tables

2.1	ELCOM Boundary Condition Keywords.	16
3.1	Files for ELCOM.	22
3.2	Temporal Boundary Condition Data-types	33

Introduction

1.1 The ELCOM model

ELCOM (Estuary and Lake Computer Model) is a numerical modeling tool that applies hydrodynamic and thermodynamic models to simulate the temporal behavior of stratified water bodies with environmental forcing. The hydrodynamic simulation method solves the unsteady, viscous Navier-Stokes equations for incompressible flow using the hydrostatic assumption for pressure. Modeled and simulated processes include baroclinic and barotropic responses, rotational effects, tidal forcing, wind stresses, surface thermal forcing, inflows, outflows, and transport of salt, heat and passive scalars. Through coupling with the CAEDYM (Computational Aquatic Ecosystem Dynamics Model) water quality module, ELCOM can be used to simulate three-dimensional transport and interactions of flow physics, biology and chemistry. The hydrodynamic algorithms in ELCOM are based on the Euler-Lagrange method for advection of momentum with a conjugate-gradient solution for the free-surface height (Casulli and Cheng, 1992). Passive and active scalars (i.e. tracers, salinity and temperature) are advected using a conservative ULTIMATE QUICKEST discretization (Leonard 1991).

1.2 Implementation language

ELCOM is implemented in Fortran 90 (with F95 extensions) so that three-dimensional space can be mapped into a single vector for fast operation using array-processing techniques. Only the computational cells that contain water are represented in the single vector so that memory usage is minimized. This allows Fortran 90 compiler parallelization and vectorization without platform-specific modification of the code. A future extension of ELCOM will include dynamic pressure effects to account for nonlinear dynamics of internal waves that may be lost due to the hydrostatic approximation.

1.3 Outline of a simulation run

Setting up and running an ELCOM simulation requires the following steps:

1. prepare `bathymetry.dat` file that provides topography and grid information (see 2.2),
2. prepare `bc.dat` file that designates sets of cells for applying boundary conditions (see 2.3)
3. prepare `levee.dat` file that designates sets of cells for levee boundary conditions (see 2.4)
4. prepare `update.dat` file that designates sets of cells for update boundary conditions (2.5)
5. run the pre-processor to produce `sparsedata.unf` and `usedata.unf` files
6. configure the simulation `run_elcom` file (see 3.6)
7. specify output in a datablock file (`*.db`) for designating data output operations (see 3.7.7)
8. prepare temporal boundary condition files (`*.dat`) for data applied to the boundary condition cells (see 3.7.2)
9. prepare initial conditions files(see 3.4)

10. run the executable, directing the screen output to a file(see 3.8)
11. post-process data (see 3.9)

1.4 Conventions and definitions

In this document, a number of conventions will be used. For clarity, the computational domain will generally be referred to as a ‘lake’ However, the simulation method is designed for applicability to estuaries and coastal ocean environments.

x,y,z space	Full Cartesian space which covers all of the bathymetry
i,j,k space	Discrete integer space of computational cells which covers all of the bathymetry
land	A computational cell that cannot contain water
open	A cell used for an open boundary condition
interior	A cell that may contain water
x_rows	Number of grid cells in the x direction
y_columns	Number of grid cells in the y direction
z_layers	Number of grid cells in the (vertical) z direction
land_value	Number used in the bathymetry file to represent the land cells
open_value	Number used in the bathymetry file to represent the open boundary cells
dx, dy, dz	Grid spacing (delta) in the x,y,z direction, may be uniform or non-uniform (in metres)

Pre-Processor

2.1 Introduction

The ELCOM pre-processor is used to convert the user's three-dimensional data of lake bathymetry, boundary forcing sections and grid structure into one-dimensional vectors used by ELCOM. Configuration for running the pre-processor comes from a `run_pre.dat` file. The `run_pre.dat` file is a text file that contains the names and location of user-supplied input files and pre-processor output files. The pre-processor requires two or three user-supplied text files: `bathymetry.dat`, `bc.dat` and the optional `levee.dat` and `update.dat`. `bathymetry.dat` describes the bathymetry of the lake and grid configuration. `bc.dat` defines sets of boundary cells. `levee.dat` allows the specification of levee cells. Levee cells allow solid wall to be placed between two adjacent ELCOM cells. `update.dat` allows the specification of update cells. Update cells allow the values of various scalars to be arbitrarily modified. The pre-processor output consists of two Fortran unformatted (binary) files: `sparsedata.unf` and `usedata.unf`. The former contains information on three-dimensional space the later contains the one-dimensional spatial relationships required for the simulation. These files may be renamed, but the user must keep track of which files are 'sparsedata' files and which files are 'usedata' files.

2.2 Setting up the bathymetry file (`bathymetry.dat`)

ELCOM is implemented in Fortran 90 (with F95 extensions) so that three-dimensional space can be mapped into a single vector for fast operation using array-processing techniques. Only the computational cells that contain water are represented in the single vector so that memory usage is minimized. This allows Fortran 90 compiler parallelization and vectorization without platform-specific modification of the code. A future extension of ELCOM will include dynamic pressure effects to account for nonlinear dynamics of internal waves that may be lost due to the hydrostatic approximation.

2.2.1 Introduction

The bathymetry file is designed to allow the user to input the geometry of the simulation domain at the resolution that the simulation will be conducted ¹. The format of the bathymetry file is rigid with regards to the amount of data provided and the order in which it is listed. Each item in the bathymetry file must be present even if it is not applicable to the present simulation. Comment lines may be added into the file at any point before the actual bathymetry values. Each line of the `bathymetry.dat` file must have at least two 'items' separated by one or more spaces. An 'item' may be an integer, a real number or a character string. Comment lines must also have at least two items, the first of which is an exclamation point (the Fortran 90 comment symbol). A common crash of the ELCOM preprocessor produces a Fortran error message regarding insufficient input data. This is typically

¹The pre-processor cannot interpolate from a coarse bathymetry to a finer grid or filter from a fine grid to a coarse grid. The user must preform this task before setting up the bathymetry file.

because (1) one or more non-optional items were omitted from the input file or (2) a data line or comment line has less than two items. The file `bathymetry.dat` is organized as a series of text lines of keywords, data and comments that follow one of the formats shown below.

```
keyword      data
data        keyword
! -         comment
keyword      keyword
data(j=1,y_columns)
```

The keywords are ELCOM-defined words that indicate the type of user-provided data presented: (1) before the keyword, (2) after the keyword, or (3) on succeeding lines of the file. Additional comments may be provided after the keywords to help the user remember the keyword purpose (the code only reads the first two items on a line). Detailed information on the keywords and the format of the file are presented in the following sections.

2.2.2 Physical and computational space coordinates

In the literature, bathymetry is generally presented as the vertical distance from some horizontal baseline (often a mean water height) and may be either positive downward or positive upward. The convention used in ELCOM is an (x, y, z) space that follows the right-hand rule and has z positive in the upward direction (Figure 2.1). The user is free to choose any suitable $z = 0$ baseline for the bathymetry as long as positive z upward is maintained as the convention for measurement. The pre-processor will adjust the user-defined $z = 0$ baseline to an internal code baseline that is suitable for the ELCOM simulation. For the most part, the user does not need to know or consider this internal baseline adjustment. However, some debugging/monitoring output is presented in terms of the code coordinate system rather than the user coordinate system, so care must be taken in interpreting debugging output that is intended for use by code developers. The position of physical space origin ($x = 0, y = 0$) is set implicitly by the user through the layout of the bathymetry data in the file `bathymetry.dat`. This will be covered in more detail below. Physical space x and y values may not be negative (i.e., the x, y origin must be in a corner of the domain).

ELCOM performs its simulation on computational cells of a three-dimensional Cartesian mesh. The user needs to understand the layout of the mesh so that boundary condition cells can be properly identified. ELCOM allows non-uniform spacing in each of the horizontal dimensions (x and y) and allows non-uniform spacing in the vertical (z) direction (Although non-uniform x and y grid spacings are now allowed users must be aware that timestep limitations are set by the smallest grid size). Figure 2.2 shows a plan view (x - y plane) of a lake with a discrete (uniform) Cartesian mesh overlay. The computational domain is divided into ‘cells’ whose centers are represented in Figure 2.2 by solid dots. Dashed lines represent the faces between the cells. Each cell center has an (i, j, k) coordinate that designates its location in discrete computational space. The i, j, k values are the integer indices of a three-dimensional array. The right-hand rule is used, with (i, j, k) being non-zero, positive integer values; $k=1$ is associated with the lower-most cell layer in the domain. The (i, j) pairs of a horizontal plane represent a matrix,

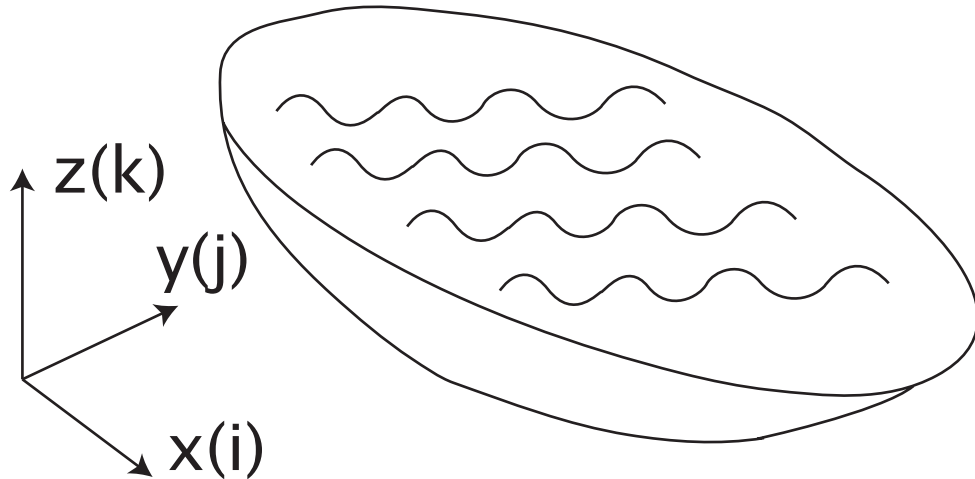


Figure 2.1 Schematic of coordinate system

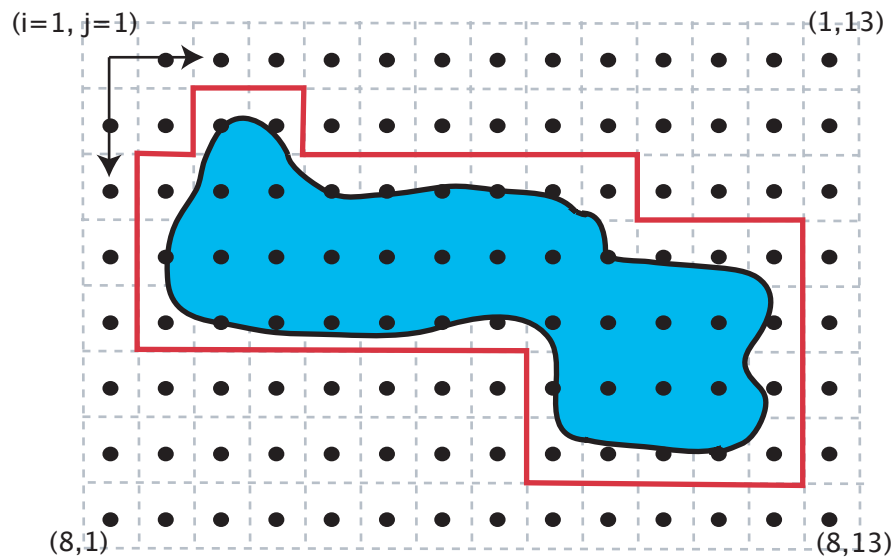


Figure 2.2 Plan view of Cartesian mesh

so that by conventional mathematical notation the $(1,1)$ position is always in the upper left corner. The ‘ i ’ index is the row index and increases *down* the page. The ‘ J ’ index is the column index and increases across the page. The (i, j) indices for corner cells are shown in Figure 2.2. For convenience and consistency, the origin $(0,0)$ of the x, y (physical space) coordinates is taken to be at the $(1,1)$ location in the computational space i, j indices.

Using the upper left corner as the origin may initially be confusing to users familiar with mirror-image storage (used by many graphics programs) where the *lower* left corner is stored in position $i = 1, j = 1$. The problem with mirror-image storage is that a simple printout of bathymetry data always appears as an upside-down mirror image of the real world when viewed with a text processor or spreadsheet. The present convention gives the user a ‘What You See Is What You Get’ view of the bathymetry simply by opening the file with a word processor or importing into a spreadsheet. This convention has been found to be intuitive for users who are not graphics experts. However,

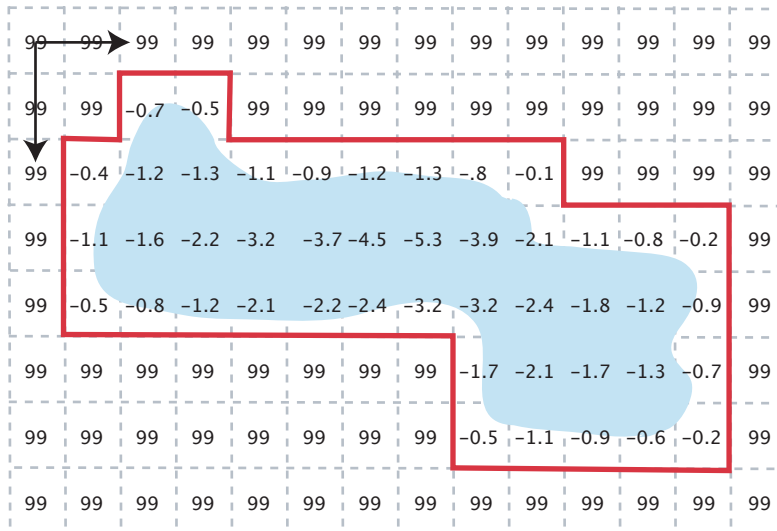


Figure 2.3 Plan view of bathymetry data

caution needs to be taken when converting raw bathymetry data from graphics programs that use a mirror-image standard. Likewise the direction of the positive u (x direction) velocity is confusing as it is positive down the page.

Figure 2.3 shows the *physical* space outline of the lake as the border of the shaded area. The discrete outline of a lake is shown with a heavy solid line. In developing a `bathymetry.dat` file for an ELCOM simulation, the user is required to provide bathymetry values (i.e. vertical distance measured from the user's $z = 0$ baseline) for each of the cells within the discrete boundary of the lake. The region outside the lake boundary is defined as the 'land' region. In Figure 2.3, the cell center dots have been replaced with numbers that represent the distance from the user's $z=0$ baseline to the bottom of the lake. In this case all of the 'land' regions have been given a value of 99. Note that the boundary of the bathymetry data set shown in Figure 2.3 has land values around the entire boundary. In general, it is required that either a land cell or open boundary cell is required around the outside of the bathymetry data; For the purposes of the `bathymetry.dat` file, the user must provide bathymetry values for *all* cells in a rectangular array, including those cells that only contain land. ELCOM will only reserve computational memory for (1) the cells which may contain water, and (2) the first layer of land cells surrounding the water cells.

Figure 2.4 shows an elevation view (in the $x - z$ plane) of a lake with a Cartesian overlay. A continuous, *physical-space* lake bottom is shown as a dashed line, with the resulting discrete lake bottom shown in a solid line. ELCOM computes the bathymetry on the face of each cell (the dashed lines) from user-supplied bathymetry data at the cell centers (on the solid dots). In this figure, the user $z = 0$ baseline is chosen as the top of the domain, so that all the bathymetry values should be negative. Note that the $k = 1$ index of i, j, k space corresponds with the lowermost layer, regardless of the user $z=0/I_i$ baseline position.

An up-to-date copy of `bathymetry.dat` is included in the examples. This file can be used as a template for creating your own `bathymetry.dat` file. The judicious use of extra comment lines and comments after the keywords,

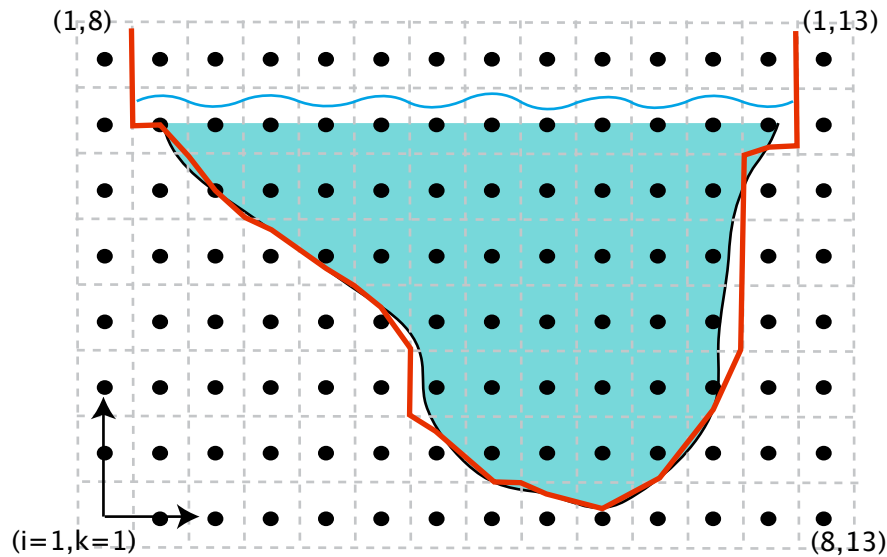


Figure 2.4 Elevation view of bathymetry data

leads to a file that is much more readable. The `bathymetry.dat` file can be renamed, but the corresponding name in the `run_pre.dat` must also be changed.

The remainder of this section will detail the use of each keyword in the `bathymetry.dat` file.

File header

keyword = FILE

The first non-comment line of the `bathymetry.dat` file must begin with the keyword FILE. Thus, the first non-comment line *must* appear as shown:

```
FILE          bathymetry.dat
```

Description lines

keywords = TITLE, ANALYST, ORGANIZATION, COMMENT

Several keywords are provided to allow the user to put identifying information at the top of the bathymetry file. This information is printed to the standard output at the start of the run to provide identification of the debugging output data. The format for the header lines is `'string'; keyword`, where the string must be a non-null character array enclosed in single quotations marks. If the string is left blank or not enclosed in quotes, the pre-processor will crash with a Fortran read error. The user must have only one COMMENT line. A typical set of header keywords is shown below.

```
'Lake Nowhere'      TITLE
'J. Citizen'        ANALYST
'A Company'         ORGANIZATION
'test run number 1' COMMENT
```

Size of input bathymetry data

```
keywords = x_rows, y_columns, z_layers, n_max, n_maxBC
```

In order to correctly read the bathymetry data from file `bathymetry.dat`, the user must tell the pre-processor the size of grid to expect. The value for `x_rows` is the number of rows in a two dimensional plan view of the bathymetry. The `y_columns` is the number of columns in the two-dimensional plan view. The number of layers in the vertical direction is identified with the keyword `z_layers`. The keyword `n_max` is used to set an estimate of the number of grid cells in the domain that are required for simulation (i.e. interior points plus the land points in one layer surrounding the interior). For most simulations, `n_max` can be set to 0, and the code will start with an estimate of `x_rows x y_columns x z_layers`. However, in some cases, this may create a temporary data array larger than can be processed using the available memory and the pre-processor will crash. The crash may produce a machine error message stating 'out of memory,' or a more enigmatic 'segmentation fault'. If insufficient memory is available, the user can estimate the actual number of grid points required for the simulation and rerun the preprocessor. Further reduction of the amount of memory used can be achieved by using the keyword `n_maxBC`. `n_maxBC` sets the maximum number of cells in any one boundary condition set. If `n_maxBC` is not specified or is set to 0 then the calculated `n_max` value is used.

```
102          x_rows
84           y_columns
22           z_layers
0            n_max
```

Land cells

```
keyword = land_value
```

The land region and is designated by a unique bathymetry value (defined using the keyword '`land_value`' that is greater than any value found within the lake. The user is free to choose the `land_value` as any convenient number that meets this criterion. Thus, if the user chooses a $z = 0$ baseline at the very top of the domain, the `land_value` may be set to zero (and all the bathymetry values should be negative numbers). If the user chooses the bottom of the domain as the $z=0$ baseline, then the `land_value` may be any positive number that is greater than the largest number appearing within the lake boundary (it is usually convenient to use 9999). In this latter case, all the bathymetry values will be positive numbers. Note that any cell whose bathymetry is not equal to the `land_value` is considered a cell within the domain; thus typographical errors can produce spurious interior cells of the domain. The user is not required to have a $z = 0$ baseline within the domain: it may be convenient to use an external reference such as sea level. This may result in the lowest point in the domain having a positive

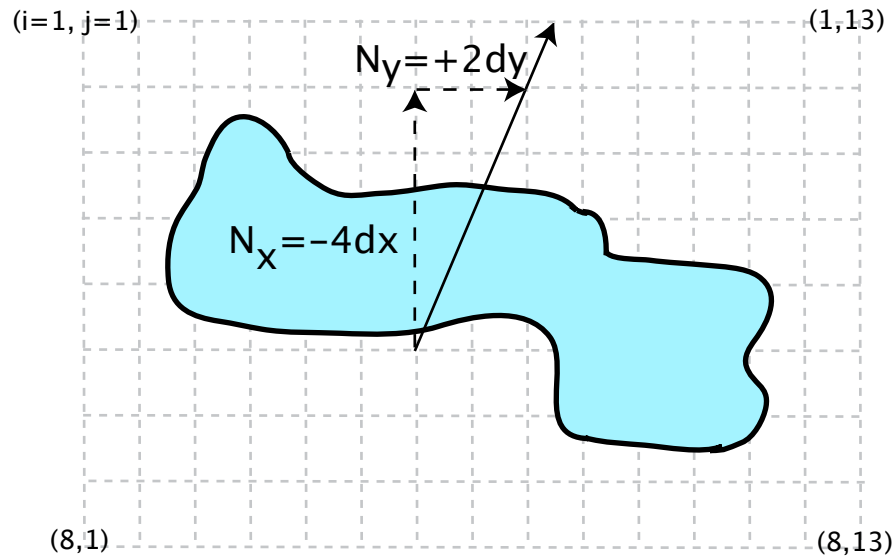


Figure 2.5 Calculation of the north vector

non-zero bathymetry value.

```
9999          land_value
```

Open boundary cells

```
keyword = open_value
```

Open boundary cells are boundaries with tidal forcing from an adjacent body of water that is not in the simulation domain. The user-defined bathymetry value that indicates an open boundary cell is defined with the keyword `open_value`.

```
8888          open_value
```

North vector

```
keywords = north_x, north_y
```

The user may create a bathymetry data file with any orientation (i.e. there is no requirement that north be at the top of the page). Since wind forcing on the free surface requires a direction, the user must designate the north direction. The north vector is determined as shown in Figure 2.5. Note that the north vector is given in terms of a vector in physical (x, y, z) space, not the discrete (i, j, k) space. Because the origin is in the upper left corner, a standard ‘north up’ data set has a north vector of $(-1, 0)$.

```
-1.0          north_x
0.5           north_y
```

Geographic position

keywords = `latitude`, `longitude`, `altitude`

The user provides the geographic position of the simulation domain using the keywords `latitude`, `longitude` and `altitude`. Longitude and altitude have no effect on the simulation but longitude is used to geo-reference output data. The latitude is used to provide appropriate Coriolis forcing in the ELCOM simulation. North latitudes are entered as positive numbers while south latitudes are entered as negative numbers. Latitude and longitude refer to the *top, left* corner of the bathymetry file.

```
texttt32.0          latitude
texttt9.46          longitude
texttt0.0           altitude
```

Horizontal grid spacing

keywords = `x_grid`, `y_grid`

The present version of ELCOM allows uniform or stretched grid spacing in the x and y direction. Separate grid spacings are implemented in each horizontal dimension so that $dx = dy$ is valid. However, the user should be careful in implementing a solution grid with different dx and dy grid spacings since this can effect the solution accuracy. For uniform grids the x and y grid spacings are designated with the key words '`x_grid`' and '`y_grid`'. If a non-uniform (stretched) horizontal grid is used a default value for '`x_grid`' and '`y_grid`' must still be defined

```
100.0              x_grid
100.0              y_grid
```

Non uniform horizontal grid spacing

keywords = `dx`, `dy` (optional)

In the horizontal direction, ELCOM version 2.0.0 allows the user to design a grid whose dimension varies in space (i.e. $dx = f[x]$ and $dy = f[y]$). Note that the dx may not vary with y or z coordinates and dy may not vary with x or z coordinates. In the `bathymetry.dat` file stretched grid horizontal grid spacings are designated by the keywords `dx` and `dy`. `dx` and `dy` are defined in the `bathymetry.dat` in the order of increasing co-ordinate (ie from $x = 1$ to $x < I >= x_rows$)

While the user is free to set the dx and dy values in an arbitrary manner, the accuracy and stability of the simulation method may be effected by the degree of non-uniformity of the grid. In general, the simulation method will perform best (i.e. with greatest accuracy) using a uniform grid. For non-uniform grids, the degradation of accuracy is a function of the rate at which the grid changes (see below for dz).

100	dx
100	dx
120	dx
132	dx
145	dx
150	dx
165	dx
171	dx
188	dx
205	dx
205	dx
205	dx
205	dx
205	dx
100	dy
100	dy
120	dy
132	dy
145	dy
150	dy
165	dy
171	dy
188	dy
205	dy
205	dy
205	dy
205	dy
205	dy

Layer thickness

keyword = `dz`

In the vertical direction, ELCOM allows the user to design a structure of horizontal layers whose thickness varies with depth (i.e. $dz = f[z]$). Note that the layer thickness may not vary with x or y coordinates. In the `bathymetry.dat` file, the successive layer thicknesses (in descending order from the top of the domain) are designated by the keyword `dz`. While the user is free to set the dz values in an arbitrary manner, the accuracy and stability of the simulation method may be effected by the degree of non-uniformity of the layers. In general, the simulation method will perform best (i.e. with greatest accuracy) using a grid with uniform dz . For non-uniform dz , the degradation of accuracy is a function of the rate at which the dz changes. Thus, a grid that has a dz that varies slowly will perform very well, while a grid with abrupt changes in grid size will be less accurate. For example, 2.6 shows two possible configurations which provide vertical resolution of 0.5 metres in the upper portion of a lake and 2.0 metre resolution in the lower portion. Both use the same number of grid cells, but the dz distribution on the left of 2.6 will show better performance.

```

0.5          dz
0.5          dz
0.5          dz
0.51         dz
0.54         dz
0.58         dz
0.65         dz
0.75         dz
0.90         dz
1.13         dz
1.41         dz
1.65         dz
1.88         dz
2.0          dz

```

Bathymetry values

keywords = BATHYMETRY DATA

The bathymetry values must be presented in the file `bathymetry.dat` following a line with the two keywords `BATHYMETRY DATA`. Each line of bathymetry values must contain exactly `y_columns` of data and there must be exactly `x_rows` of data lines following `BATHYMETRY DATA`. Each value on the line may be separated from then next by one or more spaces or tabs. Thus, for a domain with `x_rows = 8` and `y_columns = 13`, and `land_value = 99` (i.e. similar to 2.2), the bathymetry values would be entered into the `bathymetry.dat` file as shown:

BATHYMETRY DATA

```

99  99  99  99  99  99  99  99  99  99  99  99  99
99  99  -0.7 -0.5 99  99  99  99  99  99  99  99  99
99  -0.4 -1.2 -1.3 -1.1 -1.4 -2.5 -2.7 -3.3 -0.7 99  99  99
99  -1.1 -1.6 -2.2 -3.2 -3.8 -3.9 -4.1 -5.3 -5.7 -5.2 -2.4 99
99  -0.5 -0.8 -1.2 -2.1 -2.2 -2.4 -3.2 -3.3 -4.1 -5.0 -1.8 99
99  99  99  99  99  99  99  99  99  -1.7 -1.2 -0.7 99
99  99  99  99  99  99  99  99  99  -0.7 -0.9 -0.3 99
99  99  99  99  99  99  99  99  99  99  99  99  99

```

Bottom drag values (optional)

keywords = BTM_CD DATA

Version 2.0.0 allows a variable bottom drag coefficient to be specified in the `bathymetry.dat` file. Each line of values must contain exactly `y_columns` of data and there must be exactly `x_rows` of data lines following `BTM_CD DATA`. Each value on the line may be separated from then next by one or more spaces. Thus, for the domain with `x_rows = 8` and `y_columns = 13`, as shown above the values would be entered into the `bathymetry.dat` file as shown:

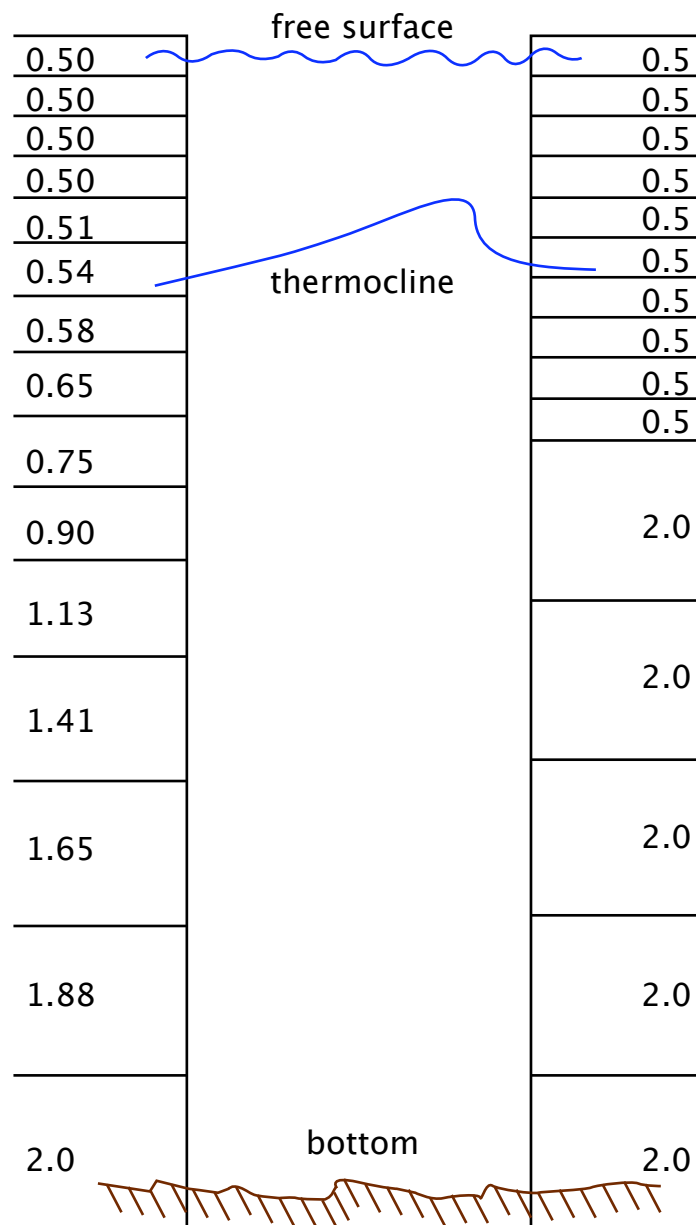


Figure 2.6 Vertical grid spacings

BTM_CD DATA

```

8      13      1e+16
1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16
1e+16 1e+16 0.002 0.002 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16
1e+16 0.002 0.002 0.002 0.002 0.002 0.002 0.002 0.002 0.002 1e+16 1e+16 1e+16
1e+16 0.003 0.003 0.003 0.003 0.003 0.003 0.003 0.003 0.003 0.003 0.003 1e+16
1e+16 0.004 0.004 0.004 0.004 0.004 0.004 0.004 0.004 0.004 0.004 0.004 1e+16
1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 0.004 0.004 0.004 1e+16
1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 0.004 0.004 0.004 1e+16
1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16

```

Bottom slope values (optional)

keywords = SLOPE_X DATA, SLOPE_Y DATA

If the 2D underflow model is to be used bottom slope values must be presented in the file `bathymetry.dat` following a line with the two keywords SLOPE_X DATA or SLOPE_Y DATA. Each line of slope values must contain exactly `y_columns` of data and there must be exactly `x_rows` of data lines following SLOPE_X DATA. Each value on the line may be separated from then next by one or more spaces or tabs. Thus, for the domain with `x_rows` = 8 and `y_columns` = 13, as shown above the slope values would be entered into the `bathymetry.dat` file as shown:

SLOPE_X DATA

```

8      13      1e+16
1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16
1e+16 1e+16 -0.006 -0.008 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16
1e+16 -0.007 -0.004 -0.008 -0.021 -0.024 -0.019 -0.020 -0.026 -0.05 1e+16 1e+16 1e+16
1e+16 -0.000 0.002 0.000 -0.005 -0.004 0.000 -0.002 0 -0.017 -0.025 -0.009 1e+16
1e+16 0.005 0.008 0.011 0.016 0.019 0.0195 0.020 0.026 0.02 0.02 0.008 1e+16
1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 0.017 0.020 0.007 1e+16
1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 0.008 0.006 0.003 1e+16
1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16

```

SLOPE_Y DATA

```

8      13      1e+16
1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16
1e+16 1e+16 -0.002 0.003 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16
1e+16 -0.008 -0.004 0.000 -0.000 -0.007 -0.006 -0.004 0.01 0.016 1e+16 1e+16 1e+16
1e+16 -0.008 -0.005 -0.008 -0.008 -0.003 -0.001 -0.007 -0.008 0.000 0.016 0.026 1e+16
1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 0.017 0.005 0.006 1e+16
1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 -0.004 0.002 0.004 1e+16
1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16

```

2.3 Setting up the boundary cell set file (bc.dat)

An up-to-date copy of `bc.dat` is included in the examples.

2.3.1 Introduction

The boundary cell set file designates logical grouping of grid cells over which boundary conditions are enforced in ELCOM. One set of points may have a river inflow, another set may have a dam offtake, another set may represent groundwater inflows, etc. Each set of boundary points is defined by a unique *reference number*, (used also by ELCOM temporal boundary condition files), *type* (one of several keywords), *title* (user reference only), and a series of cell *i, j, k* coordinates. This file gives the user complete flexibility in the assignment of diverse and complex boundary conditions around a three-dimensional topography.

2.3.2 Boundary cell set categories

The keywords for the boundary cell sets fall into seven categories, defined as follows:

land	Cell boundary which is considered to be land, on which a Dirichlet ‘no-slip’ condition is imposed on all components of the velocity, and a Neumann ‘zero-gradient’ condition is imposed on transported scalars.
slip	Cell boundary which is considered to be land, but on which the velocity components tangential to the boundary have a Neumann ‘free-slip’ condition applied, while the normal component of the velocity is a Dirichlet ‘no-flux’ condition (this may be thought of more precisely as normal Dirichlet, tangential Neumann, or NDTN). A Neumann ‘zero-gradient’ condition is imposed on transported scalars.
flow	Cell boundary through which an inflow or outflow will be imposed as a Dirichlet ‘fixed velocity’ condition. This also enforces a Dirichlet condition on transported scalars.
section	Cell boundary which is grouped into a set for application of an environmental forcing or scalar forcing boundary condition, but does not change the underlying form of the velocity boundary condition (e.g. the free surface may be divided into sections over which different wind conditions may be enforced - the type velocity boundary condition is not changed, but its magnitude is adjusted on different sections)
open	Cell boundary which is ‘open’ to inflows and outflows from another body of water where the velocity is not fixed <i>a priori</i> (i.e. not a Dirichlet boundary) but is a function of the baroclinic and barotropic forcing near the boundary.
bubbler	An interior cell on which a bubble plume destratification will be simulated.

A boundary cell set consists of a collection of cell *faces* that are the boundaries between *interior* cell and *land* cells. The **land** boundary condition is the default for all faces of cells identified by the **land_value** in the **bathymetry.dat** file and for all boundary cell faces below the bathymetry data. As such, the user will rarely need to use a **land** boundary condition in the **bc.dat** file. For a lake without any inflows, outflows or special boundary conditions, the **bc.dat** file need not be present.

2.3.3 Boundary cell set face direction

The cell faces of the boundaries are identified by the (i, j, k) coordinates of the interior cell and the direction from the cell center to the boundary. The conventions used are as follows:

xp	the $y - z$ cell face in the positive x direction from the cell center
yp	the $x - z$ cell face in the positive y direction from the cell center
xm	the $y - z$ cell face in the negative x direction from the cell center
ym	the $x - z$ cell face in the negative y direction from the cell center
top	the $x - y$ cell face above the cell
all_sides	all possible side boundary faces (i.e. all $y - z$ and $x - z$ faces)
all	all possible side boundary faces plus the bottom boundary

2.3.4 Boundary cell set keywords

The keywords used to identify boundary cell sets are a concatenation of the categories and the face direction. The valid keywords are listed below:

Table 2.1 ELCOM Boundary Condition Keywords.

land_xp	flow_xp	slip_xp	section_xp
land_xm	flow_xm	slip_xm	section_xm
land_yp	flow_yp	slip_yp	section_yp
land_ym	flow_ym	slip_ym	section_ym
land_bottom	flow_bottom	slip_bottom	section_bottom
land_all_sides	flow_all_sides	slip_all_sides	section_all_sides
land_all	flow_all	slip_all	section_all
			section_top
bubbler	bubbler_zbot	bubbler_kbot	
open_cell			
flow_multi_sides			

NOTE: bubbler_zbot and bubbler_kbot allow the user to specify the height of a bubble plume diffuser as either some height in metres (zbot) or number of cells from the bottom.

2.3.5 Organization of the boundary cell set file (bc.dat)

At the top of the boundary cell set file (`bc.dat`) the user may put any number of comment lines. Each comment line must be started with an exclamation point (!) and have at least 3 items on the line (i.e. two words or numbers separated from the exclamation point and each other by a spaces or tabs). After the initial comments, the boundary cell set file consists of a series of definitions of boundary sets. Each set consists of a *set identification line* followed by one or more lines of *cell definitions*. Each line must have three items on it or the pre-processor will crash with a Fortran file read error. Blank lines are not allowed in the boundary cell set file.

Each set identification line is of the form: `integer keyword title`. The first item, `integer`, is a *reference number* for the boundary cell set that is used in preparing temporal boundary condition data files for the ELCOM simulation run (see section 3.7.2). Each boundary cell set must have a unique `integer` reference number. The `keyword` is one of the valid keywords listed in Table 2.1. The `title` is a user-designated name for the boundary condition set. The `title` may have more than one word, although only the first word will be used by the code. After the set identification line, there will be one or more lines of cell definitions. A cell definition consists of three sets of number in the general form $i_{start} : i_{end} j_{start} : j_{end} k_{start} : k_{end}$. For all boundary condition types except bubbler_zbot and bubbler_kbot the i , j , and k are integer values of the cell positions in space for a rectangular box of cells on which the relevant boundary cell keyword will be applied. For bubbler_zbot the third column must be a single real which gives the height of the diffuser in metres above the lake bottom. For bubbler_kbot the third column must be a single integer which gives the number of cells the diffuser is above the lake bottom. An example of a simple boundary cell set definition might be

```
107 flow_xp river_inflow
   3:10  4:5  10:14
```

The preprocessor would designate as a single set, all positive x boundaries in the range $3 \leq i \leq 10$ and $4 \leq j \leq 5$ and $10 \leq k \leq 14$. A boundary cell set may have multiple lines of cell definitions:

```
13 flow_bottom groundwater_inflow
5:10  1:5  10:14
5:10  7:11  10:14
3:8  12:17  1:5
```

Note that each number pair in a cell definition must be in increasing order.

There are two simpler cases for implementing boundary cell sets (1) where i, j or k is a single value rather than a range, (2) where all possible values in i, j or k are to be included. In the former case, only a single number needs to be provided (rather than a pair), in the latter case, only the colon (:) needs to be listed. For example, the cell set for an inflow boundary along a flat side where $j = 2$ might be written as:

```
2791 flow_ym river_inflow
3:15 2 :
```

This would group all the possible y -minus cell faces for cells at any k level, where $j = 2$ and i is bounded by 3 and 15. This generalizes such that a boundary cell set may be written as

```
214 slip_all freeslip_domain
: : :
```

which enforces a free-slip boundary condition on all boundary faces (sides and bottom) of the domain.

There is no error cross-checking within the pre-processor for cell faces which have been defined as belonging to more than one boundary cell set. In some cases, multiple definitions may be desirable. For example, one might define a river inflow which covers the entire depth of the river for purposes of the velocity boundary condition; and then define a separate section at the bottom of the river that is used to place a tracer. The bc.dat file might then contain

```
2791 flow_ym river_inflow
3:15 2 :
73 section_ym tracer_inflow
3:15 2 1:5
```

In other cases, multiple set definitions for a particular face can produce unintended results. For example, if we define an inflow boundary and a global definition of free-slip boundaries on all the cells:

```
2791 flow_ym river_inflow
3:15 2 :
214 slip_all freeslip_domain
: : :
```

We then find that the cells (3:15, 2, :) get defined both with imposed velocity (Dirichlet) boundary conditions and free-slip (normal-Dirichlet, tangential-Neumann) boundary conditions. Obviously this is inconsistent and the flow simulation cannot enforce both boundary conditions. Checking for which combinations of multiple definitions are consistent and which are not is an error capturing routine that is not yet implemented. Thus, the above boundary cell sets would be processed and provided to ELCOM. Within ELCOM, only one of the two boundary conditions will actually be enforced, but one cannot tell *a priori* which boundary condition it will be without detailed examination of source code. Note that the order the items are defined may affect the operation of an inconsistent definition. Thus if the above inconsistent definition were reversed and written as:

```
214 slip_all freeslip_domain
: : :
2791 flow_ym river_inflow
3:15 2 :
```

The results in ELCOM may be different than the previous implementation *they would be wrong in a different manner*. As a general rule, the user should not provide multiple definitions of velocity boundary conditions on a single cell face. The `land`, `flow` and `slip` keywords all effect the type of velocity boundary condition; it is incumbent on the user to be sure that the cell faces defined with these types do not have multiple definitions.

The following table illustrates a bc.dat file with a few subtle uses of the boundary cell set definitions. A flow boundary has been declared on the x-minus faces of cells with $i = 2$. Thus, to get free-slip boundary conditions on all the other face, we use the keyword `slip_all` for cells with $i > 2$. Then to get free slip on the y-minus and y-plus faces at $i = 2$ requires two more definitions (These are not inconsistent with the `flow_xm` definition since they are on different faces of the same cell). This file also shows the use of the `section_top` to segregate the free surface into different sets onto which different wind data can be enforced.

```
! - - - -
! - bc.dat file
! - comment lines need three items
34 flow_xm   outflow_number_1
   2       :       :
75 section_top wind_region_a
   :       1:10    :
118 section_top wind_region_b
   :       11:21   :
25 slip_all_sides free_slip_section
3:34      :       :
26 slip_ym free_slip_near_inflow_1
2:       :       :
27 slip_yp free_slip_near_inflow_2
2:       :       :
```

2.3.6 flow_multi_sides

In ELCOM version 2.1.1 a further bc set type was introduced (`flow_multi_sides`) which allows the user to specify one inflow which enters from multiple cells on multiple faces. This was primarily introduced to allow a meandering river channel to be specified as one boundary set. And will be particularly useful when coupled with `INFLOW_DEPTH` boundary condition data. The format of `flow_multi_sides` in boundary cell set file (`bc.dat`) is slightly different from other set types.

Each `flow_multi_sides` set consists of a set identification line followed by one or more subsets of cell and face type definitions. Each line must have three items on it or the pre-processor will crash with a Fortran file read error. Blank lines are not allowed in the boundary cell set file.

Each `flow_multi_sides` set identification line is the same as for all other set types. ie

```
reference number keyword title
```

Where `keyword` must be `flow_multi_sides`. After the set identification line, there will be one or more subset definitions. Each subset consists of a *subset set identification line* followed by one or more lines of *cell definitions*.

Each subset identification line is of the form: `reference number flow_multi_sides keyword`. The subset `reference number` for each subset must be the same as the *reference number* for from the set identification line. `keyword`. must be one of `flow_xp`, `flow_xm`, `flow_yp`, `flow_y` and defines the inflow face or the subset of cells.

Following each subset identification lines is one or more lines of cell definitions. These cell definitions are the same as for an ordinary set type. ie of the general form $i_{start} : i_{end} j_{start} : j_{end} k_{start} : k_{end}$.

An example of a `flow_multi_sides` boundary cell set definition might be

```
111 flow_multi_sides   river_inflow
111 flow_multi_sides   flow_xm
  2  3  10
  3  3  9
  4  3  8
111 flow_multi_sides   flow_ym
```

```

4   3   8
4   4   7
4   5   6
111 flow_multi_sides    flow_xm
4   5   6
5   5   5
6   5   4

```

2.4 Setting up the levee file (levee.dat) (Optional)

2.4.1 Introduction

The levee set file designates any levees in the domain. Levees are solid boundaries between two adjacent cells. Pre ELCOM implements levees by adding additional ghost cells adjacent to the levee. These ghost cells are colocated with the wet cells. The format of the levee set file is essentially the same as the bc.dat file. Each set of levee points is defined by a unique *reference number*, *type* (one of several keywords), *title* (user reference only), and a series of cell i, j, k coordinates.

2.4.2 Levee set categories

The cell faces of the levees are identified by the (i, j, k) coordinates of the interior cell and the direction from the cell center to the levee. The four valid keywords for the levee sets are therefore as follows:

<code>levee_xp</code>	levee on the $y - z$ cell face in the positive x direction from the cell center
<code>levee_yp</code>	levee on the $x - z$ cell face in the positive y direction from the cell center
<code>levee_xm</code>	levee on the $y - z$ cell face in the negative x direction from the cell center
<code>levee_ym</code>	levee on the $x - z$ cell face in the negative y direction from the cell center

2.4.3 Organization of the levee set file (levee.dat)

At the top of the levee set file (`levee.dat`) the user may put any number of comment lines. Each comment line must be started with an exclamation point (!) and have at least 3 items on the line (i.e. two words or numbers separated from the exclamation point and each other by spaces). After the initial comments, the levee set file consists of a series of definitions of levee sets. Each set consists of a *set identification line* followed by one or more lines of *cell definitions*. Each line must have three items on it or the pre-processor will crash with a Fortran file read error. Blank lines are not allowed in the levee set file.

Each set identification line is of the form: `integer keyword title`. The first item, `integer`, is a *reference number*. Each levee set must have a unique `integer` reference number. The `keyword` is one of the valid keywords listed in above. The `title` is a user-designated name for the levee set. The `title` may have more than one word, although only the first word will be used by the code. After the set identification line, there will be one or more lines of cell definitions. A cell definition consists of three sets of number in the general form $i_{start} : i_{end} j_{start} : j_{end} :$. The i and j are integer values of the cell positions in space for a rectangular box of cells on which the levee cell keyword will be applied. The third value must be a colon which signifies that the levee acts on all z values.

An example of a simple levee set definition might be

```

107 levee_xp levee_bank
3 4:15 :

```

2.5 Setting up the update file (update.dat) (Optional)

2.5.1 Introduction

The update set file designates any update cells in the domain. Update cells allow the user to arbitrarily modify scalar values. Scalar values are modified at run-time using Update boundary condition files (see 3.7.4) The format

of the update set file is essentially the same as the bc.dat file. Each set of levee points is defined by a unique *reference number*, *type* (one of several keywords), *title* (user reference only), and a series of cell *i, j, k* coordinates.

2.5.2 Update set categories

The update cells are identified by the (i,j,k) coordinates of the interior cell. The three valid keywords for the update sets are therefore as follows:

<code>update</code>	scalar values are modified to the value in the Update boundary condition files
<code>update_add</code>	scalar values are modified by adding the value in the Update boundary condition files
<code>update_scale</code>	scalar values are modified multiplying by the value in the Update boundary condition files

2.5.3 Organization of the update set file (update.dat)

At the top of the update set file (`update.dat`) the user may put any number of comment lines. Each comment line must be started with an exclamation point (!) and have at least 3 items on the line (i.e. two words or numbers separated from the exclamation point and each other by spaces). After the initial comments, the update file consists of a series of definitions of levee sets. Each set consists of a *set identification line* followed by one or more lines of *cell definitions*. Each line must have three items on it or the pre-processor will crash with a Fortran file read error. Blank lines are not allowed in the levee set file.

Each set identification line is of the form: `integer keyword title`. The first item, `integer`, is a *reference number*. Each update set must have a unique `integer` reference number. The `keyword` is one of the valid keywords listed above. The `title` is a user-designated name for the levee set. The `title` may have more than one word, although only the first word will be used by the code. After the set identification line, there will be one or more lines of cell definitions. A cell definition consists of three sets of number in the general form $i_{start} : i_{end} j_{start} : j_{end} k_{start} : k_{end}$. The *i, j* and *k* are integer values of the cell positions in space for a rectangular box of cells on which the update cell keyword will be applied.

An example of a simple update set definition might be

```
12 update tracer_release
   3  4:15  2:4
```

2.6 Running the pre-processor with run_pre.dat

An up-to-date copy of `run_pre.dat` is included in the examples.

The `run_pre.dat` file is a text file that contains the names and location of user-supplied input files and pre-processor output files. The pre-processor requires two user-supplied text files: `bathymetry.dat` and `bc.dat`. The former describes the bathymetry of the lake and grid configuration. The latter defines sets of boundary cells. The `run_pre.dat` file may also contain an `update.dat` and `levee.dat` input file. The pre-processor output consists of two Fortran unformatted (binary) files: `sparsedata.unf` and `usedata.unf`. These files may be renamed, but the user must keep track of which files are 'sparsedata' files and which files are 'usedata' files.

Once the `run_pre.dat` file is in the directory from which you intend to run the pre-processor and the paths and names of the two input and two output files are specified in `run_pre.dat`, then the pre-processor can be run by invoking the name of the executable, namely:

```
pre_elcom.exe
```

ELCOM Operation

3.1 Introduction

Running an ELCOM simulation requires the user to provide several types of data: (1) initial conditions, (2) temporal boundary conditions, (3) configuration parameters, and (4) output control. ELCOM obtains this data from the user through user-prepared input files and pre-processor prepared input files.

3.2 Files for ELCOM

ELCOM is a command-line executable that communicates with the user through files. Table 3.1 provides a summary of file types, whether their presence is required or optional, the purpose of the file, and responsibility for preparation of the file. The file names in the *courier* bold font without brackets are fixed named files (i.e. the file must be present with exactly this name). File names such as `[data_name].unf` uses square brackets to indicate that various names determined by ELCOM may be in the bracketed section (these are specified in the following sections of this document). File names such as `<datablock>.db` use the `<>` brackets to indicate file names (or sections of names) which the user can set by changing the file name at the command line.

3.3 Simulation modules

ELCOM is organized into simulation modules that can be turned off/on depending on the type of simulation that is desired. A module is turned off/on by setting the value of control keywords. This is done within the `run_elcom.dat` file. These controls provide a simple method for modifying the simulation. For example: a boundary condition file can be set up to include both meteorological data and inflow data. As a simple test, the user decides to turn off the inflow. One approach would be to return to the pre-processor, eliminate the inflow boundary cell set, re-run the pre-processor, then modify the temporal boundary condition file to remove the inflow data. As a simpler approach, the user need only set `iflow=0`, and the inflow data and boundary cell sets will be ignored.

This section will give a brief overview of the modules within ELCOM and their corresponding keywords which are set within the `run_elcom.dat` file.

3.3.1 Stratification

Density stratification in ELCOM can be through temperature, salinity, or a combination of the two by using the controls `itemperature` and `isalinity` in combination with `idensity = 1`. If the user desires, temperature and salinity can be treated as passive tracers (i.e. eliminating the density term from the momentum equation) by setting `idensity = 0`. If temperature and/or salinity are turned on, they must be initialized throughout the simulation domain through the `default.***` controls, initial condition files or their associated subroutines in `elcom_user.f90`.

3.3.2 Tracers

Up to 10 passive tracers may be transported in a simulation. They may be initialized in the domain using the initial condition files or they may flow into the domain through an inflow boundary cell set using the keywords `TRACER.**` in a temporal boundary condition file.

3.3.3 Surface thermodynamics

The surface thermodynamics module controls the heat transfer across the free surface. To use this module, the user is required to provide the following data in temporal boundary condition files: solar radiation, wind speed, air temperature, relative humidity.

Table 3.1 Files for ELCOM.

File Name	Presence	Usage	Preparation	Number
<code>run_elcom.dat</code>	required	configuration control	user	1
<code><datablock>.db</code>	optional	output control	user	1
<code><environment>.dat</code>	optional	boundary control	user	10
<code><initial>.dat</code>	optional	initial condition	user	10
<code><update>.dat</code>	optional	scalar updates	user	10
<code><drifter_properties>.dat</code>	optional	drifter properties	user	10
<code><drifter_initial>.dat</code>	optional	drifter initial conditions	user	10
<code><drifter_update>.dat</code>	optional	drifter updates	user	10
<code><jet>.dat</code>	optional	jet/pump file	user	100
<code><filter>.dat</code>	optional	temporal filter constants	user	10
<code><sparsedata>.unf</code>	required	geometry data	pre-processor	1
<code><usedata>.unf</code>	required	geometry data	pre-processor	1
<code><group_name>.unf</code>	output	datablock output	ELCOM	no limit
<code>[data_name].unf</code>	output	3D data type output	ELCOM	no limit
<code>[time_]<save>.unf</code>	output	simulation space output	ELCOM	no limit
<code><restart_out>.unf</code>	output	restart file	ELCOM	1
<code>elcom.exe</code>	required	executable	make	1

3.3.4 Rain

The rain module controls the rain input into the free surface. To use this module, the user is required to provide the rain data in *m/day* in a temporal boundary condition file

3.3.5 Wind

The wind effects ELCOM simulations in three ways: (1) evaporative cooling in the surface thermodynamics module, (2) application of a wind stress to the free surface and (3) application of a wind induced momentum source to the wind mixed layer.

If the keywords `WIND_SPEED` and `WIND_DIR` in the temporal boundary condition files are associated with the boundary cell set 0 (the entire free surface), then a wind field of uniform speed and direction is applied across the entire free surface. Alternatively, the user may declare multiple boundary cell sets on the free surface (using the keyword `section_top` in the file `bc.dat` for the preprocessor). The different sections are given reference numbers in `bc.dat`, which are then used in the temporal boundary condition files to identify the `WIND_SPEED` and `WIND_DIR` for each section.

3.3.6 Inflow/outflow

Inflow and outflow boundaries must be initially declared in the preprocessor `bc.dat` file. The user must then ensure that temporal boundary conditions are provided for the flow rate in the temporal boundary condition files. For inflow conditions, all transported scalars must also be given boundary values temporal boundary condition files.

3.3.7 Conservation of Background Potential Energy

Numerical diffusion of mass tends to destratify simulated water bodies. This leads to an unintended increase in the background potential energy of the domain. Numerical diffusion of mass can become a significant problem if there are no inflows or outflows. The effects of numerical diffusion on the density stratification can be reduced by applying a vertical sharpening filter such that background potential energy is conserved. Conserving background potential energy comes at the cost of non-conservation of mass. The filtering technique attempts to reduce the

change in background potential energy during the advection calculation below a user defined threshold. At the same time the change in mass is not allowed to change beyond a second user defined threshold. The filtering technique can be performed on either salinity (`IFILTER = 2`) or temperature (`IFILTER = 3`), whichever is deemed to be the most important stratifying species. The user can alternatively set a constant filter for any valid transportable scalar through the `run_elcom.dat` file using keywords of the form `FILTER_<SCALAR>`. The `IFILTER` flag should be set to 1 in the `run_elcom.dat` file. Temporally varying filter constants can be also be applied using filter files (see section 3.7.5).

3.3.8 Bubble Plume Destratification

If the user wishes to simulate the effect of a bubble destratification device the location of bubbler outlet must first be declared in the preprocessor `bc.dat` file. The user must then ensure that temporal boundary conditions are provided for the airflow rate in the temporal boundary condition files. The temporal boundary condition files must also specify the number of ports associated with each bubbler boundary condition.

3.3.9 Underflow

If the user chooses to use the 2D Underflow model of Dallimore et al. (2001) to simulate a dense inflow then inflow boundaries and temporal boundary condition files must be specified as above. In addition to temporal boundary conditions for flow rate and scalar concentrations the user must also specify the height of the underflow at the boundary using the keyword `INFLOW_HEIGHT` within the temporal boundary condition file. The `IUNDERFLOW` flag should be set in the `run_elcom.dat` file.

NOTE: The underflow model is not presently compatible with CAEDYM.

3.3.10 Water Age Modelling

If the user wishes to simulate the average age of water in a cell. The `IRETENTION` flag should be set in the `run_elcom.dat` file. This gives each cell a water age of zero at the start of simulation, each cell is then incremented by `del.t` each timestep. Any water entering the domain is given an age of zero. The water is then transported as a valid transportable scalar.

3.3.11 Drifters / Particles

If the user wishes to simulate the movement of lagrangian particles of semi-lagrangian drifters then the number of drifters/particles simulated is given by the `NDRIFTERS` keyword in the `run_elcom.dat` file. The properties and initial conditions of the drifters are given in the `drifter_properties_file` and `drifter_in_file`. Drifter positions and properties can be updated via `update_drifter_file`

3.3.12 Jets/Pumps

If the user wishes to simulate effect of a jet/pump destratification the user must then ensure that temporal boundary conditions are provided for the jet thrust, radius and depth or height are in the jet files. The `IJET` control flag in `run_elcom.dat` is used to control if the jets are simulated.

3.3.13 Non-hydrostatic code

If the user wishes to simulate non-hydrostatic pressure effects using the code outlined in Bothel et al. The non-hydrostatic code generally requires horizontal grid sizes $O(10m)$ and drastically increases runtimes

3.4 Initialization

3.4.1 Uniform initial conditions

For simple initial conditions that (1) distribute any valid scalar over the entire simulation volume or (2) provide a flat free surface across the entire domain, the user can set the desired values through the `run_elcom.dat` file using keywords of the form `DEFAULT_****`. Any default value entered will be overwritten by the initial profile or horizontal files of user initialization subroutines (when invoked).

3.4.2 Vertical Profile

A vertical variation in the initial conditions for any valid transportable scalar can be defined using an input file (keyword = `initial_profile_file`). The input file may have any name but must be located in the `infiles` directory.

These files require a format that consists of four header lines followed by columns of data. Each column is data of a particular type (defined by a keyword in the header). The first column must be depth in metres. Each row is data at a particular depth. The file may have as many comment lines (preceded by `!`) at the top of the file as desired. Comment lines within the header data must have as many items as there are columns (i.e. if there are 3 columns a valid comment line is `! - -`). No comment lines are allowed within the columns of data. The columns must be completely dense with data (i.e. there may not be "missing" data such that a row has less data than the number of columns).

The first header line contains one number: the number of data sets (columns) in the initial profile file *not including the depth column*.

The second line in the header the number of depths, and the first column in the data file should be a `DEPTH` column. Note that the user does *not* have to supply initial condition data at the same depth interval that the simulation uses: the simulation interpolates (linearly) to find the appropriate data value at a particular depth.

The third line in the header must have the `i` and `j` coordinates for the location of the profile. This is only used if more than one profile file is given for the scalar (see 3.4.3).

The fourth line in the header contains the keyword that describes the data in the column. The first keyword should be `DEPTH` and the following keywords should be a valid transportable scalar given in the Valid Scalars table in Section 3.10.

The remaining lines in the file contain the data for the specified depths. The depth values should start at zero and be monotonically increasing. The depth is defined as the vertical distance downward from the value of the `user_default_height` (set in the `run_elcom.dat` file).

```
! Example: ic.dat
2  data sets
4  number of depths
15 20  i,j
DEPTH  WTR_TEMP    SALINITY
0.0    25.0         0.0
5.1    23.1         15.1
6.2    20.2         35.0
10.4   18.3         35.2
```

3.4.3 Multiple Vertical Profiles

To increase the flexibility of initialisation ELCOM Version 2 allows for multiple profile initial condition files. If more than one initial condition file is specified for any scalar then ELCOM interpolates between files to initialise the domain. Each profile is first linearly interpolated on to the vertical layers and then each point is horizontally interpolated using the inverse distance weighting method such that

$$S_{i,j,k} = \sum_{n=1}^N \frac{S_{n,k} r_n^\alpha}{r_n^\alpha} \quad (3.1)$$

where α is the distance weighting specified by the flag `IC_DIST_WEIGHTING` in `run_elcom.dat` (default = 2.0), r is the distance between the column and profile n and N is the number of profiles.

3.4.4 Horizontal Variation

A horizontal variation in the initial conditions for any valid transportable scalar can be defined using an input file (keyword = `textttinitial_horiz_file`). The input file may have any name but must be located in the `infile` directory.

These files require a format that consists of three header lines followed by a value for each (i,j) column of the domain. The file may have as many comment lines (preceded by !) at the top of the file as desired. Comment lines within the header data must have as many items as there are columns (i.e. if there are 3 columns a valid comment line is ! - -). No comment lines are allowed within the columns of data. The columns must be completely dense with data (i.e. there may not be "missing" data such that a row has less data than the number of columns).

The first header line contains two number: the number of x rows and the number of y columns. These must be the same as the number of rows and columns in the bathymetry file

The second line gives a land value for land cells

The third line in the header contains the keyword that describes the scalar to be initialised. This keyword should be a valid tranporstable scalar given in the Valid Scalars table in 3.10.

The remaining lines in the file contain the data. The matrix should be the same size as the bathymetry data in your bathymetry file.

```
! Example: ic.dat
8 13 xrows yrows
1e+16 land val
SALINITY
1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16
1e+16 1e+16 9.0 9.5 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16
1e+16 8.1 8.5 9.3 9.5 10.0 10.2 11.0 11.6 11.8 1e+16 1e+16 1e+16
1e+16 8.0 8.5 9.2 9.5 10.0 10.3 10.6 11.3 11.6 11.8 12.0 1e+16
1e+16 8.0 8.5 9.1 9.3 9.5 9.5 9.7 9.8 10.0 10.2 11.2 1e+16
1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 10.0 10.0 10.5 1e+16
1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 9.1 9.2 9.5 1e+16
1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16 1e+16
```

3.4.5 User subroutines

For `elcom` users who are willing to modify and compile source code, the ability to set up complex initial conditions is available through the adding of code to subroutines in the `elcom_user.f90` file. The subroutines in this file are only called if the appropriate `user_init_***` control in the `run_elcom.dat` file is non-zero. The stub routines supplied with `ELCOM` are designed to be executed with the `user_init_***` controls = 1. However, this can be modified by the user so that different initializations are performed for different values of `user_init_***`.

3.5 Configuration controls

3.5.1 Time step

The time step (keyword = `del_t`) is the number of seconds that the simulation advances in time for each simulation step.

3.5.2 Number of steps

The total number of time steps to be computed in the present simulation run is set using the keyword `iter_max` in the `run_elcom.dat` file.

3.5.3 Start date

The start date is entered in CWR Julian Day format which specifies the year and day using a real number with 7 digits to the left of the decimal place as `yyyyddd.ddd`. The first 4 digits are the year and the subsequent 3 digits are the Julian Day (days since 1st of January). Hours, minutes, seconds are represented as fractions of a day (ie. 1am on January 1st, 2000 is represented as 2000001.041667).

3.5.4 CGM tolerance

The `CGM_TOL` is the residual at which iteration of the conjugate gradient method (used for the free-surface solution) is stopped.

3.5.5 Turbulence

For stratified flow, the base closure scheme can be supplemented by unstable mixing (`iclosure = 1`) algorithms. The unstable mixing algorithm mixes momentum and transported scalars in vertical cells where an unstable gradient of potential density exists (i.e. $Ri < 0.0$). For stratified flows driven by surface wind, the wind-mixed-layer closure scheme with energy transport (`iclosure = 6`) is recommended. This approach computes the turbulent kinetic energy due to wind stirring, convective overturns and wind shear, then uses this to compute vertical mixing.

3.5.6 Output frequency

Outputs of data that are not controlled by the `datablock` file are set through the `iter_out.***` and `start_output.***` keywords. The former controls the time step interval for providing output, the latter controls the first step on which output will be provided.

3.6 Configuration using file `run_elcom.dat`

An up-to-date copy of `run_elcom.dat` is included in the examples.

ELCOM looks for the file `run_elcom.dat` in the local directory to set the configuration of the simulation run. The `run_elcom.dat` configuration file controls execution of optional modules and models within ELCOM, allowing the user to set the parameters that govern the physical and numerical simulation.

The format of the file is flexible in that lines may occur in any order, and not all items must be present. Comment lines (preceded by `!`) may occur anywhere in the file. Blank lines are not allowed in the file (the code stops reading at a blank line). Each line (including comment lines) must have at least two items separated by a blank space. The comment marker (`!`) counts as an item, so a valid line is `! !` or `! -`, while `!!` (no space between) is not valid. The general format for the file is a series of lines, with each line listing a user setting, then a keyword that defines the parameter. The individual keywords and settings are explained briefly below:

File headers

<code>FILE</code>	file name for this file (must be <code>run_elcom.dat</code>)
<code>ANALYST</code>	user name (must be inside single quotes <code>'</code>)
<code>ORGANIZATION</code>	user company name (must be inside single quotes <code>'</code>)
<code>COMMENT</code>	user comment (must be inside single quotes <code>'</code>)
<code>ANALYST</code>	user name (must be inside single quotes <code>'</code>)
<code>CASE_KEYWORD</code>	allows user to add code to be executed for a particular simulation case. Must be a string without blank spaces enclosed in single quotes

Simulation module controls

<code>iheat_input</code>	controls the surface thermodynamics 0 = no surface thermodynamics 1 = surface thermodynamics model - input solar radiation is measured value long wave model is determined by input keywords
<code>iatmstability</code>	controls atmospheric stability correction in thermodynamics 0 = no atmospheric stability correction 1 = atmospheric stability correction
<code>irain</code>	controls the rainfall input 0 = no rain 1 = rain
<code>iflow</code>	controls inflow/outflow (Dirichlet boundaries) overrides boundary condition files 0 = no inflow/outflow 1 = inflow /outflow model 1
<code>iunderflow</code>	controls 2D underflow model overrides boundary condition files 0 = modelling of inflows with the underflow model 1 = where INFLOW_HEIGHT is specified for a bc in the temporal boundary files the underflow model will be used
<code>ibubbler</code>	controls bubble plume destratification overrides boundary condition files 0 = no bubble plume destratification 1 = bubble plume destratification model 1
<code>itemperature</code>	controls whether temperature is a transported scalar 0 = no temperature simulation 1 = temperature as transported scalar
<code>isalinity</code>	controls whether salinity is a transported scalar 0 = no salinity in simulation 1 = salinity as a transported scalar
<code>idensity</code>	controls whether density (buoyancy term) is included in the simulation 0 = no buoyancy term 1 = density computed from UNESCO equation of state and used for buoyancy term
<code>ntracer</code>	sets the number of passive tracers which will be transported in a simulation 0 = no tracers 1 to 10 = number of tracers initialized or supplied through inflows or updates
<code>ndrifiers</code>	sets the number of drifters or particles which will be simulated 0 = no drifters 1 or more = number of drifters
<code>ijet</code>	controls whether jets or pumps are included in the simulation 0 = no jets 1 = jets will be simulated
<code>ICAEDYM</code>	controls whether the CAEDYM water quality module is used 0 = CAEDYM off 1 = CAEDYM on

<code>icoriolis</code>	controls whether or not the latitude (from <code>bathymetry.dat</code>) is used to compute a Coriolis term 0 = Coriolis off 1 = Coriolis on
<code>inonhydrostatic</code>	controls whether or not the non-hydrostatic approximations of Botelho et al are used are used. 0 = code is hydrostatic 1 = code is non-hydrostatic

Initialization and update options

<code>irestart</code>	use of restart file for initial conditions 0 = do not user restart file 1 = use restart file specified by <code>restart_in_file</code>
<code>user_init_u_vel</code>	controls user written subroutine for initializing x velocity 0 = ignore subroutine <code>user_u_init</code> 1 = execute subroutine <code>user_u_init</code> with <code>USER_START_U = 1</code>
<code>user_init_v_vel</code>	controls user written subroutine for initializing y velocity 0 = ignore subroutine <code>user_v_init</code> 1 = execute subroutine <code>user_v_init</code> with <code>USER_START_V = 1</code>
<code>user_init_w_vel</code>	controls user written subroutine for initializing y velocity 0 = ignore subroutine <code>user_w_init</code> 1 = execute subroutine <code>user_w_init</code> with <code>USER_START_W = 1</code>
<code>user_init_temperature</code>	controls user written subroutine for initializing temperature 0 = ignore subroutine <code>user_temperature_init</code> 1 = execute subroutine <code>user_temperature_init</code> with <code>USER_START_TEMPERATURE</code> = 1
<code>user_init_salinity</code>	controls user written subroutine for initializing salinity 0 = ignore subroutine <code>user_salinity_init</code> 1 = execute subroutine <code>user_salinity_init</code> with <code>USER_START_SALINITY</code> = 1
<code>user_init_tracer</code>	controls user written subroutine for initializing tracer 0 = ignore subroutine <code>user_tracer_init</code> 1 = execute subroutine <code>user_trace_init</code> with <code>USER_START_TRACER = 1</code>
<code>user_init_height</code>	controls user written subroutine for initializing free surface height 0 = ignore subroutine <code>user_height_init</code> 1 = execute subroutine <code>user_height_init</code> with <code>USER_START_HEIGHT = 1</code>
<code>user_init_extinction</code>	controls user written subroutine for initializing the light extinction (or attenuation) coefficient 0 = ignore subroutine <code>user_extinction_init</code> 1 = execute subroutine <code>user_extinction_init</code> with <code>USER_START_EXTINCTION = 1</code>
<code>latitude</code>	Simulation latitude (optional). Overrides value set in bathymetry file.

Turbulence modeling

<code>iclosure</code>	controls the type of closure scheme used 0 = constant eddy viscosity 1 = closure model 1 (constant eddy viscosity plus mixing of statically unstable stratification) 2 = disabled 3 = disabled 4 = disabled 5 = disabled 6 = wind-mixed layer model including mixing energy transport - RECOMMENDED
<code>DIFFUSIVITY<SCALAR></code>	Horizontal diffusivities of transportable scalars (same for both X and Y dirn). <code><SCALAR></code> must be a valid transportable scalar (see the Valid Scalars table in section 3.10) and must be in capital letters
<code>DIFFUSIVITY_X<SCALAR></code>	Horizontal diffusivities of transportable scalars in X and Y dirn respectively.
<code>DIFFUSIVITY_Y<SCALAR></code>	<code><SCALAR></code> must be a valid transportable scalar (see the Valid Scalars table in section 3.10) and must be in capital letters
<code>DEFAULT_DIFFUSIVITY</code>	Default Horizontal diffusivities of transportable scalars (same for both X and Y dirn). Overridden by values for individual scalars

Meteorological Sensor Heights

<code>WIND_SPEED_HEIGHT</code>	Height above water surface at which windspeed data was collected in metres. If it is not specified, 10m is assumed.
<code>SCALAR_HEIGHT</code>	Height above water surface at which air temperature and humidity data was collected in metres. If it is not specified, 10m is assumed.

Model Settings and Controls

<code>mean_albedo</code>	mean albedo of the water
<code>drag_btm_cd</code>	drag coefficient on bottom cells
<code>model_grav_damp_x</code>	damping of gravity for x baroclinic term
<code>model_grav_damp_y</code>	damping of gravity for y baroclinic term

Scalar maximum and minimum for debugging.

<code>min.<SCALAR></code>	Minimum and Maximum values of scalars allowed before debugging occurs.
<code>max.<SCALAR></code>	<code><SCALAR></code> must be any valid scalars (see the Valid Scalars table in section 3.10) and must be in capital letters
<code>ihardlimit</code>	controls the limiting of scalars 0 = no limiting of scalars 1 = limit scalars to <code>min.<SCALAR></code> and <code>max.<SCALAR></code>

Debugging controls

<code>debug_check</code>	controls level of error checking in code 0 = no error checking 1 = checking for errors, stop on fatal errors 2 = highest level of checking for errors, stop on fatal errors 3 = highest level of checking for errors, stop on warning
--------------------------	---

<code>debug_print</code>	controls level of debugging printout 0 = none 1 = minimal 2 = moderate 10 = extensive
<code>debug_point</code>	sparse data index point for debug output
<code>debug_baroclinic_x</code>	error trapping 0 = off 1 = on
<code>debug_baroclinic_y</code>	error trapping 0 = off 1 = on

Output frequency

<code>iter_out_monitor</code>	time step interval for output of monitoring data
<code>iter_out_save</code>	time step interval for output of saved full simulation data
<code>iter_out_restart</code>	time step interval for output of restart data

Output start time

<code>start_output_monitor</code>	time step to start providing monitoring data
<code>start_output_save</code>	time step to start providing simulation save files
<code>start_output_restart</code>	time step to start providing simulation restart files

Time controls

<code>start_date_CWR</code>	start day in CWR Julian Day format yyyyddd (real)
<code>del_t</code>	size of time step (in seconds)
<code>iter_max</code>	maximum number of time steps in the simulation

Iterative (conjugate gradient method) solution controls

<code>CGM_TOL</code>	Tolerance (residual) for stopping conjugate-gradient solution of free-surface equation.
<code>CGM_MIN</code>	Minimum number of iterations that must be computed for the conjugate gradient method to solve the free-surface (over-rides the <code>CGM_TOL</code>)
<code>CGM_MAX</code>	Maximum number of iterations that must be computed for the conjugate gradient method to solve the free-surface

Scalar Filtering controls

<code>IFILTER</code>	Filter control 1 = off (default) 2 = constant filter or filter from filter files 2 = background potential energy filter - salinity 3 = background potential energy filter - temperature
<code>filter_bPE_threshold</code>	Maximum allowable change in background potential energy during one advection timestep. 1e-9 is used for Lake Kinneret.
<code>filter_mass_threshold</code>	Maximum allowable change in mass during one advection timestep. 1e-7 is used for Lake Kinneret.

<code>FILTER<SCALAR></code>	Constant filter value for a transportable scalars. <code><SCALAR></code> must be a valid transportable scalar (see the Valid Scalars table in section 3.10) and must be in capital letters Overridden by values from temporal filter files
<code>DEFAULT_FILTER</code>	Default filter constant for all transportable scalars. Overridden by values for individual scalars

Default (uniformly distributed) values

<code>DEFAULT_HEIGHT</code>	Default water height (in metres). May be over-riden by user-prepared subroutine if <code>user_init_height = 1</code> . Note that default height must be lower than the maximum height of the domain or an error will occur.
<code>DEFAULT_WIND_SPEED</code>	Default wind speed (in ms^{-1}). May be over-riden by user-prepared input file (see following section)
<code>DEFAULT_WIND_DIR</code>	Default wind direction (in degrees measured counter-clockwise from North in direction wind is coming <i>from</i>). May be over-riden by user-prepared input file (see following section)
<code>DEFAULT_<SCALAR></code>	Default value of scalar. May be over-riden by user-prepared subroutines and initial condition files <code><SCALAR></code> must be a valid transportable scalar (see the Valid Scalars table in section 3.10) and must be in capital letters
<code>DEFAULT_PAR_EXTINCTION</code>	Default light extinction coefficient for PAR band. May be over-riden by user-prepared subroutine if <code>user_init_extinction = 1</code> .
<code>DEFAULT_NIR_EXTINCTION</code>	Default light extinction coefficient for NIR band. May be over-riden by user-prepared subroutine if <code>user_init_extinction = 1</code> .
<code>DEFAULT_UVA_EXTINCTION</code>	Default light extinction coefficient for UVA band. May be over-riden by user-prepared subroutine if <code>user_init_extinction = 1</code> .
<code>DEFAULT_UVB_EXTINCTION</code>	Default light extinction coefficient for UVB band. May be over-riden by user-prepared subroutine if <code>user_init_extinction = 1</code> .
<code>DEFAULT_BC</code>	Default boundar conditions. 1 = no-slip all 2 = free-slip all 3 = drag all 4 = neumann all 5 = free-slip sides, no-slip bottom 6 = free-slip sides, drag bottom 7 = free-slip sides, neumann bottom 8 = values defined in <code>elcom_mixing.f90</code> 9 = Turbulent BBL

Input file names (require exact file name, including extension)

<code>3D_data_file</code>	Name of the file containing the 3D data produced by the pre-processor (sparsedata.unf)
<code>preprocessor_file</code>	Name of the file containing the 1D data (maps) produced by the pre-processor (usedata.unf).
<code>initial_profile_file</code>	Name(s) of the files initial profile information
<code>initial_horiz_file</code>	Name(s) of the files initial horizontal information
<code>boundary_condition_file</code>	Name(s) of the files containing boundary condition information
<code>profile_boundary_condition_file</code>	Name(s) of the files containing profile boundary condition information

<code>update_file</code>	Name(s) of the files containing update boundary condition information
<code>filter_file</code>	Name(s) of the files containing temporal scalar filter information
<code>drifter_properties_file</code>	Name of the file containing physical information about drifters
<code>drifter_in_file</code>	Name of the file containing initial information and run controls for drifters
<code>update_drifter_file</code>	Name of the file containing update information for drifters
<code>jet_file</code>	Name(s) of the files containing jet/pump destratification device properties
<code>datablock_file</code>	Name of the file containing output datablock configuration

File Directories (require quote marks around; note that incomplete quotes can cause code to hang)

<code>infile_dir</code>	input files directory (ie.datablock.db,sparsedata.unf, usedata.unf, boundary condition, update, initial condition and filter files)
<code>outfile_unf_dir</code>	output *.unf files directory. Unformatted binary output files specified by datablock.db file.
<code>outfile_txt_dir</code>	ELCOM diagnostic ascii format files.

Output files names (do not include extension)

<code>restart_save_file</code>	kernel of name used for simulation save file (time stamp and .unf are appended by ELCOM)
<code>restart_out_file</code>	kernel of name used for simulation restart output file (.unf is appended by ELCOM)

3.7 Input files

3.7.1 sparsedata.unf and usedata.unf

The files `sparsedata.unf` and `usedata.unf` are required files for ELCOM. These are the Fortran unformatted (binary) files produced by the pre-processor. The user may rename these files, but must keep track of which file is the `sparsedata` file (keyword = `3D_data_file`) and which is the `usedata` file (keyword = `preprocessor_file`). ELCOM will look for the files specified in the `run.elcom.dat` file. If the incorrect files are used, the simulation will crash.

3.7.2 Temporal boundary condition (1D) files

The one-dimensional temporal boundary condition files (keyword = `boundary_condition_file`) are set up by the user to provide environmental forcing information that changes as a function of time. These files require a format that consists of four header lines followed by columns of data. The data supplied are values for particular instants of time that are applied at particular points in space (i.e. over a boundary cell set). Each column is data of a particular type (defined by a keyword in the header). The first column must be time in CWR format (yyyyddd.dd). Each row is data at a particular time. The file may have as many comment lines (preceded by !) at the top of the file as desired. Comment lines within the header data must have as many items as there are columns (i.e. if there are 3 columns a valid comment line is ! - -). No comment lines are allowed within the columns of data. The columns must be completely dense with data (i.e. there may not be "missing" data such that a row has less data than the number of columns).

The first header line contains one number: the number of data sets (columns) in the boundary condition file *not including the time column*.

The second line in the header must contain the number 0, indicating that the CWR time format will be used in the data below, and the first column in the data file should be a `TIME` column. Note that the user does *not*

Table 3.2 Temporal Boundary Condition Data-types.

Keyword	Units	Description
HEIGHT	m	water height for open boundary condition
INFLOW	$m^3 s^{-1}$	inflow volume
INFLOW_MAXDEPTH	m	maximum depth of inflow (see below)
INFLOW_MINDEPTH	m	minimum depth of inflow (see below)
INFLOW_HEIGHT	m	inflow height for underflow model
OUTFLOW	$m^3 s^{-1}$	outflow volume
AIRFLOW	$m^3 s^{-1}$	airflow rate for bubble plume destratification
N_PORTS		Number of ports in destrat device
RAIN	$mday^{-1}$	rainfall rate
WIND_SPEED	ms^{-1}	wind speed
WIND_DIR	$^{\circ}$	wind direction ($^{\circ}$ measured clockwise from north that the wind comes from)
SOLAR_RAD	Wm^{-2}	solar radiation that reaches water surface
LW_RAD_IN	Wm^{-2}	Incoming longwave radiation
LW_RAD_NET	Wm^{-2}	Net loangwave radiation (+ve incoming)
ATM_PRESS	Pascals	atmospheric pressure
CLOUDS		fractional cloud cover (<i>non-dimensional number between 0 and 1 inclusive</i>)
REL_HUM		relative humidity (<i>non-dimensional number between 0 and 1 inclusive</i>)
AIR_TEMP	$^{\circ}C$	air temperature
<SCALAR>	-	valid transportable scalar

have to supply boundary condition data at the same time interval that the simulation uses as the computational time step (`del_t`): the simulation interpolates (linearly) between the temporal boundary condition data to find the appropriate data value at an instant in time.

The third line in the header must have an integer for each column. This is the reference number for the applicable set of boundary cell set (defined in the file `bc.dat` during the setup of the pre-processor) that each column of data will be applied to. For example, the example `bc.dat` file shown at bottom of section 2.3.5) provides three sets of boundary points with reference numbers 34, 75, 118, 25, 26, 27. These numbers are used in the third line of the header to identify the points to which a column of data applies. Note that a given reference number may be used for multiple data columns in multiple files, so that (for example) inflow volumes and temperatures for a particular set of boundary points may be supplied in one file, while tracer inflow for the same set of points may be supplied in a different file. Any temporal data boundary condition applied to the *entire* free surface (which is not required to be defined as a boundary cell set in `bc.dat`) is given a reference number of 0 (zero).

The fourth line in the header contains the keyword that describes the data in the column. The keywords are limited to a set specifically defined and understood by ELCOM and are listed in the table below.

The remaining lines in the file contain the data for the specified time (or time intervals).

vspace3mm

The `INFLOW_MAXDEPTH` and `INFLOW_MINDEPTH` bc type allows user to specify a depth range for an inflow. ie. Any bc cells below `INFLOW_MAXDEPTH` and any bc cells above `INFLOW_MINDEPTH` will not be enforced for the timestep. `INFLOW_MAXDEPTH` allows users to implement a bc along the thalweg of a reservoir and then specify the inflow to only come in at the top 1m (say) of the bc. The inflow then enters the reservoir at the point where the thalweg meets the free surface. This allows improved smiulation of reservoirs with large changes in surface height. These two keywords also operate on outflow boundaries and allow for simulation of selective withdrawal systems that are specified as a depth below the surface.

3.7.3 Temporal profile boundary condition (2D) files

The two-dimensional temporal profile boundary condition files (keyword = `profile_boundary_condition_file`) are set up by the user to provide scalar boundary information that changes as a function of time and depth. These files require a format that consists of four header lines followed by sections of data.

The first header line contains one number: the number of data sets in each section in the boundary condition file *including depth but not including the time row*.

The second line in the header must contain the number 0, indicating that the CWR time format will be used in the data below, and the first column in the data file should be a `TIME` column. Note that the user does *not* have to supply boundary condition data at the same time interval that the simulation uses as the computational time step (`del_t`): the simulation interpolates (linearly) between the temporal boundary condition data to find the appropriate data value at an instant in time.

The third line in the header must have an integer for each datatype (including `TIME` and `DEPTH`). This is the reference number for the applicable set of boundary cell set (defined in the file `bc.dat` during the setup of the pre-processor) that each section of data will be applied to. Note unlike 1D boundary condition files each profile file may only apply to one boundary set.

The fourth line in the header contains the keyword that describes the data in the sections (including `TIME` and `DEPTH`).

The sections of data supplied are values of scalars with depth for a particular instants of time. Each section of data consists of one header row, one row giving the depth values and a number of rows for each datatype. The header row gives the time of the profile in (CWR format `yyyddd.dd`) and an integer specifying the number of depths. The format of the depth row is the keyword `DEPTH` followed by a depth value for each profile point. Each depth is specified in metres below the free surface. Each subsequent row in a section gives the value of one scalar at each depth point. The format of the depth row is a valid transportable scalar keyword `<SCALAR>` followed by a scalar value for each profile point. Each data row must have a value for each depth point. Each section must have the same number of rows and the order of each datatype must be the same for each section.

3.7.4 Update boundary condition files

The update boundary condition files (keyword = `update_file`) are set up by the user to provide updates to scalar values on update boundary conditions. These files have primarily been designed for the simulation of tracer releases from lake interiors. The format of update file is exactly the same as temporal boundary condition file except update files contain two time columns. The relevant scalar values will be updated on the boundary set for all timesteps between the first and second time column.

If the current timestep is not between the first and second time column of any row then no changes are made to the scalar values of the boundary set by the update routine.

3.7.5 Temporal filter constant files

The temporal filter constant files (keyword = `filter_file`) are set up by the user to provide temporally varying scalar filter constants. The format of filter files is exactly the same as 1D Temporal boundary condition files. Although the 3rd line of the file defining the `bc` set to be used is given it is not used by ELCOM. All keywords must be valid transportable scalars.

3.7.6 Drifter files

Drifter Properties Files

The drifter properties file (keyword = `drifter_properties_file`) is set up by the user to provide physical information about a semi-lagrangian drifter, this file can be blank if only lagrangian particles are simulated. A semi-lagrangian drifter is modelled as a number of different physical components (in V2.2.0 the maximum number of components is 5)

Each line of the `drifter_properties_file` should have four columns.

The first column specifies a value; the second column gives the drifters that the value will be applied to (:, for all drifters, `n1:n2`, for drifters `n1` to `n2`); the third column gives the drifters components that the value will be applied to (:, for all components, `n1:n2`, for components `n1` to `n2`); the fourth column is a keyword which specifies which property we are setting.

Valid properties are:

<code>mass</code>	the mass of the component (kg)
<code>area</code>	Cross sectional area of the component (m^2)
<code>length</code>	vertical length of the component (m)
<code>drag_cd</code>	Drag coefficient for force balance
<code>distance</code>	Vertical distance from bottom of probe to bottom of component (m)
<code>shape</code>	Shape of the component
	1 = sphere
	2 = cylinder
	3 = rectangle
<code>type</code>	Type of component
	1 = submerged - forced by modelled water velocities
	2 = subsurface - forced by surface wind shear velocity and surface
	3 = surface - forced by wind

Therefore an example file for drifters consisting of five components (antenna, surface buoy, subsurface buoy, pole connecting the drogue to the subsurface buoy and the drifter sails) may look something like:

```
! - - - -
!! Example Drifter Properties File
!! NB each comment line should have 4 words
! - - - -
! Component 1 sail
26.0  :  1  mass
1.5   :  1  area
1.0   :  1  length
0.3   :  1  drag_cd
0.0   :  1  distance
3     :  1  shape
1     :  1  type
! Component 2 Pole
3.0   :  2  mass
0.001 :  2  area
1.0   :  2  length
0.3   :  2  drag_cd
1.0   :  2  distance
2     :  2  shape
1     :  2  type
! Component 3 sub-surface float
0.2   :  3  mass
0.2   :  3  area
0.2   :  3  length
0.3   :  3  drag_cd
2.3   :  3  distance
3     :  3  shape
```

```

2      :      3      type
! Component 4 surface float
0.1    :      4      mass
0.01   :      4      area
0.1    :      4      length
0.3    :      4      drag_cd
2.5    :      4      distance
3      :      4      shape
3      :      4      type
! Component 5 antennea
1.0    :      5      mass
0.01   :      5      area
0.3    :      5      length
0.3    :      5      drag_cd
2.6    :      5      distance
2      :      5      shape
3      :      5      type

```

Drifter Initial Files

The drifter initial files (keyword = `drifter_in_file`) are set up by the user to provide initial condition and simultaion properties for drifters and particles.

Each line of the `drifter_in_file` should have three columns. The first column specifies a value; the second column gives the drifters that the value will be applied to (:, for all drifters, `n1:n2`, for drifters `n1` to `n2`); the third column is a keyword which specifies which property we are setting.

Valid keywords are:

<code>start_t</code>	starting timestep for drifter
<code>end_t</code>	end timestep for drifter
<code>solution_method</code>	type of drifter 1 = Drifter with constant elevation 2 = Drifter with constant depth 3 = Drifter with constant density 4 = Lagrangian Particle with constant elevation 5 = Lagrangian Particle with constant depth 6 = Lagrangian Particle with constant density 7 = 3D Lagrangian Particle
<code>beach_method</code>	1 = Remove drifter when it hits land 2 = Drifter continues if it hits land NB Drifters are always removed if they cross an open boundary
<code>cfl_max</code>	Maximum timestep for drifter (drifters are subtimestepped if $CFL > cfl_max$)
<code>drogue_density</code>	Density of drogue/particle for constant density solutions
<code>centre_of_mass</code>	Distance of centre of mass from drogue bottom, used for constant density solutions
<code>depth</code>	Depth of bottom of drogue/particle for constant depth solutions
<code>start_i, start_j,</code> <code>start_k</code>	Starting position of drifter in i, j, k coords eg: if $start_i = 8.5$, $start_j = 8.5$ and $start_k = 5.5$ the drifter will start in the center of the cell with ($i = 8$, $j = 8$ and $k = 5$)

`start_x, start_y,` Starting position of drifter in x, y, z coords
`start_z`
`start_cell` Starting position of drifter in cell number of 1D sparse array

An example file may look something like:

```

! - - - -
!! Example Drifter Initial File
!! NB each comment line should have 3 words
! - - - -
8.5   :   start_i
14.5  :   start_j
6.7   :   start_k
2     :   solution_method
2     :   beach_method
1.5   :   depth
  
```

.6.4 Drifter Update Files

The drifter update files (keyword = `update_drifter_file`) are set up by the user to provide updates to drifter positions or solution methods during a simulation. These files have primarily been designed for the updating drifter positions against real-time measurements. The format of drifter update files is exactly the same as normal update boundary condition files except instead of specifying boundary condition set number on the 3rd line a drifter number is given. The relevant drifter property will be updated for the drifter set for all timesteps between the first and second time column.

If the current timestep is not between the first and second time column of any row then no changes are made to the drifter.

Valid keywords are:

<code>solution_method</code>	see above
<code>beach_method</code>	see above
<code>cfl_max</code>	Maximum timestep for drifter (drifters are subtime-stepped if $CFL > cfl_max$)
<code>drogue_density</code>	Density of drogue/particle for constant density solutions
<code>centre_of_mass</code>	Distance of centre of mass from drogue bottom, used for constant density solutions
<code>depth</code>	Density of bottom of drogue/particle for constant depth solutions
<code>i, j, k</code>	New position of drifter in i,j,k coords
<code>x, y, z</code>	New position of drifter in x,y,z coords
<code>cell</code>	New position of drifter as a cell number in 1D sparse array

An example file may look something like:

```

! - - - -
!! Example Drifter Update File
!! NB each comment line should have 3 words
! - - - -
2 data sets
0 seconds between data
0         0         1   1
TIME     TIME     i   j
2005001.50 2005001.60 6.7 7.2
  
```

Jet Files

The jet files (keyword = `jet_file`) are set up by the user to provide properties of a jet or pump used for artificial destratification. These files require a format that consists of three header lines followed by four columns of data. The data supplied are values for particular instants of time that are applied for a particular jet. The file may have as many comment lines (preceded by `!`) at the top of the file as desired. The columns of data must be completely dense with data (i.e. there may not be "missing" data such that a row has less data than the number of columns).

The first header line contains one number: the jet reference number this is used for debugging purposes only.

The second line in the header must contain two numbers: the `i` and `j` location of the jet.

The third line in the header contains headers for the data column and must be

```
TIME THRUST RADIUS DEPTH
```

```
TIME THRUST RADIUS HEIGHT
```

The four data columns are	TIME	The date in CWR Julian Day format (yyyyddd.ddd)
	THRUST	Force of the pump in Newtons
	RADIUS	radius of the pump in metres
	DEPTH or HEIGHT	Either depth of the pump from the surface level or height from the bottom (metres)

```
! - - - -
! Example Jet File
! - - - -
1 jet ID number
6 12 i,j location of jet data
TIME      THRUST RADIUS DEPTH
2005001.0 3000  2.0   2.2
2005011.0 3000  2.0   2.2
```

3.7.7 Output definition file (Datablock.db)

The file `datablock.db` is a text file produced by the user, and contains the configuration data used to set up the users output. The name for this file is specified by the user.

A `datablock.db` file is structured as a series of blocks of instructions. Each block requires an exact number of lines, with specific instructions on each line. Each line contains a single instruction followed by a comment. The comment must begin with a `!` character. In manually editing `datablock.db` files, it is important to keep in mind that while blocks of instructions may be added or removed, lines within a block can only be modified, but not removed. The following will discuss the structure of each block of instructions.

Datablock Description

The first 4 lines of a `datablock.db` are the datablock description. In order, they are the block descriptor, the current datablock version number, the number of output groups, and the number of output sets. Output groups and sets are described below.

Group Description

Following the datablock description are the group description blocks. A group represents a group of data to be output. A data group is defined as an output filename, datatypes to be output (ie. `TEMPERATURE`, `SALINITY`, etc...), the location within the simulation domain of the points at which data is to be output, and the time at which data is to be output. If several groups are desired, the group descriptions must follow each other without breaks between them.

A group description block contains 11 lines, which are as follows:

- | | |
|--------------------------------------|---|
| 1. BEGIN GROUP_DESCRIPTION | begin block statement |
| group reference number | an integer labelling the group, this can have any value |
| 2. user label | a name containing no blank spaces labelling the group |
| 3. output filename | should include .unf suffix |
| 4. set reference number | an integer corresponding to a description of a set of points at which the data are to be output (sets will be described after groups) |
| 5. number of datatypes in group | an integer indicating the desired number of datatypes to be output (ie. TEMPERATURE, SALINITY ,etc...) |
| 6. should read .TRUE., do not modify | |
| 7. output interval | an integer indicating the desired number of timesteps between outputs |
| 8. output start time | an integer indicating which time step to begin outputting data |
| 9. output end time | an integer indicating which time step to stop outputting data, 0 indicates entire run |
| 10. should read 0, do not modify | |

Set Description

A set is a set of points at which data are to be output. This can include a vertical column of points, a contiguous series of column of points (curtain), a horizontal sheet of points, a collection of drifters, a collection of outflows, or any group of points in the three dimensional domain. The valid set types are

```
PROFILE_1D
CURTAIN_2D
SHEET_2D
DRIFTER
OUTFLOW
ALL_3D
GENERAL_3D
```

Profiles and curtains require only the (x,y) locations of the base of each column to describe their location. Sheets can be referenced to a layer, a height above user datum, a height above bottom, or a height below the free-surface. Drifters are referenced by their drifter number from 1 to `n_drifters`. `OUTFLOW` outputs total flow and average scalar concentrations for flow out of the domain. Outflows are referred to by their bc reference number in `bc.dat`. `ALL_3D` simply outputs all points in the domain, and `GENERAL_3D` at points whos coordinates are individually specified.

A set description block contains 16 lines, which are as follows:

- | | | |
|---------------------------|------------------------------|---|
| 1. | <code>BEGIN</code> | begin set statement |
| | <code>SET_DESCRIPTION</code> | |
| 2. set reference number | | an integer labelling the set, this can have any value |
| 3. user label | | a name containing no blank spaces labelling the set |
| 4. type of set | | on of the 5 valid set type names described above |
| 5. number of cells in set | | for type <code>PROFILE</code> this is the desired number of columns,
for type <code>CURTAIN2D</code> this is the number of columns making up the desired curtain
for type <code>DRIFTER</code> this is the number of drifters to output
for type <code>OUTFLOW</code> this is the number of outflow bc sets to output
for types <code>SHEET_2D</code> and <code>ALL_3D</code> this value is 0 |

6. sheet type	<p>for type SHEET_2D this indicates the SHEET datum, for all other set types the keyword is NULL. Valid sheet type keywords are</p> <p>LAYER horizontal layer referenced as layer number up from bottom as described in <code>bathymetry.dat</code> file</p> <p>HEIGHT horizontal layer referenced as vertical distance in metres up from datum as defined in <code>bathymetry.dat</code> file</p> <p>BOTTOM two dimensional surface defined as a vertical distance in metres up from the bottom</p> <p>SURFACE two dimensional surface defined as a vertical distance down from the free-surface (note: the location of this sheet will evolve in time as the free-surface evolves)</p> <p>FLAT a two dimensional sheet made up of calculated properties in each water column, the calculation type is specified below</p>
67 sheet calc	<p>calculation type for SHEET_2D, for all other set types the keyword is NULL. Valid sheet calcs keywords are:</p> <p>NEAREST Assigns each sheet value to the grid value vertically nearest the specified sheet height. Required for sheet types LAYER, HEIGHT, BOTTOM, and SURFACE. Not used with sheet type FLAT</p> <p>MAX Assigns each sheet value to the maximum value in a column. Used only with sheet type FLAT</p> <p>MIN Assigns each sheet value to the minimum value in a column. Used only with sheet type FLAT</p> <p>AVG Assigns each sheet value to the average value in a column. Used only with sheet type FLAT</p> <p>TOTAL Assigns each sheet value to the integrated value in a column. Used only with sheet type FLAT</p>
8. sheet value	<p>Layer number for sheet type LAYER. Vertical distance in metres for sheet types HEIGHT, BOTTOM, and SURFACE. 0 for sheet type FLAT.</p>
Lines 9-16.	<p>Not implemented, do not modify.</p>

4 Group data types

This block begins with the desired number of data types, followed a listing of the desired data types.

1.BEGIN GROUP_DATA_TYPES	Begin data type description
2. group reference number	An integer referencing the group number defined in the group description
3- data types	One data type descriptor per line. The possible data type descriptors are:

Output Datatype Keywords

<SCALAR>	(valid transportable scalar)	
U_VELOCITY	V_VELOCITY	W_VELOCITY
DENSITY		
EXTINCTION	F_PRESSURE	T_PRESSURE
RI	SHEAR	MIX_ENERGY
MIX_FRACTION	MIX_REGIME	
SHEAR_ENERGY	WIND_ENERGY	BBL_ENERGY
DISSIPATION	RETENTION_T	
HEIGHT	SLOPE_X	SLOPE_Y
WIND_SPEED	WIND_DIR	
WIND_U	WIND_V	
RAIN	REL_HUM	
LW_RAD_IN	LW_RAD_NET	SOLAR_RAD
ATM_PRESS	CLOUDS	AIR_TEMP
AUX_1	AUX_2	AUX_3
AUX_4	AUX_5	
EVAP_VOL_FLUX	EVAP_MASS_FLUX	EVAP_HEAT_FLUX
SENS_HEAT_FLUX		
HUMIDITY_ALT	HUMIDITY_SURF	
C_WIND	C_HEAT	

Underflow Variables

<SCALAR>_UND		
H_UND	U_UND	V_UND
DENSITY_UND	PRESSURE_UND	E_COEFF_UND
RI_UND	E_VOLUME_UND	E_VEL_XS_UND
XP_F_UND_INDEX	YP_F_UND_INDEX	
XN_F_UND_INDEX	YN_F_UND_INDEX	

For DRIFTER set types the value output is the value at the location of the bottom of the drifter

For OUTFLOW set types the scalar value output is the average value for the total flow out. NB: Scalar is actually estimated using a first order upwind scheme rather than the actual ULTIMATE-QUICKEST flux values.

5 Group Time List

This block is required to initialize a time stamp for each group. It requires only two lines as follows:

1. BEGIN GROUP_TIME_LIST Begin group time list
2. group reference number An integer referencing the group number defined in the group description

6 Set Cell Data

This block describes the spatial location of each datablock set. The locations are described as row, column, and/or layer number.

1.BEGIN SET_CELL_DATA	Begin set cell data
2. set reference number	An integer referencing the set number defined in the set description
3- cell locations	This input varies for different set types as follows: PROFILE Each profile requires a row of input. Each row requires 4 integers: 1) x-row, 2) y-column, 3) 0, 4) 0 CURTAIN_2D Each profile making up the curtain requires a row of input. Each row requires 4 integers: x-row y-column 0 0 SHEET_2D No Input DRIFTER The drifters to be output. Each row requires 4 integers: drifter number 0 0 0 OUTFLOW The bc sets to be output. Each row requires 4 integers: bcset number (from bc.dat) 0 0 0 ALL_3D No input GENERAL_3D Each point requires a row of input. Each row requires 3 integers: x-row y-column z-layer

3.8 Running ELCOM

ELCOM is executed at the command line by typing `elcom.exe` within the directory *where the `run_elcom.dat` and all the appropriate input data files* are located. The monitoring/debugging output is sent to the standard output of the computer (the display terminal on most computers). Unless the simulation run is very short, it is recommended that the user redirect the output to a file with a execution redirection such as:

```
elcom.exe >& run_output.txt
```

3.9 Post-Processing

3.9.1 Conversion of datablock file to NETCDF

The datablock output files are named and configured through the `datablock.db` file . ELCOM produces Fortran unformatted (binary) files as specified by the datablock output configuration. In general, one datablock output file contains one or more data types (e.g.velocity, temperature) applied to a set of points at a specified time interval. These files can be reprocessed into NETCDF files using the command-line program `dbconv.exe`

The program `dbconv.exe` is used for converting datablock unformatted by binary files created by ELCOM to NETCDF files which can then be processed and viewed in a program such as Matlab. The `dbconve.exe` executable should be placed either in the working directory or as recommended in the `/bin` directory which is in the operating system's search path. `dbconv` is run through a configuration file called `run_dbconv.dat`.

The `run_dbconv.dat` file contains 9 lines as follows:

```
Comment line, up to 64 characters
.unf file name and path
sparsedata.unf file name and path
.nc netcdf output file name and path
Start time step, -1 for beginning
End time step, -1 for end of file
Overwriting of .nc file? - clobber for overwriting, no_clobber for no overwriting
Output mapping info to text file? - map for writing, no_map for no overwriting
Output diagnostics to monitor? - echo to output to monitor, no_echo to not output to monitor
```

Once the `run_dbconv.dat` file is correctly configured, `dbconv` can be executed at the command line by typing:

```
dbconv.exe run_dbconv.dat
```

Notes: `run_dbconv.dat` can have any file name, however the correct file name must be invoked at run time. The `run_dbconv.dat` can only be configured for one file at a time.

For batch processing of several `.unf` files, it is recommended that a `run_dbconv.dat` file be created for each `.unf` to converted and `dbconv.exe` be invoked for each `run_dbconv.dat` using a batch file.

3.9.2 Save and Restart output files

Save and Restart output files are Fortran unformatted (binary) files that contain run configuration information used to allow the user to restart ELCOM runs from a certain point. As described in the `run_elcom.dat` section, the kernel name and output frequency of save and restart files are defined in the `run_elcom.dat` file.

The save files are created starting at the time step specified by the value assigned to the `run_elcom.dat` keyword `start_output_save`, and subsequently at intervals defined by `iter_out_save`. The file name of the save files begin with the user-specified kernel, given by `restart_save_file`, appended by a time stamp, appended by `.unf`.

The restart files are created at intervals defined by `iter_out_restart`, starting from the first time step. Subsequent restart files overwrite existing restart files. The file name of the restart files begin with the user-specified kernel, given by `restart_out_file`, appended by `.unf`. All start and restart files are written in the outfiles directory specified in the `run_elcom.dat` file.

To restart an ELCOM simulation using a save or restart file, the user must place a copy of the desired save or restart file in the infiles directory as specified in the `run_elcom.dat` file. In the `run_elcom.dat` file the name of the desired save or restart file must be assigned to the `restart_infile` keyword, and keyword `irestart` must be assigned the value 1.

3.10 ELCOM Valid Scalars

SALINITY	PSU	salinity (<i>practical salinity units which can be treated as parts per thousand within the numerical resolution of the model</i>)
WTR_TEMP	degC	water temperature for inflowing water
TRACER_1 - TRACER_10	concentration	tracer input
<CAEDYM>	concentration	CAEDYM Scalars