Connectivity Modeling System

User's Guide

CMS v 2.0

Authors: Claire B. Paris, Ana C. Vaz, Judith Helgers, Sally Wood, Rebecca Ross

March 2017

University of Miami

Rosenstiel School of Marine and Atmospheric Science

Department of Ocean Sciences

Paris' Lab: Physical-Biological Interactions Laboratory

Miami Florida 33149 USA

Contents

1. Introduction

Most coastal and benthic marine organisms have a pelagic larval phase, which might be their only opportunity to disperse with the currents and connect geographically separated populations. During the last decade, numerical models of the early life stages of marine organisms have emerged as a powerful tool for investigating the linkages and mixing between sedentary populations and the spatial history of successful migrants. These models are generally referred to as coupled bio-physical models. In such approaches, the early life of target species and their interaction with the environment is simulated by combining a stochastic biological model with ocean circulation models. These models typically use a Lagrangian particle-tracking framework to deal with explicit individuals, and use information on currents and environmental conditions from ocean circulation models to track the movement of a large number of individuals through space and time.

In many practical applications, several hundred millions of abiotic or biotic particles need to be simulated by coupling their varying traits with an ocean circulation model. In addition, many particles (larvae) can be advected with the currents along very long distances. Tracking particles for longer time periods, from shallow coastal areas to the deep ocean and back, is typically solved by a tradeoff between domain extent and spatial resolution. It becomes thus critical to use simultaneously several distinct circulation models to evaluate coastal and ocean scale conditions to improve the accuracy of Lagrangian predictions. These requirements are often computationally complex and very demanding, with large data management efforts that limit the scope of the applications.

Connectivity Modeling System

CMS is a community multi-scale biophysical modeling system, based on a stochastic Lagrangian framework. It has been developed to study complex larval migrations and give probability estimates of population connectivity. In addition, CMS can also provide a Lagrangian descriptions of oceanic phenomena (advection, dispersion, retention) and can be used in a broad range of applications, from the dispersion and fate of pollutants to marine spatial conservation.

To facilitate community contributions to CMS release, a Beta version was created and made available to external collaborators and developers on August 1, 2011. The Beta version underwent extensive application and evaluation. The final release was made in December of 2012 with CMS version 1.0. This user guide covers some of the new developments since that official release, in what is tentatively called v2.

CMS is open source and depends on the contribution of the user community for continuing development and bug fixes. Please report any bugs or suggested enhancements on the CMS github repository at github.com/beatrixparis/connectivity-modeling-system/issues. Better still, become a contributor and add your own fixes and improvements. There is also a user forum for advice, help and discussion: groups.google.com/forum/#!forum/connectivity-modeling-system-club.

CMS operates off-line using archived ocean velocity data, and includes the following modules designed to gain efficiency, flexibility, and applicability:

Parallel Module

CMS is a parallel implementation of a stochastic particle-tracking model and provides the computational efficiency of evaluating a full range of transport and fate variability. It runs in parallel ensemble simulations of millions of passive and/or active particles over large time scales.

Multiscale Nesting Module

Particles can be tracked seamlessly over a series of nested multiple nested-grid ocean model domains. The nesting is independent from the ocean modeling system (e.g., nest 1 can be HYCOM data, while nest 2 is ROMS).

Landmask Boundary Module

CMS is particularly apt for moving particles along complex topography such as steep slopes and convoluted coastlines, avoiding the boundary with a variable spatial interpolation scheme of the velocity field around the particle. Alternatively, CMS provides the option to stop integration of the particles at the boundary (i.e., make landfall).

Vertical Movement Module

CMS has great flexibility in vertical movement by simulating various behavior and transport processes: 1) buoyancy (particle size/density, fluid temperature/salinity), 2) ontogenic vertical migration, 3) diel vertical migration, 4) tidal stream transport, 5) upwelling and subduction, and 6) in any combinations of these types of vertical behavior in the terminal velocity of individual particles.

Connectivity Matrix Module

CMS is a true matrix-based model as it couples units of the benthic and coastal seascape (Seascape Module) to the Lagrangian Stochastic Particle Module (LSPM) and particle behavior (Biological Module) to generate a transition probability connectivity matrix as output of the ensemble simulation.

Flexible Input

CMS can access on-the-fly ocean circulation data via OPeNDAP (Open-source Project for a Network Data Access Protocol) protocol or access locally stored files of archived blocks of data. CMS is also model-independent and can be coupled to ocean circulation model data on Arakawa grids (A, B, and C-grid), in a Cartesian or tilted format.

Flexible Output

CMS is designed to speed up the post-process of both oceanographic and connectivity applications by providing two types of output data 1) connectivity matrix output (source/sink pairs), and 2) trajectory output (location/status). The model can produce output on NETCDF or ASCII formats. In addition, CMS has the option to generate a single file of ensemble simulations (e.g., oceanographic application) or as multiple files corresponding to different initial conditions (e.g., connectivity application).

About this User's Guide

This User's Guide describes the model architecture and provides information for the setup and implementation of the OSS package. The basic idea is that the user tells CMS where, when and how many particles he/she wants to release. In turn, CMS moves the particles using hydrodynamic fields and behaviors provided by the user, and outputs individual trajectories, fate, and a connectivity matrix of the ensemble of particles.

CMS is composed of two main executable programs to eliminate dependencies and duplications across the oceanographic and biological modules: 1) `getdata` accesses the ocean circulation data by reading the nest input files, and 2) `cms` moves and tracks the particles by reading the input parameter and configuration files. The CMS package also provides some Matlab® code for output visualization and matrix-based post-processing.

Section 2 of this User's Guide describes the general workings and algorithms of the advective code, Section 3 explains how to install the code and set up CMS for use. Sections 4 and 5 deal with setting up the parameters for the oceanographic data and for the IBM and other modules, respectively. Section 6 explains how to run the code and the outputs are described in Section 7. Finally, there are 3 supplementary sections: Section 8 briefly describes the individual files in the code, Section 9 gives a check-list of steps for running the CMS, and Section 10 provides a troubleshooting list of common errors and warnings.

For further help and discussion, feel free to join the CMS users forum (groups.google.com/forum/m/#!forum/connectivity-modeling-system-club).

Developers

The Connectivity Modeling System (CMS) is the result of an interdisciplinary effort at the University of Miami led by Claire Paris, Department of Ocean Sciences of the Rosenstiel School of Marine and Atmospheric Science (RSMAS), with the collaboration of Judith Helgers and Ashwanth Srinivasan from the Center for Computational Science (CCS) of the University of Miami, and Ana Vaz, Department of Ocean Sciences RSMAS.

Citation

We ask to make proper acknowledgement of the Rosenstiel School of Marine and Atmospheric Science, individual developers, agencies and funding agencies. One way to do this is to cite the following publication:

Paris CB, Helgers J, Van Sebille E, Srinivasan A (2013) Connectivity Modeling System (CMS): A probabilistic modeling tool for the multiscale tracking of biotic and abiotic variability in the ocean, Environmental Modelling & Software, http://dx.doi.org/10.1016/j.envsoft.2012.12.006.

Acknowledgements

We thank Matthieu Le Hénaff, Andrew Kough, Sally Wood, Dan Holstein, Erica Staaterman, Eric Van Sebille, and Karlissa Callwood for code testing and valuable suggestions. We thank the CCS for their help

in releasing the OSS of CMS. Funding was provided by the National Science Foundation Biological Oceanography Program (OCE-1048697, OCE- 0928423, OCE-0825625) to C. B. Paris.

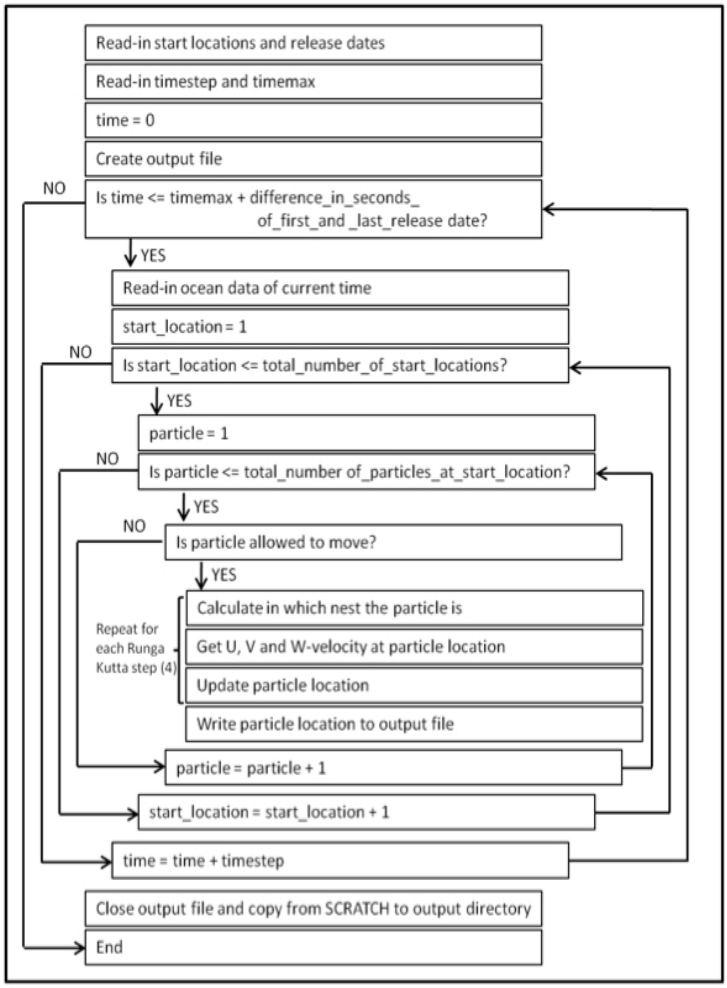
Open Source Licence

The CMS is an open-source model and licensed under the GNU Lesser General Public License. Here is a copy of the CMS model license file:

2. Algorithms

2.1 Flowchart

The basic flow chart of CMS looks as follows:



2.2 Multiple Nests

CMS can track the three-dimensional movement of particles across nested domains by using different sets of overlapping circulation model domains (see Fig. 1).

Motivation: In order to track the pathways of particles originating from shallow coastal areas or the deep benthic environment to the open ocean and back, it is important to use a series of multiple grids from ocean and coastal circulation models in a single application. Similarly, assessment of the transport requires that processes extending across oceanic scales and scales relevant to coastal dynamics be taken into account simultaneously. Requirements: the use of multiple nested-grid ocean models is useful if there is a high resolution circulation model domain overlapping a lower resolution domain. The “Multiple Nests” option requires that the models are at least one-way nested (i.e., the high resolution nest uses the boundary conditions of the larger domain); better performance can be achieved when models are two-way nested (i.e., information is exchanged between the nested-grid system).

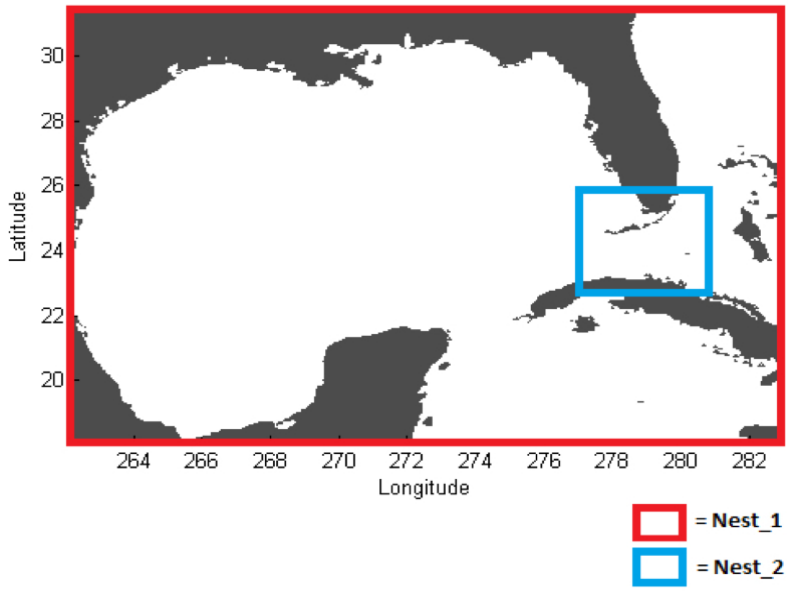


Figure 1. An example domain with two overlapping grids, which are called “nests” by CMS.

2.3 Integration Method

The movement of particles in CMS are integrated using 3D velocity field of ocean circulation models. The velocity field can present three components: U (the zonal component or x-direction), V (the meridional component or y-direction), and W (the vertical component or z-direction). In essence, the distance travelled by a particle is calculated by multiplying each velocity component by a user-prescribed time step. Here, we use a 4th order Runge-Kutta scheme to calculate particle advection, and this scheme is applied both in space and time. Each 4th order Runge-Kutta step uses four local velocities: one at the initial particle position, and three others at trial positions. From these velocities the final particle position is calculated (Fig. 2). Suppose a particle starts at location (xold, yold, zold) for longitude, latitude and depth, respectively. The following algorithm calculates the new position (xnew, ynew, znew) of the particle:

First step:
* Obtain u1, v1 and w1 from appropriated nest at position (xold, yold, zold) and time t

Second step:
* Calculate first trial position (xtmp1, ytmp1, ztmp1) from (xold, yold,zold) using u1, v1, w1 and 0.5*timestep
* Obtain u2, v2 and w2 from appropriated nest at position (xtmp1, ytmp1, ztmp1) and time t + 0.5*time step

Third step:
* Calculate second trial position (xtmp2, ytmp2, ztmp2) from (xold, yold,zold) using u2, v2, w2 and 0.5*timestep
* Obtain u3, v3 and w3 from appropriated nest at position (xtmp2, ytmp2, ztmp2) and time t + 0.5*timestep

Fourth step:
* Calculate third trial position (xtmp3, ytmp3, ztmp3) from (xold, yold,zold) using u3, v3, w3 and timestep
* Obtain u4, v4 and w4 from appropriated nest at position (xtmp3, ytmp3, ztmp3) and time t + time step

Final step:
* Calculate the final velocities uf, vf, wf:

```
uf=(u1 + 2*u2 + 2*u3 + u4)/6;
```

```
vf=(v1 + 2*v2 + 2*v3 + v4)/6;
```

`wf=(w1 + 2*w2 + 2*w3 + w4)/6`

* Calculate new position (xnew, ynew, znew) from (xold, yold,zold) using uf, vf, wf and at time t

The Fortran code of the Runga-Kutta method can be found in the module **move.f90**.

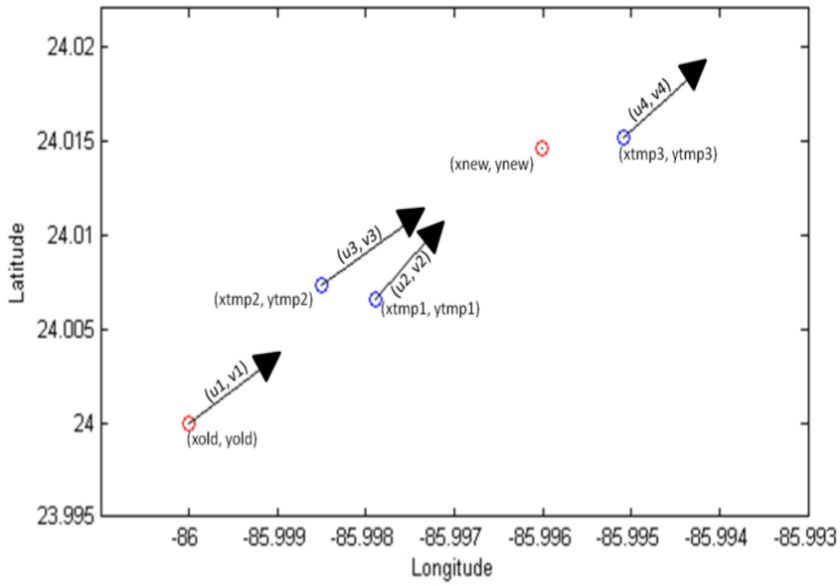


Figure 2. Example of the 4th order Runga-Kutta method in a 2D velocity field.

2.4 Interpolation Method for Position

There are two interpolation methods used to interpolate the water properties to the position of the particle, depending on the particle location on the model grid.

The CMS first choice is to use a tricubic interpolation (Fig. 3), where 64 neighboring nodes (4x4x4) are used. The tricubic interpolation is done for each dimension of the model output, by using a third-degree (cubic) polynomial. The cubic polynomial is fitted to the circulation model values found on the four closest grid points of each dimension. The resulting polynomial is then used to calculate the variable value on any point between grid points.

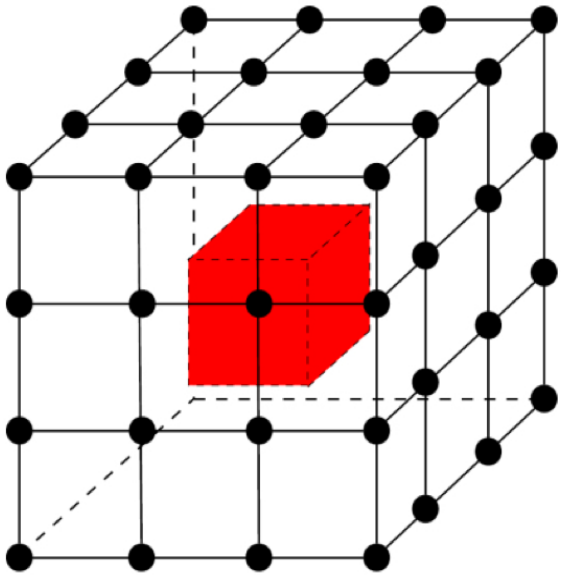


Figure 3. The 64 neighboring grid points used in the tricubic interpolation if the particle is located within the red area.

Note that, by definition, the tricubic interpolation can only be performed if the 64 neighboring points are located on ocean. If at least one of the points lies on land, then a trilinear interpolation will be used, where only 8 neighboring points (2x2x2) are needed (Fig. 4).

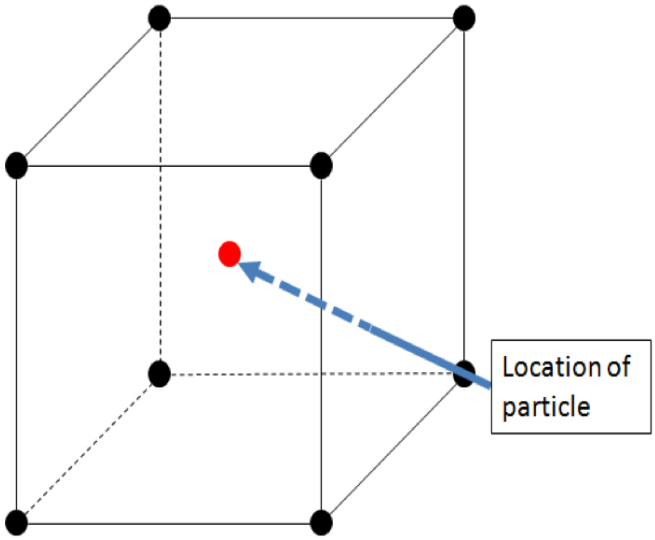


Figure 4. The 8 neighboring points used in the trilinear interpolation if the particle is located at the red dot.

Since CMS can be used for different applications, ranging from simulating abiotic particles to marine organisms, it is necessary that particles can have the ability to avoid being washed ashore. For example, fish larvae are able to avoid getting stranded on the bathymetry, while plastic or oil particles would reach the coast and be washed ashore or settle at the bottom. Thus, the user can choose if their particles will be able to avoid reaching land by setting up the flag avoidcoast in the input file runconfig.list. If this flag is set to false, particles will be able to reach land, and will be removed from the simulation if more than one of the 8 neighboring points required for the triliniar interpolation is located on land. However, if the flag is set to true, the particles will continue to move, by using the nearest velocities available to the particle (as will be described in Section 5.1).

If using 2D velocity fields (without a depth dimension), then a faster interpolation is used. In this case, CMS will perform a bicubic interpolation (Fig. 5) if the 16 nearest grid points to the particle are located on water. Otherwise, a bilinear interpolation is applied, for each only 4 neighboring grid points located on water are needed.

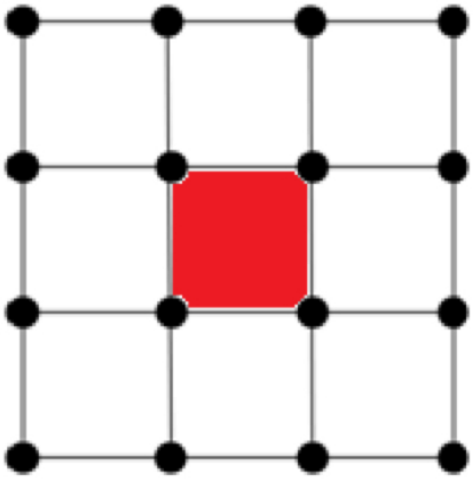


Figure 5. The 16 neighboring points if the particle is located within the red area.

The Fortran code for the interpolation can be found in the module **fldinterp.f90**.

2.5 Interpolation Method over Time

Oceanographic model files can be stored at different time intervals, thus it is necessary to interpolate the velocity files considering their time resolution, so the advection of particles on times in-between different oceanographic files can be adequately represented. If the velocity files are available at time intervals smaller than months (e.g., weeks, days, hours our seconds), CMS interpolates velocity fields from different files linearly over time.

For example, if a particle was advected for 35640 seconds since uvel1 was loaded (as marked by a red cross on Fig. 6), and the time interval among velocity files is one day (86400 seconds), the U-velocity at the present time-step will be given by:

```
uvel = 35640/86400 * uvel1 + (86400-35640)/86400 * uvel2
```

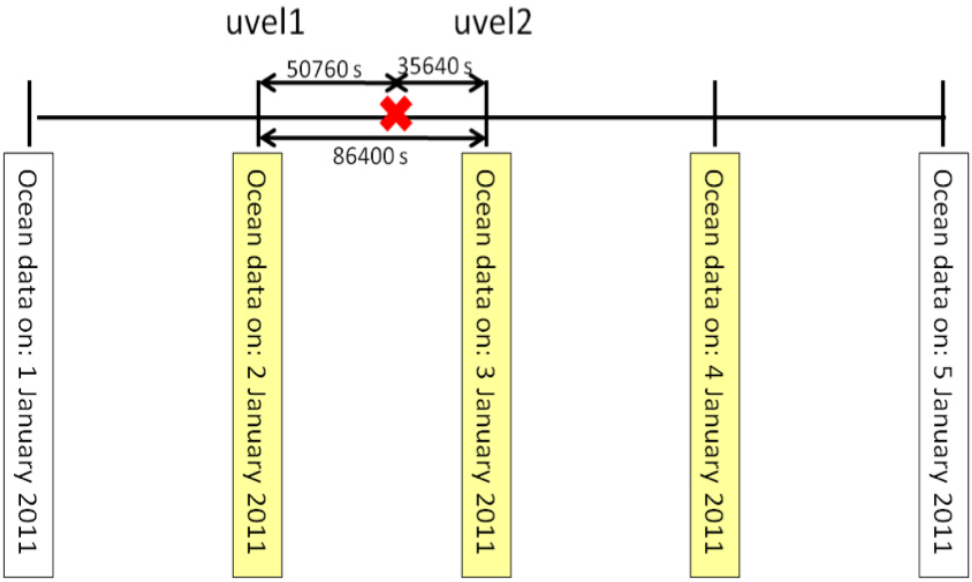


Figure 6. Example of the linear interpolation scheme for instantaneous velocity fields between consecutive snapshots

The Runge-Kutta method uses velocities from the present time and from future time-steps. Thus at each CMS time-step, three velocity files for each nest are stored in the computer's memory. In the example above, these three files would be for January 2, 3 and 4. If one of these files does not exist, CMS cannot move the particle, which will then be removed from the simulation with an exitcode of -5 in the trajectory output file (see Section 7 for details of output files).

If the nest files are stored in monthly intervals, then four files for each nest will be used by the CMS to interpolate oceanographic properties.

If the particle is on the time represented by the red cross on Fig. 7, then the U-velocity (u) at that time is given by:

```
x  = 20/31 (fraction of days advected)
x1 = 11/31 (fraction of days to advect until next file)
```

```
uvel  = (x1*(1+x*(1-1.5*x))*uvel2) + (x*(1+x1*(1-1.5*x1))*uvel3) + (-0.5*x*x1*x1*uvel1) +
(-0.5*x1*x*x*uvel4)
```

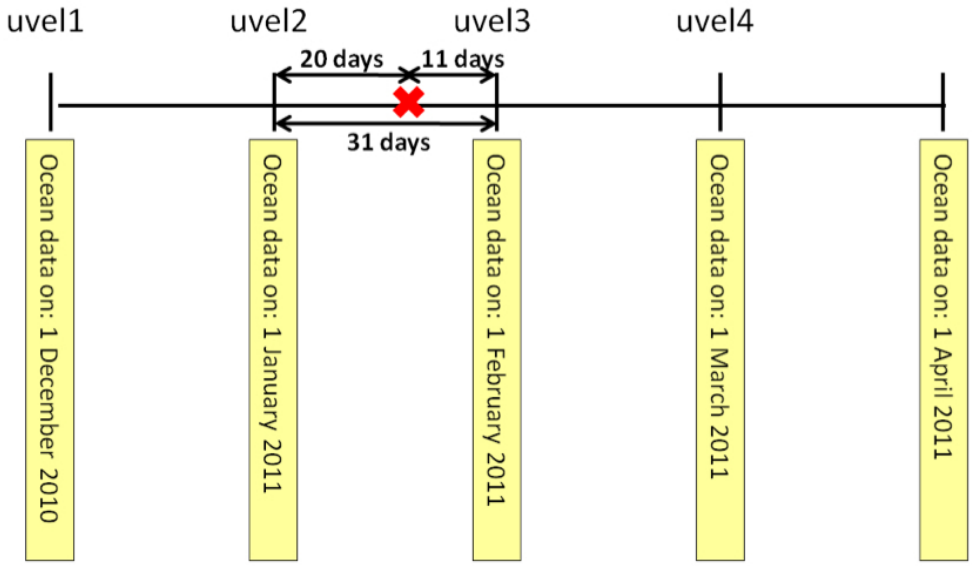


Figure 7. Example of the scheme used to interpolate between ocean velocity files stored at monthly intervals.

When using monthly output, CMS will store in memory five temporal files of each nest. Again, if one of the five files does not exist, the particle will be removed from the simulation with an exitcode of -5.

The Fortran code for the interpolation in time can be found in the module **getphysicaldata.f90**.

3. Installing and preparing CMS for use

3.1 Installing dependencies

Before installing CMS you have to have some supporting programs installed:

Fortran 90 compiler with MPI binding (mpif90):

Most of the CMS code is written in Fortran 90. The Fortran code uses the MPI Message Passing Interface so you need to install the mpif90 compiler. MPI is used to run the CMS on multiple processors. The MPI code that CMS uses is:

```
! MPI include file
use mpi
! initialise MPI
call MPI_INIT(ierr)
! how many processors are there in total?
call MPI_COMM_SIZE(MPI_COMM_WORLD, npes, ierr)
! what is the id of my processor?
call MPI_COMM_RANK(MPI_COMM_WORLD, my_id, ierr)
! quit MPI
call MPI_FINALIZE(ierr)
```

If you only have a Fortran 90 compiler that does not support MPI then it is not possible to run the CMS on multiple processors but you still can use the CMS. To use the CMS with a Fortran compiler without MPI remove the above MPI code in the file **src/cms.f90** (these lines are commented with "remove if not using mpi").

The CMS is tested with the mpif90, gfortran and ifort compiler.

C-compiler:

One file is written in C so a C-compiler is also required. The CMS is tested with the GCC compiler.

NetCDF version 4.1 or later with DAP support enabled:

CMS uses NetCDF (.nc) files as input and output, thus requires installation of the appropriate NetCDF libraries. The libraries can be found at: http://www.unidata.ucar.edu/software/netcdf.

CMS can also use OPeNDAP (Open-source Project for a Network Data Access Protocol) to download ocean model output. With the OPeNDAP software, you access ocean model outputs using a URL. Versions of netCDF 4.1 (or higher) enable access of data through OPeNDAP servers. For this, use the --enable-dap flag to enable DAP support when configuring netCDF. DAP support is automatically enabled if a usable curl library can be located using the config flag --with-curl-config, or it can be enabled using the --enable-dap flag, in which case you still need to provide access to the curl library.

After the above programs are installed and working properly, the CMS package can be copied to the computer.

3.2 Downloading the CMS package

The CMS software is stored on a Github repository at https://github.com/beatrixparis/connectivity-modeling-system. The CMS package is contained in the directory *cms-master* and comprises:

- The source code for installing and running CMS (directories */arch* and */src*)
- Example input files (*/expt/input_example*)
- Some Matlab files for post-processing (*/postproc*, see Section 7.3)

It is important that you make sure you download/check-out the latest version from this repository to ensure that the code has all the latest bug fixes and enhancements. Once you have downloaded the latest package to your system, you will be ready to install the CMS.

3.3 Installing the CMS

CMS itself consists of three directories:

- /arch: contains the file with the compiler options
- /src: contains the CMS source code
- /expt: the directory containing the experiments (input and output files, see 3.4)

Before use you will you need to 'build' the CMS in the directory */src*. Included with the source files is a file called *makefile*, which specifies how to derive the target program. This uses another file named *mpi_compiler* in the */arch* directory, which contains the compiler options. You will need to adjust this file to

your computer settings (as described below). After this configuration, you only need to use the command 'make', which automatically builds the executable programs and libraries from the source code.

3.3.1 Configuring the Makefile

Open the file *mpi_compiler* in the */arch* directory. If necessary, change the values of the following parameters:

```
FC: The Fortran compiler that is used to create the object files
FCFLAGS: The flags that the Fortran compiler needs to create the object files
CC: The C-compiler that is used to create the object files
CCFLAGS: The flags that the C-compiler needs to create the object files
LD: The Fortran compiler that is used to create the executable
    (In most cases this will be the same Fortran compiler as you filled in at FC)
LDFLAGS: The flags that the Fortran compiler needs to create the executable from the object files.
EXTRALIBS: The extra libraries needed to compile the program
```

3.3.2 Compiling the code

Go to the directory */src*, which contains the *Makefile* file. Type the command 'make' to compile the code and create all the object files and executables. If needed, use the command 'make clean' first to remove all existing object files and executables.

3.4 Setting up the run environment

Before you are able to run CMS, the first step is to prepare the necessary directories and input files. Here is a brief overview of the CMS file organization. Detailed information about inputs is given in Sections 5 and 6.

3.4.1 CMS directory structure

Input and output files for different experiments are stored in the directory */expt*. Within */expt*, each experiment should have a corresponding */input_name* and */expt_name* directory (as illustrated in Fig. 8). */input_name* stores the experiment configuration files (see Section 5), while */expt_name* stores the oceanographic input files (nest files, see Section 4) and the CMS output (the results of your experiment, see Section 7).

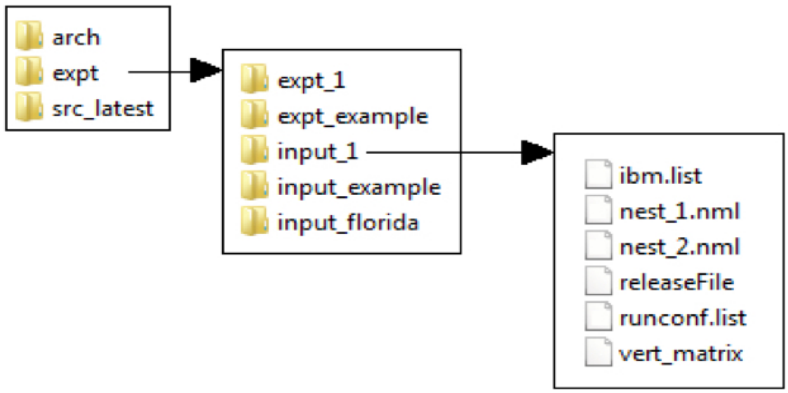


Figure 8. A tree structure for the the /input_name directory.

Input_name directory

There are 4 input files that must be present in the */input_name* directory to run CMS:

- 1) *nest_number.nml* (for example *nest_1.nml*): Configuration file for the oceanographic input (see Sections 4 and 6).
- 2) *runconf.list*: configuration file for for running CMS (see Section 5.1).
- 3) *ibm.list*: the file with the options for the particles (see Section 5.2).
- 4) The release file: contains the positions, times and depths where particles will be released (See Section 5.1).

These input files are created by the user. There are additional optional input files which are detailed in Section 5. Examples of all input files are provided in the directory */expt/input_example*, which can be copied and adapted.

Expt_name directory

The */expt_name* directory will contain 3 sub-directories:

- /nests: Containing the oceanographic input files (see Section 4).
- /output: Containing the CMS output files. This directory is created automatically on running *cms*.
- /SCRATCH : Containing output files that failed to finish. This directory should be empty if the run was successful (i.e, no errors were found).

These directories are created automatically by the program.

3.4.2 Creating links to executables

CMS has 2 executables:

1. `getdata`, which is used to download and format input oceanographic files for use by CMS (see Section 4). If you already have input files in the correct format, running `getdata` will not be necessary.
2. `cms`, which runs the connectivity model.

Once `getdata` has been run to prepare the oceanographic files, `cms` can be run multiple times using the same input files.

Before running `getdata` or `cms`, you need to create a link to the executable files in the */expt* directory, using the command:

```
ln -s ../src/cms
ln -s ../src/getdata
```

CMS is now ready to run from the */expt* directory.However, you will first need to set up your inputs, as will be described in sections 4 and 5.

3.5 Testing CMS has installed correctly

In order to check that your installation has worked, example input files are given in the CMS Github repository for you to run a simple cms test case: https://github.com/beatrixparis/connectivity-modeling-system/tree/master/cms-master/expt. Follow these steps following installation:

- 1) Make sure you are in the directory /expt and that you have created links to the executables cms and getdata as described above. You should see the directories *input_getdata* and *input_example*.
- 2) First run `getdata` to download the oceanographic files needed to run the test by typing:

  > ./getdata getdata_test 1

- 3) Then, create a link to the downloaded files (nest files) in the expt_example directory by typing:

  > ln -s ../expt_getdata/nests ./expt_example/

- 4) Then, run the `cms` by typing:

  > ./cms example

In this case, CMS is run passively for 3 particles released on 3 consecutive days, with no turbulence or IBM.

- 4) If the test has worked correctly, you should get an output file called **traj_file_1.nc** in the directory *expt_example/output*.
- 5) You can plot the output using the matlab script **draw_traj_netcdf.m** provided in the directory */postproc* and compare with the example output provided (e.g. see *expt/expt_example/output/traj_file_example.jpg*).

4. Oceanographic input files

Before you can run CMS you must obtain oceanographic input files, which will be prepared in a format specific for the CMS, and from then on referred to as '**nest files'**. The parameters for creating and reading the nest files is contained in the *nest_x.nml* input file (for example *nest_1.nml*) in the directory *input_name*, with one *nest_x.nml* file for each nest (if using multiple nests, see Section 2.2).The hydrodynamic files will be stored in *expt_name/nests*.

There are two ways to create the nest files:

1. You can use the program included in the CMS package `getdata` to either i) download and format files from an online server (see Section 4.3.1), or ii) format already downloaded files for use in CMS (see Section 4.3.2).
2. Alternatively, you can adjust the *nest_1.nml* file for the parameters of nest files that you have obtained from another source in their native format (see Section 4.4).

Using the *getdata* method is straightforward, and particularly useful if you wish to download files stored on online servers. However, it can only handle data on Arakawa A and B grids (see Section 4.1). Note that `getdata` will interpolate any data stored on B grids onto A grids.

The manual method is more versatile, as it doesn't require regridding the files to an Arakawa A grid and so allows for use of data on C grids. However, it is also slightly more difficult to set up, and can't be used to download files over OPeNDAP. The method is similar to the Local Files method (Section 4.3.2), but there are some small differences. These are explained in Section 4.4.

4.1 Raw oceanographic files

The native grid on which the CMS works is the Arakawa A grid, where all variables are given on the same location. However, CMS can automatically regrid B-grids to A-grids. When you use `getdata`, raw oceanographic files have to be in A-grid or B-grid and the flag **agrid** in the *nest_x.nml* input file has to be set to **.true.** (see Section 4.3.1).

You can also use oceanographic files with variables stored on C-grids in CMS. In this case, set the **agrid** flag in the *nest_x.nml* input file to **.false.**, and the files will be read directly by the CMS (not using `getdata`, as will be explained in Section 4.4).

Note that CMS does not allow you to use data stored as sigma levels (i.e. terrain following vertical layers). To use such data, you must first convert the sigma levels to z-levels (given by a specific depth).

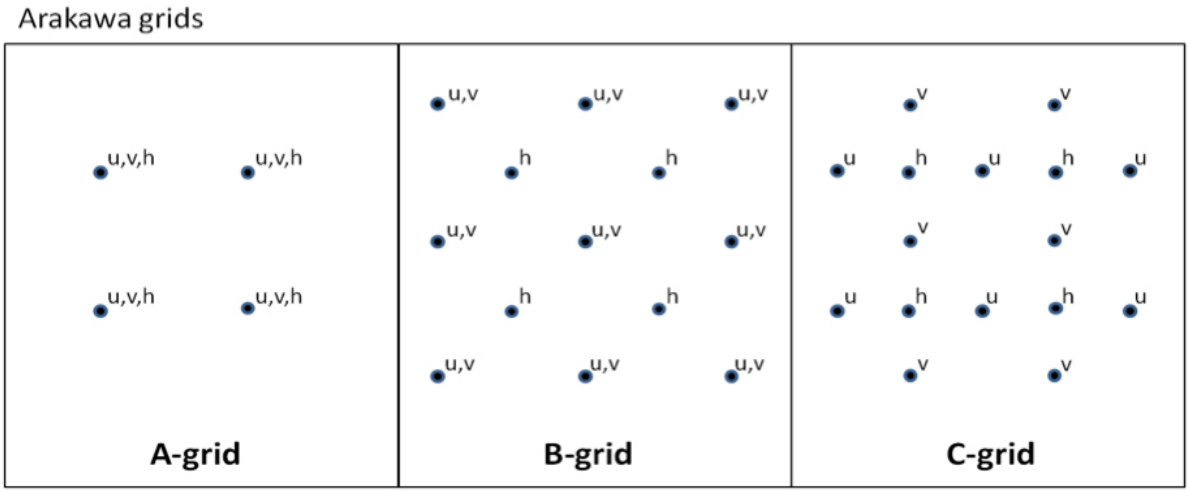


Figure 9. Three different Arakawa grids.

Raw oceanographic files (either on the OPeNDAP server or as local files) must meet the following constraints:

Longitude (required)

- unit: degrees
- difference between the smallest and largest values should be less than 360 degrees
- can present one or two dimensions, ordered as (Latitude, Longitude)
- must be in ascending order

Latitude (required)

- unit: degrees
- values varying between -90 and 90
- can present one or two dimensions, ordered as (Latitude, Longitude)
- if latitude is two dimensional, values must be in ascending order

Depth

- units: meters, centimeters or kilometers
- accepted as positive or negative
- must be ordered from surface to bottom

Time

- unit: seconds, days, hours or months from a specific date
- must present a regular time-step (CMS cannot handle missing data, so `getdata` will skip missing days, e.g. if the 1st Jan 2015 is missing, `getdata` will skip to the 2nd Jan and name that nest file 1st Jan 2015. This will be resolved in future releases of CMS)

U and V-velocities (required)

- if using four dimensions, must be ordered as (Time, Depth, Latitude, Longitude)
- if using three dimensions, must be ordered as (Time, Latitude, Longitude)

Optional fields:

- w-velocity
- salinity (in PSU)
- temperature (in degree Celsius)
- density (in kg/m^3 only)
- sea surface height (in meters)

If used they must be presented in the same structure as the U- and V-velocities.

4.2 Regridding two-dimensional files

The CMS code works with one dimensional longitudes and latitudes. When using latitudes and longitudes with two dimensions, `getdata` will regrid them to one dimension (as shown in Fig. 10). The Fortran code for re-gridding can be found in the module **mod_getdata.90**.

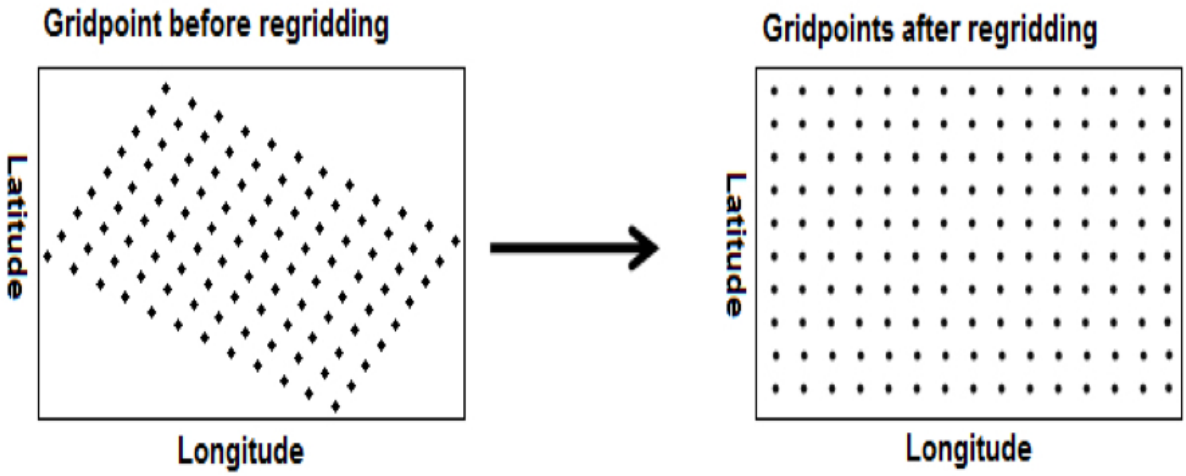


Figure 10. Example of the re-gridding done by `getdata` if the original grid is tilted.

4.3 Using getdata

Included with the CMS package is a program called `getdata`, which can download and save oceanographic files in formats suitable for CMS. You can use `getdata` to:

- download oceanographic files from a internet server using OpenDAP, or
- convert oceanographic files stored in your computer to the CMS format

To use `getdata` you need to create an input file with the parameters for downloading the oceanographic data (or with the parameters of the local files), as will be described below.

To run `getdata` type:

```
./getdata expt_name nest_number
```

> e.g.
>
> > ./getdata example 1

4.3.1 Downloading files using OPeNDAP

The program `getdata` can use OPeNDAP to access the ocean data. This requires that OPeNDAP was enabled at installation of netCDF (Section 3.1). The files will be downloaded as separate nest files in the directory */expt_name/nests/* (for example */expt_example/nests*)

The names of the downloaded files are:

```
nest_nestnumber_yyyymmddhhmmss.nc
```

Where:

```
nestnumber = the number of the nest (if using multiple nests, see Section 2.2)
yyyy = year
mm   = month
dd   = day
hh   = hour
mm   = minutes
ss   = seconds
```

> For example:
>
> > **nest_1_20101004010000.nc** is the oceanographic data for nest 1 for 10/4/2010 at 1am.

Nest_x.nml using OPeNDAP

The parameters for downloading the ocean data are stored in the *nest_x.nml* file in */input_name*, e.g.:

```
nest_1.nml
```

It is possible to use multiple nested-grids from ocean models, there is no limit on the number of grids. However, there has to be one *nest_x.nml* for each nested-grid domain.

The following is an example for nest_x.nml:

```
&nest_input
filename = "http://tds.hycom.org/thredds/dodsC/GLBu0.08/expt_91.1"
xaxis = "lon"
yaxis = "lat"
zaxis = "depth"
```

```
taxis = "time"
xstart= 270
xend  = 290
ystart= 20
yend  = 29
zstart= 0
zend  = 100.00
zaxis_positive_direction= "down"
tstart_yy = 2015
tstart_mm = 1
tstart_dd = 1
tend_yy   = 2015
tend_mm   = 2
tend_dd   = 1
time_step = 86400
time_units= "hours"
jdate_yy  = 2000
jdate_mm  = 1
jdate_dd  = 3
lon_name  = "lon"
lat_name  = "lat"
depth_name= "depth"
depth_conversion_factor=1
time_name = "time"
uvel_name = "water_u"
vvel_name = "water_v"
wvel_name = ""
wvel_positive_direction = ""
velocity_conversion_factor=1
dens_name = ""
temp_name = ""
saln_name = ""
ssh_name  = ""
angle_file= ""
fill_value= -30
agrid     = .true.
$end
```

Nest_x.nml input fields description:

```
filename
```

The url of the server where the oceanographic files are stored, written between double quotation marks. The CMS will use the OPeNDAP protocol (through `getdata`) to access the oceanographic files. If you want to read the oceanographic files directly from your computer (stored in *nests/raw*), then you would remove this line (see Section 4.3.2).

```
xaxis, yaxis, zaxis, taxis
```

The names of the file dimensions:

- xaxis: X dimension, e.g.: Longitude, lon, i, X
- yaxis: Y dimension, e.g.: Latitude, lat, j, Y
- zaxis: Z dimension, e.g.: Depth, Layer, Z
- taxis: T dimension, e.g.: Time, MT, T

The names of the dimensions can be found by typing in a web browser the name of the server where the data is archived with .html at the end.

> For example:
>
> > http://tds.hycom.org/thredds/dodsC/GOMl0.04/expt_20.1/2004.html
>
> will give:
>
> > u[MT = 1464][Depth = 7][Latitude = 361][Longitude = 437]
>
> thus:
>
> > xaxis = "Longitude", yaxis = "Latitude", zaxis = "Depth", "taxis" = MT

The names of the dimensions always need to be written between double quotation marks.

If there is no depth, the line with the zaxis has to be removed from the nest.nml file.

```
xstart,xend, ystart, yend
```

Give the range of the longitudes (x) and latitudes (y) to download your variables.

You need to make sure that these ranges are on the server or you will get the following error message: *'The data for longitude is not available. Choose a different value in the nest file'*.

To see which values there are on the server, type: name of the server plus .asc?lon_name.

> For example:
>
> > http://tds.hycom.org/thredds/dodsC/GOMl0.04/expt_20.1/2004.asc?Longitude

These 4 values define the boundaries of your nest area, for example:

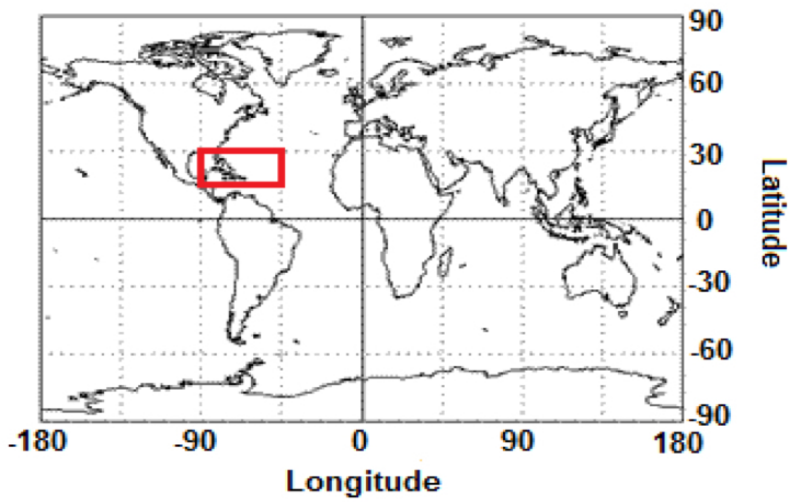


Figure 11. Area (red square) that will be downloaded if xstart = -90, xend = -45, ystart =15, yend =30. If xstart > xend then another part of the world will be downloaded (see Fig. 12)

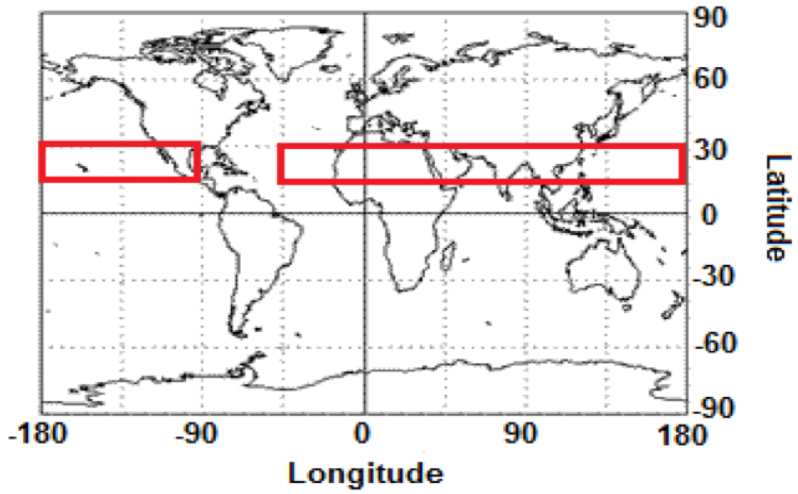


Figure 12. Area (red square) that will be downloaded if xstart = -45, xend = -90, ystart =15, yend =30

Longitudes can be given either as -180 to 180 or 0 to 360 degree formats.

```
zstart, zend
```

Depth range to download the oceanographic variables

Must be given in meters and ordered from surface to bottom. Thus zstart will always be smaller than zend.

If the depths are negative then use positive numbers for zstart and zend but set the parameter *zaxis_positive_direction* to *“up”*.

```
zaxis_positive_direction
```

Gives the direction of the depth axis. The only accepted values are *“down”* or *“up”*

If depths are positive (0,100,250,500): *zaxispositivedirection* = *“down”*

If depths are negative (-0,-100,-250,-500): *zaxispositivedirection* = *“up”*

```
time_units
```

Gives the interval between stored oceanographic files. Four values are accepted: *"months", “days”, “hours” or "seconds"*.

> For example:
>
> > http://tds.hycom.org/thredds/dodsC/GOMl0.04/expt_20.1/2004.htm
>
> you see that at the *time_name* it says:
>
> > *"units: days since 1900-12-31 00:00:00"*
>
> so this means that the *time_units* are *“days”*

```
time_step
```

The *time_step* is the time difference between oceanographic variables.

time_step is always given in seconds and must be uniform.

To see which values are on the server, type the name of the server plus .asc?time_name.

> For example:
>
> > http://tds.hycom.org/thredds/dodsC/GOMl0.04/expt_20.1/2004.asc?MT
>
> you will obtain
>
> > 39083.0, 39083.25, 39083.5, 39083.75, 39084.0, 39084.25,
>
> meaning that the oceanographic data is stored on quarter day intervals.
>
> As there are 86400 seconds in 1 day, the time_step in this example is 0.25*86400 = 21600.

If *time_units* is "months", then use a time step of 2635200.

```
tstart_yy, tstart_mm, tstart_dd, tend_yy, tend_mm, tend_dd
```

Duration of oceanographic files to be downloaded.

> For example:
>
> if
>
> tstart_yy = 2008,
>
> tstart_mm = 8,
>
> tstart_dd = 1,
>
> tend_yy = 2008,
>
> tend_mm = 8,
>
> tend_dd = 30
>
> Then oceanographic files will be downloaded from 1 August 2008 to 30 August 2008.

As CMS cannot handle missing data, `getdata` will skip missing days, e.g. if the 1st Jan 2015 is missing, `getdata` will skip to the 2nd Jan and name that nest file 1st Jan 2015. All subsequent nest files will then be named a day 'behind'. You can correct this manually by duplicating nest files to cover missing days.

```
jdate_yy, jdate_mm, jdate_dd
```

The time of the oceanographic file given by *time_name* is calculated from an “origin” date

> For example:

> Looking at:
>
> > http://tds.hycom.org/thredds/dodsC/flkeys.html
>
> you can see that at time_name it says:
>
> > "days since 1900-12-31 00:00:00"
>
> This means that jdate_yy = 1900, jdate_mm =12 and jdate_dd=31
>
> When using local data these three parameters are not used, and these three lines can be removed from the nest file (see Section 4.3.2).

```
lon_name, lat_name, depth_name, time_name
```

The names of the longitude, latitude, depth and time fields on the server.

If there is no depth in the data then the line with the *depthname has to be removed, or depthname* ="".

Names of variables always have to be written between double quotation marks.

```
depth_conversion_factor
```

Depths must be given in meters.

If the unit is different, the depths need to be recalculated to meters using the parameter *depth_conversion_factor*:

```
depth_new = depth_old * depth_conversion_factor
```

> For example:
>
> If the unit of depth is m then the *depth_conversion_factor* is 1.
>
> If the unit of depth is cm then the *depth_conversion_factor* is 0.01.

```
uvel_name, vvel_name, wvel_name
```

uvel_name: name of the zonal velocity component

vvel_name: name of the meridional velocity component

wvel_name: name of the vertical velocity

The names of the velocity variables need to be written between double quotation marks.

Velocities are necessary to move the particles, thus *uvel_name* and *vvel_name* must be given for CMS to run a simulation. The value of the *wvel_name* is optional, and if not using vertical velocity you can remove this line.

```
wvel_positive_direction
```

wvelpositivedirection is defined by either *"upward"* or *"downward"*.

the *wvelpositivedirection* is *"upward" *if a particle will move towards the surface with a positive w-velocity;

the *wvelpositivedirection* is *"downward"* if a particle will move toward the bottom of the sea with a positive W-velocity.

If not using vertical velocity you can remove this line, or leave the name blank (i.e. = " ").

```
velocity_conversion_factor
```

u, v and w velocities must be given in meters per second (m/s).

If the velocity unit is different, the velocities will be recalculated to m/s using the parameter *velocity_conversion_factor*:

```
velocity_new = velocity_old * velocity_conversion_factor
```

> For example:
>
> Unit velocity: velocity_conversion_factor
> m/s: 1
> cm/s: 0.01
> km/s: 1000
> mph: 0.44704

```
dens_name, temp_name, saln_name, ssh_name
```

The names of the density, temperature and salinity fields on the server, respectively;

Variable names need to be written between double quotation marks.

These are optional variables; you can remove the lines with the variable names that you do not want to download.

It is important to note that if density is not available, temperature and salinity fields are used to compute the particle's buoyancy. Thus if using the buoyancy or mass spawning flags (see Section 5) the nest files need to include these variables.

```
angle_file
```

If the grid of the original file has an angle with respect to the Cartesian grid, then the grid will be re-gridded to a straight Cartesian grid by using the original angle. The inclination (angle) of each grid point must be given in a netCDF file. The parameter *angle_file* is the name of this netCDF file (without the .nc).

```
fill_value
```

The value of the points with no velocity (for example land or landmask points).

`getdata` will replace the fill value of the downloaded data with the CMS value of 1.2676506E30.

```
agrid
```

The grid on which the data is stored (see Section 4.1). If the data is on an A or B grid, set **agrid** to **.true.**. `getdata` will automatically convert data on B grids to an A grid for use by CMS. If the data is on a C-grid, the files cannot be downloaded using `getdata`, and must be prepared by the user (see section 4.4).

Download.txt

In the same directory where the downloaded nest files are stored, there will also be a file called *download.txt*, which is created for each nest. This file saves a list of the nest files that were downloaded by `getdata`. This way, if there is a server crash while downloading the oceanographic files, `getdata` will start downloading only new files. If you wish to remove your nest files, remember to remove this file as well.

> For example, if you see the message: “Data file: 2007-4-25 06:00:00 already exists”, then this data file will not be downloaded again.

4.3.2 Formatting locally stored files with getdata

If you already have the oceanographic files stored on your computer (i.e. you do not need to download them using OPeNDAP), you can still use `getdata` to format them for use in CMS. In this case, you have to place the oceanographic files in the directory */expt_name/nests/raw*:

> For example: /expt_example/nests/raw.

It is important that the files are in NetCDF format and that the names of the files are in the same format as described above (*nestnestnumberyyyymmddhhmmss.nc*). So each file should only contain the data of one snapshot.

Running `getdata` will adapt the files for the CMS format. All modified files will be written in */expt_name/nests/* (for example */expt_example/nests/*).

If all your fields are in different files, you have to add an extra letter to the file to tell `getdata` which field is in the file.

> For example:
>
> > nest_1_20101004010000u.nc
> >
> > u = u-velocity
> >
> > v = v-velocity
> >
> > w = w-velocity
> >
> > s = salinity
> >
> > t = temperature
> >
> > d = density
> >
> > ssh =sea surface height

Nest_x.nml if using local files

The *nest_x.nml* file follows almost the same configuration as when using the OPeNDAP method, however, the line containing the filename has to be removed when using local files.

The fields *time_units*, *jdate_yy*, *jdat\e_mm* and *jdate_dd* are not used for local data, therefore they can also be removed.

4.4 Using your own nest files

As of CMS v1.1, users can also directly provide oceanographic data in the nests folder in their native format for CMS to read (i.e. not using `getdata`). An advantage of this method is that oceanographic files on any grid (A, B or C) can be used. In this case, if you are using a native B or C grid, set the **agrid** variable in *nest_x.nml* to **.false.**, and if using an A grid, set it to **.true.**

As on B or C grids the zonal and meridional components of the velocity are given at different locations, if using this method you will need to provide at least two nest files for each time step; one for u and one for v, named *nest_1_yyyymmddhhmmssu.nc and nest_1_yyyymmddhhmmssv.nc*.

The date format **yyyymmddhhmmss** is the same date string as used by `getdata` (see Section 4.3.1).

You can also add vertical velocity (w), temperature (t), and salinity (s) files as needed. So for each time step for which oceanographic files are available, you will need at least two and up to 5 different files. Each file will store only one field (u, v, w, t or s).

For each of your variables, you can define different variable names for longitude, latitude and depth. The names of the variable dimensions can be set in the *nest_x.nml* file as **lon_nameU**, **lat_nameU**, **dep_nameU**, **lon_nameV**, **lat_nameV**, **dep_nameV**, **lon_nameW**, **lon_nameT** etc.

The names of the variables themselves can also be set in *nest_1.nml* as **uvel_name**, **vvel_name** etc. If **agrid** is set to **.false.**, CMS will use these values to determine how to read the fields.

CMS uses the **tstart_yy**, **tstart_mm**, and **tstart_dd** parameters in *nest_1.nml* to determine when to start the run, so it is important to also provide these (see Section 4.3).

If **agrid** is set to **.false.**, CMS can also handle netcdf files with multiple snapshots in a file. The easiest way to get CMS to work with files that have, for instance, five snapshots, is to create five symbolic links to the same file, following the normal CMS calendar convention. Do this for every file. CMS will then automatically iterate through the individual snapshots in a file. It is important to note, however, that the **tstart** values (see above) point to the first snapshot of a file, and the **tend** values point to the last snapshot.

Currently, work is underway to generalize the grids that CMS can handle even further. As of yet, the grid needs to be “orthogonal”, meaning that the values for longitude have to be independent of latitude and vice versa. A common Mercator grid will work in CMS but more complicated tri-polar grids do not yet work. Use `getdata` if you want to work with these.

5. Other Input Files

5.1 Run configuration file: runconf.list

The runconf.list file contains the values for variables that are necessary to run the model. It also contains a list of options for turbulence, forward or backward advection mode, behavior of the particles at boundaries, and output format.

The following is an example of the file runconf.list:

```
&runconf
!============================================================!
nnests              = 1
timeMax             = 8640000   ! in seconds
timestep            = 2700      ! in seconds
outputFreq          = 43200     ! in seconds
releaseFilename     = "releaseFile"
!============================================================!
!Turbulence Module
turb                = .true.
horDiff             = 1         ! horizontal diffusivity (m2/s2)
vertDiff            = 0.5       ! vertical diffusivity (m2/s2)
turbTimestep        = 2700      ! in seconds
!============================================================!
!Periodic Boundary Condition
periodicbc          = .false.
!============================================================!
!Landmask Boundary Condition
avoidcoast          = .false.
!============================================================!
!Backward Tracking Module
backward            = .false.
!============================================================!
!Output files in ASCII instead of netCDF
ascii               = .false.
!============================================================!
!Flag to limit particle vertical movement to upper layer
upperlevelsurface   = .true.
!============================================================!
!Flag for looping through velocity files
loopfiles           = .false.
loopfilesstartyear  = 1980
loopfilesstartmonth = 1
loopfilesstartda    = 5
loopfilesendyear    = 2006
loopfilesendmonth   = 12
loopfilesendday     = 29
!============================================================!
!Options for saving restart files and restarting
```

```
writerestart        = .true.
restartfromfile     = .false.
restartwritefreq    = 432000    ! in seconds
!====================================================================!
!Options for mixed layer physics
mixedlayerphysics   = .false.
mixedlayerwmax      = 0.1
!====================================================================!
&end
```

NOTE if you are not using the optional modules (i.e. flags set to **.false.**), values must still be entered for their associated parameters (e.g. 0).

Description of runconf.list variables

Number of nests

```
nnests
```

The number of nests to be used

- For each nest there has to be a nest.nml file.
- **nnests** needs to be a positive integer number.
- There is no limit on the number of nests that can be used.

Time of simulation

```
timeMax (in seconds)
```

Defines how long particles from each release should move. For example, this parameter would represent the Pelagic Larval Duration (PLD) in larval dispersal studies.

- **timeMax** should be given in seconds.
- must be a positive integer number.

Simulation time step

```
timestep (in seconds)
```

The time interval between calculating the new particle position. This parameter determines the 'accuracy' or 'resolution' of the particle trajectories, i.e. how frequently along the particle path you sample the currents. This will involve a trade off between trajectory 'resolution' and run time.

- **timestep** must be given in seconds and be a positive integer number.
- Make sure that 86400s (1 day) divided by timestep is also an integer number.

Output frequency

```
outputfreq (in seconds)
```

Defines the time interval for saving trajectory information (longitude, latitude, depth, and other) to output (traj_files, see Section 7.1).

- Must be given in seconds.
- Must be a positive integer number.
- **outputfreq** should be equal to or a multiple of **timestep**.

File with release information

```
releaseFilename
```

The name of the file with the release information.

Without this file there is no simulation

The release file is a separate file containing the information to set up particle release, including the exact position (longitude, latitude and depth), time (year, month, day, seconds), and the number of particles to be released. It can also include the polygon where the particle is released (if using the flag **.polygon.**, see Section 5.2).

For example:

```
Polygon Longitude Latitude Depth  Number  Year  Month  Day  Second
1       277.2     24.6     1      10      2008  08     02   3600
10      277.4     24.6     1      20      2008  08     10   0
```

Make sure the following is correct:

- The depth is given in meters
- There must be oceanographic files (nest files) for the release date plus the maximum advection duration (**timeMax**)
- Even if you are not using polygons (see Section 5.2) then the first column is a number, e.g. this could represent the release event number

Longitudes can be given either as -180 to 180 or 0 to 360 degree formats.

Turbulence module

```
turb
```

A random component can be added to the motion of the particles to represent the subgrid-scale motion unresolved by the model. This “random kick” can be added by setting the flag **turb** to **.true.**. The turbulent velocity is added to the horizontal velocities (U and V-velocities), following the random walk or random displacement method (RDM) described by Okubo (1980, Diffusion and Ecological Problems: Mathematical Models. Springer, Berlin):

```
uturb = ((2*horDiff)/turbtimestep)^0.5 * random_number
```

- horDiff : horizontal diffusivity (m^2/s). You can enter a different horizontal diffusivity for each nest, by separating different values with a comma. This value should be scaled by the grid size of your model (e.g. see Okubo 1971, *Deep Sea Research* 18(8), 789-802).
- vertDiff: the vertical diffusivity (m^2/s). You can enter a different vertical diffusivity for each nest and separate the different values with a comma.
- turbtimestep: the time (seconds) after which the random component of the velocity is added to the particle movement.
- random_number: a Gaussian random number between 0 and 1 (calculated by CMS)

turbTimestep tells CMS how often the turbulent velocity is calculated. It can be defined as equal to or a multiple of **timestep**. If **turbTimestep** is not specified, CMS will use a **turbTimestep** equal to **timestep**.

If the flag **turb** is set to **.false.**, turbulence will not be added to the velocities and the model will run deterministically by advection only. In this case, a single particle can be released for each initial condition (IC).

Periodic boundary condition

```
periodicbc
```

Motivation: For global water mass transport studies where particles need to recirculate or for population connectivity studies in the Pacific, the particles need to cross the boundaries of the global nest file.

The flag **periodicbc** can have two values: **.true.** or **.false.**

If **periodicbc** is **true** then the x direction is periodic and a particle can loop around the nest in the longitude direction.

Avoid landmask boundary

```
avoidcoast
```

Motivation: Marine larvae of fish and of some invertebrates do not get stranded on the topography but are capable of swimming away. Alternatively, pollutants or organic matter can settle on the seafloor or be washed ashore.

When is a particle on land?

Before explaining how CMS prevents particles from 'stranding', it is necessary to define when a particle is on land. The land gridpoints always have the same U, V and W-velocity values, defined as the fill value. CMS checks the surrounding eight points of the location of the particle in a trilinear interpolation (see Fig. 3)

The variable **countLand** stores how many of the eight grid points are land points, so can have a value between 0 and 8. CMS also defines the distance from each of the eight grid points to the location of the particle. A particle will be considered on land if the closest grid point is on land, or if more than 2 points are located on land.

How does CMS prevent a particle from stranding?

If the new position of a particle is on land and the flag **avoidcoast** is set to **.true.**, then CMS will find a new position located on water for this particle, following the steps outlined below. If the particle is still on land, CMS will move on to the next step. If a particle is on water after a step, this will be the new position of the particle.

- CMS places the particle back to its old location and moves the particle with U and W-velocity.
- CMS places the particle back to its old location and moves the particle with V and W-velocity.
- CMS places the particle back to its old location and moves the particle with U and V-velocity.
- CMS places the particle back to its old location and does not move the particle.

How does CMS make it possible for a particle to reach land?

If the flag **avoidcoast** is set to **.false.**, if a particle reaches the land-mask at the coastline or anywhere on the topography, it will stop moving and exit with an exitcode = -2 in the trajectory output file (see Section 7.1); the model will continue to run with the next particle.

Backward tracking module

```
backward
```

Motivation: This module is useful to estimate the origin of particles, e.g. the spawning locations of larvae caught in plankton studies.

If **backward** is set to **.true.**, then particle displacement will be calculated by integrating the velocity field backward in time, answering the question “Where did a particle come from?”

If **backward** is **.false.**, then the trajectories are moved forward in time, answering the question “Where did a particle go?”. This is the CMS default mode.

To calculate the backward trajectories CMS reads the nest files in opposite order and changes the sign of the velocity components. When using the backward module, you will not be able to use **buoyancy**, **massSpawning**, and **tidalMovement** (see Section 5.2). If you wish to use the ontogenic vertical matrix migration with backward trajectory, you need to reverse the vertical matrix.

Output files in ASCII

```
ascii
```

CMS standard format output is netCDF. If ascii is set to true in the file runconf.list then output files will be in ascii format (see Section 7 for more details).

Prevent particle from dying at the uppermost level

```
upperlevelsurface
```

In the default CMS mode, the vertical movement of particles is confined to the uppermost layer (typically the sea surface), preventing particles from “flying into the air.”

If, however, particles should be killed when reaching surface, the flag **upperlevelsurface** has to be set to **.false.**.

Loop through the nest files

```
loopfiles
```

Motivation: In certain applications it is necessary to advect particles for periods of time longer than oceanographic files are available. In its default mode, CMS remove particles from the simulation when there are no more nest files (exitcode -5, see Section 7.1).

By setting **loopfiles .true.**, particle advection will continue until the maximum integration time. CMS will loop through the available files, using the first oceanographic file once it gets past the last file.

Create restart files

```
writerestart
restartwritefreq
restartfromfile
```

If writerestart = **.true.**, CMS will create a binary file with all the information necessary to restart a run for passive particles, that is, not using the *ibm.list* options (all options in ibm.list set to **.false.**, see section 5.2). CMS will save this file at a frequency specified by the variable ***restartwritefreq***. The restart files are written in the SCRATCH directory.

To restart from a saved restartfile, set the **restartfromfile** variable to **.true.**.

Note that the traj_files will now be overwritten with new data from the restart time, so if you need to keep the old output you must first move these files from */expt*.

Mixed layer parameterization

```
mixedlayerphysics
```

If information on the depth of the mixed layer is available in the oceanographic nest files, CMS can randomly move particles that are above that depth vertically through the mixed layer at each time step. Set the parameter to **.true.**.

The variable **mixedlayerwmax** defines the maximum vertical velocity that a particle can attain in the mixed layer.

5.2 Individual Based Model: ibm.list

ibm.list is an input file that allows you to give particles specific behaviors. If using CMS on passive-particle mode (physical applications), you can delete this file from your input directory.

```
&ibm
!===============================================================!
! Buoyancy Module
Buoyancy         = .false.
buoyancyFilename = "buoyancytest"
!===============================================================!
! Seascape Module
polygon          = .false.
polyFilename     = "Caribbean.xyz"
settlementStart  = 160            !in days
!===============================================================!
! Probability Matrix of Vertical Migration
ibio             = .true.
ibioFilename     = "vert_matrix"
ibioTimeStep     = 259200        !in seconds
!===============================================================!
! Mortality Rate
mort             = .false.
halflife         = 90000         !in seconds
!===============================================================!
! Different Particle Attributes Module
diffpart         = .false.
diffpartFilename = "diffpart_matrix"
!===============================================================!
! Combined Buoyancy and Vertical Migration
massSpawning     = .false.
larvaStart       = 7             !in days
!===============================================================!
! Selective Tidal Stream Transport
tidalmovement    = .false.
tstStart         = 14            !in days
!===============================================================!
! Adding strata for 3D polygons
strata           = .false.
strataFilename   = "strataFile"
!===============================================================!
! Output temperature/salinity in netcdf file
outputtemp       = .false.
outputsaln       = .false.
!===============================================================!
$end
```

NOTE if you are not using the optional modules (i.e. flags set to **.false.**), values must still be entered for their associated parameters (e.g. 0).

Buoyancy Module

```
buoyancy
```

Motivation: Transport of eggs and early larval stages is considered to be one of the major factors impacting recruitment success. The eggs of most marine fish are positively buoyant at time of release. How millions of eggs are initially transported and where they end up before hatching depends on their buoyancy, flow stratification and turbulence. The terminal velocity of particles (eggs) is computed as the force balance between the particle specific density, water density and the kinematic viscosity.

So, besides using vertical velocities to move particles up and down, CMS can also add the particle buoyancy to the particle velocity.

To move particles taking their buoyancy into account, set the flag **buoyancy** to **.true.**. You will need to specify the name of file which stores the particle diameter and density information at **buoyancyFilename**.

The diameter and density of eggs also changes with time, so you can define these characteristics at time discreet intervals. Also, variability in size and density of eggs can be observed for the same species (or even the same breeder), thus it is possible to set a range of densities and diameters to be used in your experiments.

The buoyancy depends on the particle's attributes and also on the environmental conditions, so your nest files must contain either temperature and salinity (then used to calculate water density by the model), or density fields.

The buoyancy input file should have the following structure:

```
4
432000 864000 1728000 2160000
1010 1015 1020 1027
1012 1017 1022 1029
0.00025 0.00020 0.00015 0.00010
0.00027 0.00022 0.00017 0.00012
```

- Line 1 = number of columns, which represents the time steps at which the particles properties will be changed
- Line 2 = time (in seconds) that each property column is valid
- Lines 3 and 4 = lower and upper boundaries of the particle density (kg/m^3)
- Lines 5 and 6 = lower and upper boundaries of the particle diameter range (meters)

To simulate the transport not considering a range of values, simply add the same values for the lower and upper boundary. To use the same values for all time steps, create a file with one column, with a time interval equal to your simulation maximum time, e.g.:

```
1
2160000
1010
1010
0.00010
0.00010
```

The velocity determined by buoyancy is calculated by the Stoke's formula, which is then added to the W-velocity (from the velocity field):

```
V_t = w + (9.81*diam_particle2 *(dens_particle - dens_water))/(18*viscosity)
```

Where:

- diam_particle: the particle diameter (meters)
- viscosity: the seawater viscosity computed with the following formula:

  ```
  1.88e-3 - (T*4.e-5)
  ```

  where T is the temperature of the water (therefore temperature must be present in the nest files).

- density_particle: density of the particle (kg/m^3)
- dens_water: the density of the seawater (kg/m^3). If density is not provided in the nest files, CMS will calculate the water density from the temperature and salinity using the following formula:

  ```
  c1=-1.36471E-01, c2=4.68181E-02, c3=8.07004E-01, c4=-7.45353E-03, c5=-2.94418E-03,
  c6=3.43570E-05, c7=3.48658E-05

  density = (c1+c3\*saln+temp\*(c2+c5\*saln+temp\*(c4+c7\*saln+c6\*temp)))
  ```

The coefficients can be changed in the Fortran module **util.f90**.

Seascape Module

```
polygon
polyFilename
settlementStart
```

Motivation: Polygons are the representation of marine habitat, describing the location of both the spawning and nursery areas. Inside their boundaries two essential processes will be simulated: release and retention. Therefore, the polygons must be carefully defined accordingly to the region, species and processes you are intending to study.

By setting the flag **polygon** to **.true.** you can add marine habitat. Polygons are described in a separate polygon file, the name of which is given by the variable **polyFilename**. This gives the coordinates of the polygon vertices. There can be an unlimited number of vertices for each polygon, depending on it's shape (the simplest being a 3-point triangle). A single polygon in the polygon file looks as follows:

```
Longitude   Latitude    Polygon
277.2766    24.6665     1
277.2715    24.6235     1
277.2507    24.5921     1
277.2243    24.5786     1
277.1763    24.5777     1
277.1600    24.5564     1
277.1417    24.5851     1
```

```
277.1531    24.6684    1
277.2766    24.6665    1
```

- Polygon numbers must start from one, and be consecutive and in ascending order.
- If using strata (3D polygons) you will need to add a 4th column with the depth of each polygon vertex (as will be described in the 'strata' section, below)

Particle settlement in the habitat represented by polygons will start once particles reach the age defined by the variable **settlementStart** (given in days).

How to create the polygon file

Polygon files can be generated in a GIS-based software. You need to begin with projected habitat location data. Then create a vector grid to overlay the habitat data. Set the grid size you want for the size of the polygons. Perform a spatial join with the grid and reef polygons. The next step is to remove all polygons which fall completely within the land mask (Fig. 13).

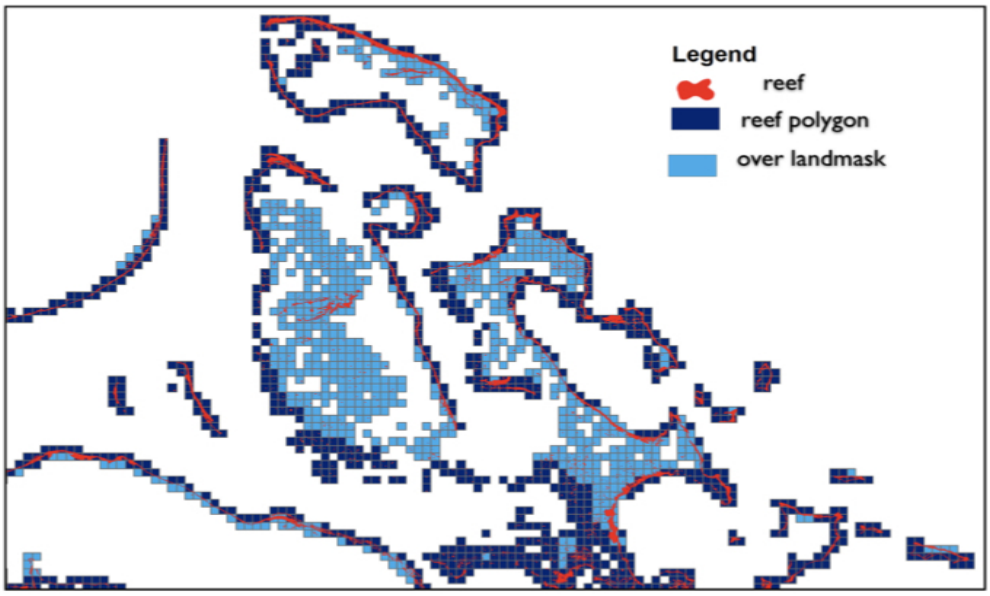


Figure 13. An example of 8km x 8km grid polygon in the Bahamian Archipelago.

Another way to create polygons is to create a buffer around the habitat and split the resulting polygons in equal segments using a tension factor representing the scale of the unit polygons (Fig. 14).

Assign each polygon a unique number which much match the release locations.

Finally, to obtain the latitudes and longitudes of your polygon vertices for input into the CMS, convert the polygons to points. This gives you a point feature layer with the vertices for each polygon preserved with their original polygon number. The attribute table can be exported directly into a text file.

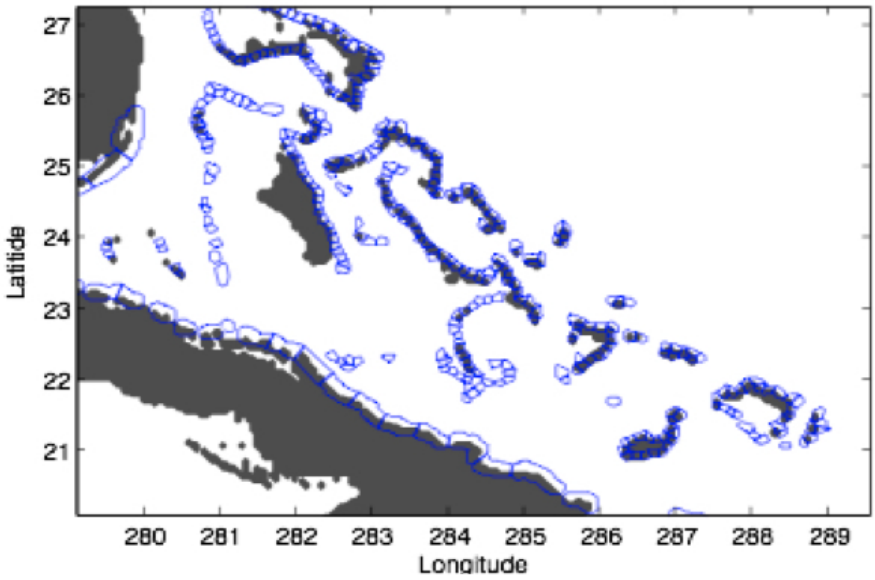


Figure 14. Example of a non-grid polygon file composed of irregular ca. 5km x 10km units for the Bahamian Archipelago and ca. 9km x 50km units for North Central Cuba.

How to create release points from polygons

You could find the centroids of the polygons to ensure each release point is within the polygon itself. This should ensure all your release points are in the 'ocean'. However any position within the polygon boundary will be adequate, and you can have more than one release position per polygon. If you wish to release from outside of a polygon while the **polygon** flag is set to **true**, the polygon number in the releaseFile must not be a number used as a polygon number in the polygon file. A warning will be displayed for this polygon number when you run CMS but it will still run.

Vertical Migration

```
ibio
ibioFilename
ibioTimestep
```

Motivation: Most marine larvae undergo ontogenetic vertical migration, whereby they swim or adjust their buoyancy to move downward from the upper part of the water column to deeper layers or reversely, upward into the neuston as they age. Vertical distribution patterns can be observed via plankton surveys and can be statistically described as stage-specific probability density distributions with depth. This imprinted behavior is species-specific and the center of mass and vertical spread of a particular organism can be constrained by varying environmental conditions; yet on average the observed distributions are typically robust.

To add vertical migration, set the flag **ibio** to **.true.**. CMS will then move particles vertically by using a probability matrix, which should represent the observed distribution of larvae through ontogenetic or diel vertical migration.

The probability matrix is described in a separate file, the name of which is described by the variable **ibioFilename**. The probability matrix should look as follows:

```
11
7
10 20 30 40 50 60 70
86400 86400 86400 432000 432000 432000 259200 259200 259200 259200      259200
20 10 10 00 00 00 00 00 00 00 00
20 10 10 05 05 05 05 00 00 00 00
20 10 10 05 05 05 05 00 00 00 00
40 70 60 20 20 20 20 10 10 10 10
00 00 10 40 40 40 40 50 50 50 50
00 00 00 20 20 20 20 30 30 30 30
00 00 00 10 10 10 10 10 10 10 10
```

- Line 1 = number of columns (x), representing how many times the vertical distribution of larvae will change along a simulation
- Line 2 = number of rows (y), representing the number of depth levels used
- Line 3 = maximum depth of each row in meters
- Line 4 = number of seconds each column of vertical distributions is valid
- Lines 5 to 5+y-1 = array with probabilities (x by y). The sum of probabilities for each column should be equal to 100

Each row (y) in the array represents a depth and each column (x) corresponds to a time interval. The value at each point (x,y) refers to the percentage of larvae that can be found at that depth at that time interval. The model randomly chooses based on the matrix what the depth of a particle is.

Particles will change vertical layers at the period defined by **ibioTimestep**. The **ibioTimestep** can be equal or an integer multiple of **timestep** (given in *runconf.list*). If **ibioTimestep** is not defined, CMS will use **timestep** instead.

If you are using **ibio**, you are not able to use **buoyancy** (which will be ignored).

Mortality Rate

```
mort
halflife
```

Motivation: Larval mortality is an important parameter that can change the dispersal distance of an organism, influencing the structure of the metapopulation. Larval mortality rates are not constant throughout larval life and larvae acquire and lose competency at varying times/ages. As a result, the pelagic larval duration (PLD) is often plastic. Similarly, abotic particles may have important fate processes that need to be taken into account during dispersion. Here we introduce a simple half-life mortality function that can be further modified to match experimental/observational survivorship curves.

If the flag **mort** is set to true, then a particle can die. The mortality rate of particles is based on an exponential decay formula, following their half-life time.

Half-life is the period of time it requires for a quantity to fall to half its value. For larvae, it is the time it takes for a larval cohort to be reduced by half; for a chemical compound undergoing decay, it corresponds to the time it takes to decrease by half. The decay constant is given by:

```
lambda = ln(2) / half-life
```

This equation can be changed in the file **mod_mort_larva.f90**.

The half-life time of the particle is entered in *ibm.list* as **halflife**, in seconds.

Different Particle Attributes Module

```
diffpart
diffpartFilename
```

Motivation: this module allows defining variable particle attributes such as size, density, and mortality rate observed in the early life history traits. Similarly, this can be used for the dispersion of abiotic particles with varying characteristics and fate.

If **diffpart** is set to **.true.** then the particles can have different sizes, densities and half-life times. This is only useful if you are adding buoyancy and/or mortality.

If **diffpart** is **.true.** then the values of the parameters **diam_particle**, **density_particle** and **half_life** in *ibm.list* will be ignored.

The different attributes are described in a separate diffpart input file, the name of which is given by the variable **diffpartFilename**. The diffpart file looks as follows:

```
3
33  33  34
820 840 860
840 860 880
2e-6    2e-4    1e-3
2e-4    1e-3    1e-2
86400   432000  900000
```

- Line 1: number of different categories (x).
- Line 2: the x probabilities that a particle sits in a particular category.
- Line 3: the x different minimum densities.
- Line 4: the x different maximum densities. The density of a particle will be a random number between the minimum density and maximum density.
- Line 5: the x different minimum sizes.
- Line 6: the x different maximum sizes. The size of a particle will be a random number between the minimum size and maximum size.
- Line 7: the x half life times in seconds.

Combined Buoyancy and Vertical Migration (mass spawning)

```
massSpawning
larvaStart
```

Motivation: Many organisms gather in large aggregations for synchronized mass spawning. These gatherings produce enormous numbers of gametes concentrated in space and time that disperse initially

as a cloud of buoyant particles, become less buoyant after fertilization, and increase in specific gravity prior to hatching into swimming larvae.

This module resolves the transition from the egg to the larval stages. When using the flag **massSpawning**, particles will move in the vertical sequentially using first buoyancy and then vertical migration.

Initially, particles will move in the vertical following the parameters defined by **buoyancy**.

When eggs reach the hatching age, which is defined by **larvaStart** (in days), larvae will then move vertically following the probabilistic vertical migration matrix, given by **ibioFilename** (see Vertical Migration, above).

Selective Tidal Stream Transport

```
tidalMovement
tstStart
```

Motivation: Many estuarine fish and invertebrate species undergo vertical migrations that are coordinated with phases of the tide in order to achieve horizontal movement. This general mechanism is known as Selective Tidal Stream Transport (STST), and more than one behavior has been associated with it. A flood-tide transport occurs when organisms use the flood phase for shoreward transport and immigration to estuaries. An ebb-tide transport or ebb-phased migration is used for seaward transport and out-migration from estuaries.

If **tidalMovement** is set to **.true.**, particles will only move if the sea surface height is rising. When particles do not move, their depth in the trajectory output file (Section 7.1) will be set to -999.99.

The days after which the STS movement starts is given by the value **tstStart** (in days).

To use **tidalMovement**, the variable sea surface height will need to be present on the nest files. The rise or decrease of the SSH is given by the gradient between the SSH from the two nest files closes to the current time (before and after). This rule can be changed in the source code.

3-Dimensional polygons

```
strata
strataFilename
```

Motivation: Marine organisms can be distributed over large ranges of depths, however, their young might need specific conditions met at a smaller range of depths to survive. Also, understanding how habitats located at different depths are connected is becoming a central challenge for the demographic quantification of populations and the design of conservation measures.

To account for discreet 3D habitats a new module was added to CMS which allow users to use 3D polygons and control depths at which settlement will take place.

When using the flag **strata**, polygons will be defined at different vertical strata in the water column, and larvae will settle at the nearest polygon after their competency period.

- The strata boundaries are flexible and defined by the user using the strata file, given by **strataFilename**.
- The depths of polygons must be present in the polygon file (given by **polyFilename**) to use the strata option.
- The spawning and settlement strata will be recorded in the connectivity output file when using this option (see Section 7.2).

The strata file has the following structure:

```
6
0 21 41 61 81 101
```

- Line 1: number of vertical strata to be used + bottom boundary of vertical strata.
- Line 2: initial depth of each vertical strata (meters).

The maximum depth of each strata is assumed to be the next strata initial limit - 1 meter. The last strata depth represents the bottom boundary of vertical strata: larvae found deeper than this value will therefore not settle.

> In example file above, there are 5 vertical strata, ranging from:
>
> > 1) 0-20 m
> >
> > 2) 21-40 m
> >
> > 3) 41-60 m
> >
> > 4) 61-80 m
> >
> > 5) 81-100 m
>
> Larvae found at depths deeper than 101 meters will not settle in any polygon.
>
> A larva located at 15m will settle only in a polygon located between 0-20m, while a larvae located at 41m will settle between 41-60m.
>
> These calculations will be done by the model, which will define at which vertical strata larvae and polygons are located based on the user defined strata file.

The polygon file will therefore have the following configuration, where the 4th column is the depth of the polygon in meters:

```
Longitude   Latitude    Polygon   Depth
277.2766    24.6665     1         20
277.2715    24.6235     1         20
277.2507    24.5921     1         20
277.2243    24.5786     1         20
277.1763    24.5777     1         20
277.1600    24.5564     1         20
277.1417    24.5851     1         20
277.1531    24.6684     1         20
277.2766    24.6665     1         20
```

Polygons with the same configuration (same vertices) located at different depths must be given unique polygon numbers. The connectivity output file using this option will record the spawning strata and the settlement strata.

Output temperature/salinity in netcdf file

```
outputtemp
outputsaln
```

If these flags are set to true, the temperature and/or salinity of the water that the particle passes through will be outputted to the trajectory output file (Section 7.1). These variables therefore need to be in the nest files.

6. How to run CMS

6.1 Prepare the nest.nml file

If you have used `getdata` to download/format the oceanographic input data, you will need to prepare a new *nest.nml* file in */input_name* for running cms. This tells cms what the variables are called, and which nest files to load in for the run. The nest.nml file will be almost the same as that used for `getdata`, although with the following changes:

- As you want cms to read the now locally stored files, remove the line with the url for download (**filename**) if used.
- Change the **tstart** and **tend** variables to match the time required by your release file (i.e. from the first release date to the last release date plus maximum advection time (set by the variable **timeMax** in *runconf.list*). This ensures that cms only loads in the nest files that it needs for the specific run, minimising memory usage. If you try to run cms for a longer time period than given in *nest_x.nml* you will get an error, even if the oceanographic data files exist for those dates.
- Replace the fill_value used for the download, which will have been converted by `getdata` to the value used by cms: 1.2676506E30

6.2 Running the program

After the input files are set up, you can run the model. For this go to the directory /expt and type:

```
./cms name_of_inputfile_directory
```

The name of the input file directory tells CMS which input files to use (i.e. the name of the experiment in *input_name*).

> For example:
>
> > ./cms example

In this case, CMS will use the input files stored in *input_example* to run the simulation, and the output files will be saved in the directory */expt_example/output/*.

It is also possible to run the model on multiple processors to reduce the running time. To achieve this, type :

```
mpirun -np number_of_processors ./cms name_of_inputfile_directory
```

> For example:
>
> > mpirun –np 8 ./cms example

In which case CMS will run on 8 processors.

CMS handles multiprocessing at startup by distributing the lines in the release file (see Section 5.1) over the number of processors. If the number of lines in the release file is less than the number of processors, the CMS will not run.

7. Output files

7.1 Trajectory output file

The trajectory output file can have two format options. The standard output is on NetCDF format. You can choose to set the format to ascii.

Option 1: NetCDF output

Dimensions:

```
Particle : number of particles in the NetCDF file
Time     : number of time steps saved in the NetCDF file
```

Variables:

```
int time(time)
    units     = "seconds"
    long_name = "Time after release"
    FillValue = -1

int location(particle)
    long_name = "Line number in the release file"
    FillValue = -1

float lon(particle,time)
    units     = "degrees_east"
    long_name = "Longitude"
    FillValue = 1.267651e+30f

float lat(particle,time)
    units     = "degrees_north"
    long_name = "Latitude"
    FillValue = 1.267651e+30f
```

```
float depth(particle,time)
    units     = "meter"
    long_name = "Depth"
    FillValue = 1.267651e+30f

float distance(particle,time)
    units     = "kilometer"
    long_name = "Cumulative distance"
    FillValue = 1.267651e+30f

int exitcode(particle)
    long_name = "Status with which the particle exits"

double releasedate(particle)
    units     = "Julian date"
    long_name = "Date the particle is released"
```

Only in output file if the flag **polygon** is turned on:

```
int releasepolygon(particle)
    long_name = "Number of release polygon"
```

Only in output file if the flag **buoyancy** is turned on:

```
float density(particle)
    units     = "kg/m3"
    long_name = "Density of particle"

float diameter(particle)
    units     = "meter"
    long_name = "Diameter of particle"
```

> Example:
>
> Two particles are released at different locations and days, and are dispersed for 10 days, as seen in Fig. 15. The output in the yellow box is for particle 1, while the red box is for particle 2.
>
> Particle 2 leaves the model area on day 7 (as denoted by the exit code -1) and does not move after this day, while particle 1 moves until the end of the simulation.

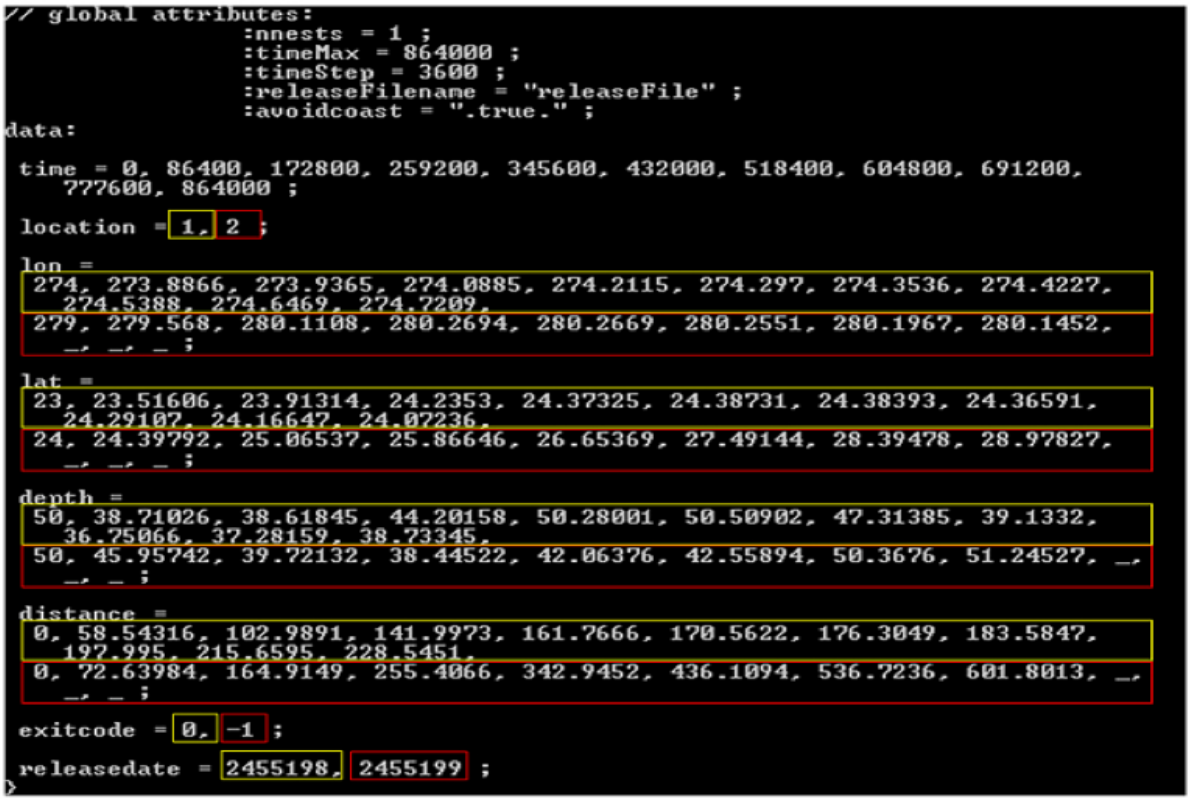


Figure 15. Example of an output file in NetCDF format.

Option 2: Ascii output

```
Location  Particle  Time    Longitude  Latitude  Depth  Distance  Exit_code  Release_date
1         1         0       282.380    24.480    0.000  0         0          2453431
1         2         0       277.034    4.615     1.000  0         0          2453431
2         1         0       278.000    25.000    100.0  0         0          2453432
1         1         43200   282.310    24.654    2.000  20.64     0          2453431
1         1         86400   282.321    24.833    4.000  40.44     -1         2453431

Polygon  Density  Diameter
5        820      0.005
5        820      0.005
1        840.5    0.0025
5        820      0.005
5        820      0.005


Location id
```

Line number in the release file.

```
Particle id
```

The number of the particle at a release location.

```
Time
```

The time in seconds from release.

```
Longitude, Latitude, Depth
```

Location of the particle in degrees, and depth in meters.

```
Distance
```

The total distance the particle travelled from its starting point (cumulative distance), in kilometres.

```
ExitCode
```

Status with which the particle exits

```
Releasedate
```

Date the particle is released in Julian date.

```
Release polygon
```

The number of the release polygon, only added if the flag polygon is turned on.

```
Density
```

The density of the particle in kg/m^3, only added if the flag buoyancy is turned on.

```
Diameter
```

The diameter of the particle in meters, only added if the flag buoyancy is turned on.

Distance calculation

The distance is calculated only using the longitudes and latitudes, so not the depth.

The formula used is:

```
dist = sin((lat2-lat1)/2)^2 + sin((lon2-lon1)/2)^2 * cos(lat1) * cos(lat2)

distance = R * 2 * atan2(sqrt(dist), sqrt(1-dist))
```

where R is the is earth's radius (mean radius = 6371.22 km)

This formula calculates the distance between position 1 (lon1, lat1) and position 2 (lon2, lat2).

Exit codes

Possible options of exit codes:

- **0** Particle was moving.
- **-1** Particle left the model area.
- **-2** Particle was too close to land.

- **-3** Particle died. This exit code appears only if using mortality.
- **-4** Particle settled in a polygon. This exit code appears only if using polygons.
- **-5** There were no oceanographic data files for this date.

7.2 Connectivity output file

The other output file is a file that contains the connectivity of the particles. This output file will only be created if the flag **polygon** in the file *ibm.list* is set to **.true.**.

```
Release_polygon  Settlement_polygon  Settlement_year  Settlement_month  Settlement_day  Age
1                4                   2008            8                 2               1209600
1                3                   2008            8                 3               1296000

Depth  Release_year  Release_month  Release_day
1.00   2008          7              20
1.00   2008          7              20
```

Each line of the connectivity file gives information about where and when a particle settled.

```
Release polygon
```

The number of the polygon where the particle was released

```
Settlement polygon
```

The number of the polygon where the particle settled

```
Settlement year, month, day
```

The year, month and day that the particle settled

```
Age
```

The age of the particle in seconds at the moment the particle settled

```
Depth
```

The depth in meters at the moment the particle settled

```
Release Year, month, day
```

The year, month and day that the particle was released

When using the flag **.strata.** two extra columns will be recorded in this file; spawning and settlement strata:

```
Spawning strata
```

The strata at the moment the particle was released

```
Settlement strata
```

The strata at the moment the particle settled

7.3 Post processing

The CMS package also provides Matlab® code for output visualization.

- **draw_velocities.m**: draws either the U-velocity, the V-velocity or the current speed (a combination of U and V velocity).
- **draw_traj_ascii.m**: draws the trajectories and land (using a nest file) when the outputfile is in Ascii format.
- **draw_traj_netcdf.m**: draws the trajectories and land when the outputfile is in netCDF format.
- **make_mtx.m**: computes and draws a square connectivity matrix.

All four codes have parameters at the start of the file, these parameters have to be filled in before you run the code.

8. Code

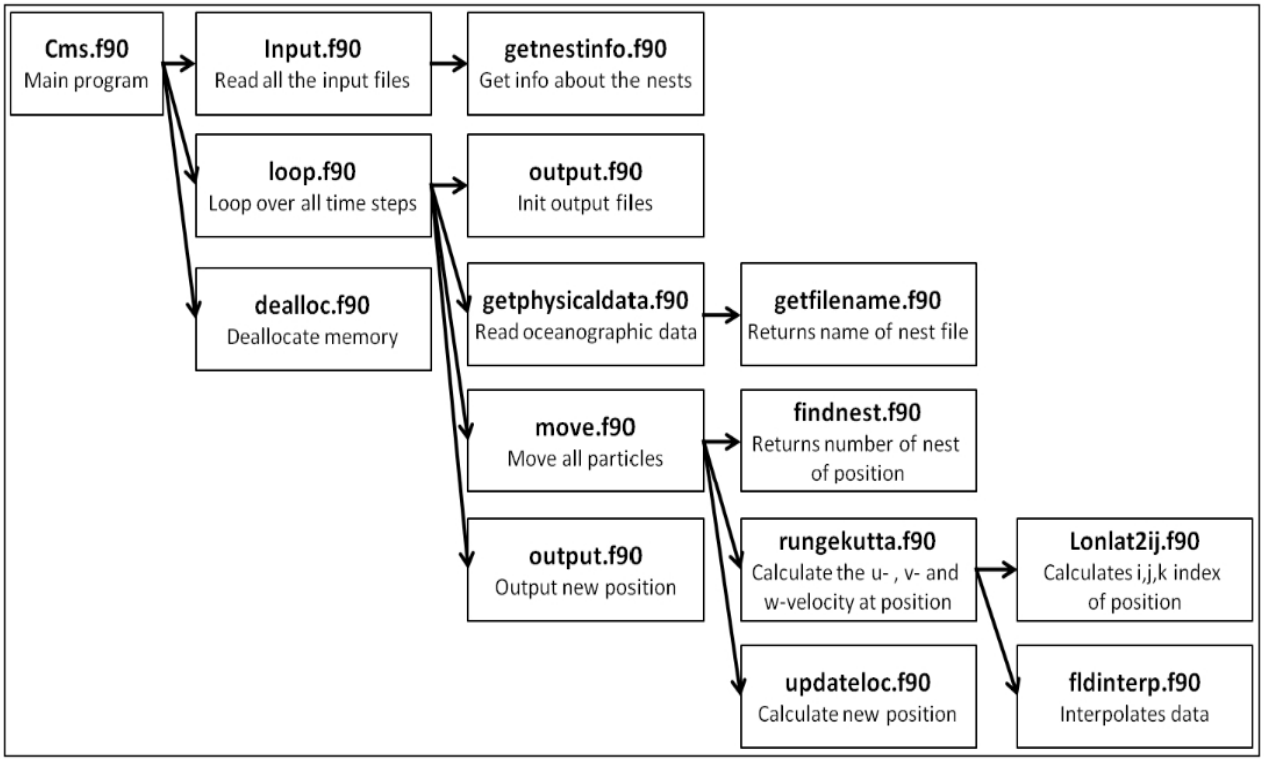


Figure 16. Main CMS subroutines.

Following is a complete listing of source code files with a short description of their content. The subroutines that define the most important variables are the following:

General

def_nests.f90 Defines types to store the information of the nests, such as boundaries, dates, and others

def_particle.f90 Defines types to store particles information, such as release position and date.

def_globalvariables.f90 Defines global variables

def_constants.f90 Defines constants

Modules

mod_kinds.f90 Specifies options for kind types

mod_netcdf.f90 Defines error messages related to NetCDF

mod_iounits.f90 Routines for automatic allocation of i/o units

mod_random.f90 Gaussian random number generator

mod_netcdfoutput.f90 Routines that create and write output to the NetCDF trajectory file

mod_directory.c Creates directories and copies files from one directory to another

mod_calendar.f90 Compute calendar date from Julian date and Julian date from calendar date

mod_nciorw.f90 Routines that create, read and write output to NetCDF files

mod_getdata.f90 Routines that read oceanographic files from an OPeNDAP server or from locally stored files

Modules used according to flags

mod_ibio.f90 Routines for vertical migration (**ibio** and **massSpawning**)

mod_turb.f90 Routines for turbulence movement

mod_mort_larva.f90 Routines for mortality

mod_buoyancy_larva.f90 Routines for buoyancy

mod_reef.f90 Routines for using polygons

mod_diffpart.f90 Routines if using diffpart

mod_mixedlayerphysics.f90 Routines for using mixed layer physics (flag **mixedlayerphysics**)

mod_strata.f90 Routines for using 3-dimensional polygons (flag **strata**)

9. CMS work flow: User check list

9.1 Source and prepare oceanographic data

The options are to:

a) Use `getdata` with online servers to download oceanographic data, or

b) Use your own files with `getdata` to utilise its regridding and formatting routines for optimal use with CMS (files should be in folder */expt/expt_name/nests/raw*)

To run `getdata` type:

```
./getdata name nest_no
```

Where *name* is the experiment name in *expt_name/input_name*, and *nest_no* is the nest number aligned with the *nest_x.nml* file you are using.

c) Use your own files as they are (files should be in folder */expt/expt_name/nests*)

All steps require at least one nest_x.nml file (in */expt/input_name/*) correctly parameterised for the raw data files wherever they come from, and nestfiles named in the format *nest#yyyymmddhhmmss.nc*. See section 4 for more information.

9.2 Prepare input files for CMS run

a) Check you have oceanographic files in */expt/expt_name/nests* (if you used `getdata` on files in the */expt/expt_name/nests/raw* folder the new formatted files will be in */expt/expt_name/nests*). You can use a symbolic link to a nests folder if you store your files somewhere else.

b) Ensure you have a correct *nest_x.nml* file (in */expt/input_name/*) for each set of nests (there must be at least one) relating to the oceanographic files that are in */expt/expt_name/nests*. N.B. if you obtained or formatted files using `getdata`, ensure that the fill value is now changed to the CMS fill value (1.2676506E30), and the tstart and tend dates reflect only the date range you wish to simulate over.

c) Set up your *runconf.list* and check all variables are correct (e.g. timestep, releaseFilename, backwards tracking, output file format, looping, etc).

d) Set up your *releaseFile* with the list of release locations, depths, numbers of particles and dates. Preferably a tab delimited text file with the following format:

```
Release Event or Polygon number, lon, lat, depth(+ve), number of particles, year, month, day, time (seconds)
```

e) [optional] Set up your *ibm.list* if using aspects such as buoyancy, seascape (polygons), strata, mortality, vertical migration, etc. Note that if you are using any one of the following modules, additional files are necessary:

- Buoyancy

- Seascape Polygons
- Strata for Polygons
- Vertical Matrix
- Different Particles Matrix

9.3 Run CMS

Change directory to */expt* and run cms by typing:

```
./cms name
```

Where *name* is the experiment name in *expt_name/input_name*.

To run the model on multiple processors, type :

```
mpirun -np number_of_processors ./cms name
```

> For example:
>
> > mpirun –np 8 ./cms example

Should you come across an error or warning, first check section 10 for guidance.

For further help and discussion, feel free to join the CMS users forum (groups.google.com/forum/m/#!forum/connectivity-modeling-system-club).

10. Troubleshooting

Here follows a non-exhaustive list of common errors and warnings, along with some suggested solutions. This list has been accumulated from posts on the CMS forum (groups.google.com/forum/m/#!forum/connectivity-modeling-system-club). Please feel free to join the forum yourself to ask for more specific advice and participate in discussions related to CMS – there is also plenty of installation advice if you are still stuck there. If there is anything you would like more clearly explained in the user guide please report this under the Github repository 'issues' tab.

If you diagnose any bugs in the code please report them at the Github site (github.com/beatrixparis/connectivity-modeling-system) or better yet, become a code contributor and fix them yourself.

If you are using an old version of CMS it is always a good idea to download and compile the latest version from the github to ensure that you have the benefit of any recent bug fixes.

```
"fortran runtime error end of file"
```

This error usually relates to the releaseFile and results from an error in the format of the contents. Check whether all columns are present in the correct order, and ensure there is no header data, or excess carriage returns adding blank lines to the end of the file. On this note, there must be a carriage return at the end of your input lines (i.e. 1 blank line at the end) for Fortran to read the last line in the file. Check this if you are getting e.g. 1 less release than the number of lines in your release file.

```
Fast run, all particles start with exit code -1
```

Usually this is a result of lon and lat columns in the release file being in the wrong order. Double check and straighten up! (if it's a mismatch between the dates and the oceanographic data the exit code will be -5).

```
Exit codes of -2
```

These particles are stuck on land. If this coincides with a time of 0 seconds then the launch position is not in the ocean according to the bathymetry of your oceanographic model (the coarse resolution of the bathymetry in broadscale oceanographic models is especially likely to cause this). If so, you must move the release location to a sensible spot according to your oceanographic model bathymetry/land mask.

```
"Error in the netCDF file:… no such file or directory"
```

If this names a file this is the file name it is expecting. Check that your equivalent file is a) present and b) named correctly.

```
"At line 97 of file input.f90 ….Fortran runtime error: End of file"
```

This is an example of an error which can happen with different line numbers and references to different source code files. If this happens go to the code file mentioned in the */src* directory (here i**nput.f90**) open it in a text editor and scroll to the appropriate line number. The example error shown here calls the *runconf.list*, so there may be a missing or additional carriage return or unexpected punctuation mark in that file.

```
"Error: probabilities in column 11 of vertical matrix exceed the 100%"
```

This is a mismatch between the **timeMax** (duration of the simulation), and the times allocated in the vertical matrix. Make sure your vertical matrix instructions end within the **timeMax**.

```
"Warning: polygon id   1   in the release file is not correct"
```

This is a mismatch between the polygon numbers in the polygon file and the polygon numbers in the release file, resulting in release locations outside of the polygon it is supposed to release from within.

Check that lat and lon columns are the right way round in both files, and that release locations are within the boundaries of your polygons. If you are intending to release particles from locations outside of a polygon then ensure that their number in the release file (column 1) doesn't match any of those assigned to a polygon, and ensure that the warning refers only to these polygon numbers (e.g. you have four polygons numbered 1 to 4, any release positions within these will also have a number between 1 and 4 representing the polygon they are being released from. If you want to release from outside of a polygon assign numbers 5 and up and ensure any warnings refer to polygon 5 or more).

N.B. This is only a warning, the CMS will still run these particles, but the warning is helpful if this is not what you were expecting.

```
The trajectories of particles seem to pass through land/topography at speed
```

You haven’t adjusted your fill value to the correct CMS fill value. This can be because you used `getdata` to download files using the **fillValue** on the server that provides the files, but when `getdata` downloads the files it changes the fill value to the CMS value (1.2676506E30). As a result, if you use the old *nest_x.nml* file with the old fill value (e.g. -30) when running `cms` then CMS takes the fill value as legitimate u and v velocities, resulting in particles being propelled through land at often very high speed.

```
"Error: You use more processors than the number of lines in the release file.  Decrease the number of processors"
```

This error occurs if you are doing an MPI CMS run that splits up the workload between the number of processors you designated. If you have designated e.g. 10 processors, and your release file has fewer than 10 lines of releases, then this error occurs. The number of processors must be less than or equal to the number of lines in the release file.

```
"NetCDF: Invalid dimension ID or name"
```

There is a mismatch between nest_x.nml dimension names, and those actually found in the NetCDF files. Check spellings and capitalisations are exactly matching.

```
"NetCDF: Index exceeds dimension bound"
```

There is an error in your nest_x.nml file that results in `getdata` or `cms` interpreting the wrong number or length of dimensions in the netcdf file. Again check that all dimension names (e.g. latitude, longitude, time, depth) and start and end values correspond with those in the netCDF files you are using as the oceanographic data.