Tutorial 09 LiveCROCO: Simulation with Particles

1. Purpose

In this tutorial we will review how to perform a simulation of Benguela domain including simultaneous release on line of passive particles, the important files for this type of simulation, and its validation.

2. Creating working directory PARTICULAS

The simplest example of CROCO is the configuration called BENGUELA_LR which corresponds to a low resolution Benguela Upwelling Zone domain. This is the default configuration in CROCO code and what we will do is similar to what is described in Penven et al. (2001)

We will assume that you already have a local copy of the CROCO code and CROCO_TOOLS in your root directory. This is achieved by following the steps in Tutorial 01.

The first step is to edit the file **create_run.bash** with the instructions to create a new working directory that we will name **PARTICULAS**

```
cd croco
nano create_run.bash
```

Now you have to modify this section to put the correct directories

```
#
# Name of the configuration directory defined by the user
#
MY_CONFIG_NAME='PARTICULAS'
#
#
# END USER MODIFICATIONS
#==========================================================================================

```

and then execute the statement

```
./create_run.bash
```

3. Checking physics - cppdefs.h

The **cppdefs.h** file is one of the most important when performing a simulation with CROCO. It is very convenient to understand the structure of this file. You will have a copy of **cppdefs.h** file in your working directory, and this is the one you need to modify. Some aspects that we will highlight now are:

3.1. REGIONAL configuration

In this section it is defined that we will use the REGIONAL type configuration, which is suitable for realistic simulations, associated with a particular geographical area.

In **cppdefs.h** file corresponds to this part of the code:

```
#if defined REGIONAL
/*
!===================================================================
```

```
!               REGIONAL (realistic) Configurations
!=====================================================================
!
!----------------------
! BASIC OPTIONS
!----------------------
!
*/
                        /* Configuration Name */
# define BENGUELA_LR
```

Within the REGIONAL configuration, the first thing we define is a keyword to identify our configuration. In this case, that keyword is **BENGUELA_LR**. This keyword is used to connect **cppdefs.h** and **param.h** file, as we will see later.

3.2. Parallelization

Enable parallelization for your numerical simulation

```
                        /* Parallelization */
# undef  OPENMP
# define  MPI
```

3.3. Particles - cppdefs.h

In this section you activate the FLOATS application or module by defining it as *define FLOATS*. The advection of particles will be done simultaneously with hydrodynamics calculation of our domain.

```
                        /* Applications */
# undef  BIOLOGY
# define  FLOATS
# undef  STATIONS
# undef  PASSIVE_TRACER
# undef  SEDIMENT
# undef  BBL
```

3.4. Lagrangian model

Another relevant section is the Lagrangian model. By default, the configuration is defined as current advection only. The advection-diffusion scheme can be activated by defining: *define RANDOM_WALK*, but for now we will leave them undefined (*undef*), that is, deactivated.

```
                        /*   Lagrangian floats model    */
# ifdef FLOATS
#  undef  FLOATS_GLOBAL_ATTRIBUTES
#  undef  IBM
#  undef  RANDOM_WALK
#  ifdef RANDOM_WALK
#   define DIEL_MIGRATION
#   define RANDOM_VERTICAL
#   define RANDOM_HORIZONTAL
#  endif
# endif
```

4. Compiling CROCO

Once we check that **cppdefs.h** has the physical options we want, and that **param.h** file defines the dimensions of our domain correctly, we compile the CROCO executable using the following instructions.

```
cd PARTICULAS
./jobcomp
```

5. Input files

The input files to be read by **croco** executable will be created with **CROCO_TOOLS** tool. It mainly depends on two files, **oct_start.m** and **crocotools_param.m**

5.1. PARTICLES - using Octave

To create input files using Octave, we first have to load the program using

```
octave-cli
```

The instructions to use in Octave, from the **PARTICULAS** working directory are:

```
oct_start
make_grid
make_forcing
make_clim
```

It is RECOMMENDED to use the initial condition generated from third or fourth year of climatological run, which implies that circulation in the domain is already developed (it does not start at rest). You can download the file of initial condition that corresponds to the fourth year of climatological simulation. The file must be placed in /PARTICULAS/CROCO_FILES/ directory.

```
wget http://mosa.dgeo.udec.cl/CROCO2021/Tutorial09/Particulas/croco_ini.nc
```

5.2. Generating the particle file - floats.in

In this section we define the positions of particles x (x,y,z,t) to release using an Octave subroutine (*i.e. initfloats.m*) that is downloaded to the working directory (PARTICULAS) by the following instruction,

```
wget http://mosa.dgeo.udec.cl/CROCO2021/Tutorial09/initfloats.m
```

Reload the Octave program using,

```
octave-cli
```

Run the *initfloats.m* program and verify that *floats.in* output file has been created in the working directory (PARTICULAS). Information on the structure of the floats.in file can be found in the *floats.in_README* file that is downloaded with the command,

```
wget http://mosa.dgeo.udec.cl/CROCO2021/Tutorial09/floats.in_README
```

General structure of the *floats.in* file.

```
1  Ftitle (a80)
ROMS 1.0 - Initial Drifters Locations
2  Ft0,   Fx0,   Fy0,     Fz0, Fgrd,Fcoor,Ftype,Fcount, Fdt,   Fdx,   Fdy,   Fdz
  0.0   15.0000 -26.7918   -10.0   0   1    0    1       0.00   0.000   0.000   0.0
```

Some parameters are described below,

```
Ftitle='ROMS 1.0 - Initial Drifters Locations';
Fgrd  =0;     Fgrd      Nested grid number (ng)
Fcoor =1;     Fcoor     Initial horizontal location coordinate type:
              Fcoor = 0,  grid units:  1 =< Fx0 =< Lm(ng)+1,
                                       1 =< Fy0 =< Mm(ng)+1

              Fcoor = 1,  Fx0 is longitude (west values are negative).
                          Fy0 is latitude (south values are negative).
Ftype =0;     Float trajectory type:
              Ftype = 0,  neutral density 3D Lagrangian particles.

              Ftype = 1,  isobaric (constant depth) floats.
Fcount=1;     Number of floats to be released at the (Fx0,Fy0,Fz0)
              location.  It must be equal or greater than one.
              If Fcount > 1, a cluster distribution of floats
              centered at (Fx0,Fy0,Fz0) is activated.

              NOTE:  The total number of floats trajectories to compute
              ====   must add to NFLOATS.
```

6. The options file - croco.in

In the croco.in file, the recording period of particle positions is defined, by default it is defined every 6 h. Input and output filenames are also defined,

```
floats: LDEFFLT, NFLT, NRPFFLT / inpname, hisname
           T      6      0
                                  floats.in
    CROCO_FILES/floats.nc
```

Once this is done, we launch the simulation

```
./croco croco.in
```

Once the simulation finishes successfully, we will find in **CROCO_FILES** directory the following output files

```
croco_avg.nc
croco_his.nc
croco_rst.nc
floats.nc
```

7. Output file-floats.nc

Below is the structure of some variables in the output file (*/CROCO_FILES/floats.in*). It is important to note that trajectory of each particle is stored in each column of matrix (lon, lat, temp, salt).

```
double lon(ftime, drifter) ;
                lon:long_name = "longitude of floats trajectories" ;
                lon:units = "degree_east" ;
                lon:field = "lon, scalar, series" ;
double lat(ftime, drifter) ;
                lat:long_name = "latitude of floats trajectories" ;
                lat:units = "degree_north" ;
double temp(ftime, drifter) ;
                temp:long_name = "temperature" ;
                temp:units = "degrees Celsius" ;
double salt(ftime, drifter) ;
                salt:long_name = "salinity" ;
                salt:units = "PSU" ;

```

To visualize the trajectories of particles, you can adapt following program,

```
wget http://mosa.dgeo.udec.cl/CROCO2021/Tutorial09/graph_particles.m
```

8. Expected results

You must be able to generate

1. Plots of particle trajectories, as well as temperature and salinity properties associated with their trajectory.
2. Do a run with 100 particles released at a given position (point source)
3. Activate the RANDOM_WALK module and compare the trajectories of the point source (100 particles)

9. Advanced Work

Run a particle simulation for a domain of interest.

10. Conclusion

In this tutorial you learned more about **cppdefs.h** files and modifications that need to be done to perform a simulation with Lagrangian particles.

Developing:
Mauro Santiago
For more information:
Andrés Sepúlveda (asepulveda@dgeo.udec.cl)
Collaborations:
Marcela Contreras
Sebastian Inzunza

If you found this tutorial useful, please send a postcard to:

Dr. Andrés Sepúlveda
Departamento de Geofísica
Casilla 160-C
Correo 3
Concepción
Chile

11. Helpful Links

- Recommended articles

Marinone, S. G., Lavín, M. F., & Parés-Sierra, A. (2011). A quantitative characterization of the seasonal Lagrangian circulation of the Gulf of California from a three-dimensional numerical model. Continental Shelf Research, 31(14), 1420{1426. doi:10.1016/j.csr.2011.05.014.

Maslo, A., Azevedo Correia de Souza, J. M., Andrade-Canto, F., & Rodríguez Outerelo, J. (2019). Connectivity of deep waters in the Gulf of Mexico. Journal of Marine Systems, 103267. doi:10.1016/j.jmarsys.2019.103267.

Rivas, D., & Samelson, R. M. (2011). A Numerical Modeling Study of the Upwelling Source Waters along the Oregon Coast during 2005, Journal of Physical Oceanography, 41(1), 88-112.

Mantovanelli, A., Heron, M. L., Prytz, A., Steinberg, C. R., & Wisdom, D. (2011). Validation of radar-based Lagrangian trajectories against surface-drogued drifters in the Coral sea, Australia. OCEANS'11 MTS/IEEE KONA. doi:10.23919/oceans.2011.6107233.