MOHID – LAGRANGIAN - Short Guide

PROLOGUE

This manual is a short guide in order to run MOHID-Lagrangian code and start producing outputs and explore different setups and options. Many of you have asked Ricardo and me about how to use the software so before a long technical manual I decided to write this short manual for CleanAtlantic members. I wrote it in two days so I am sure that it will contain many English mistakes. I will correct it and improve it

INDEX

1. INTRODUCTION:

The recommended way to use MOHID – Lagrangian is following the scheme:

1. **Base template**
2. **Copy it**
3. **Personalize it**

That is, you should copy a directory with a template or another working simulation test, then modify the setup files inside to fit in your needs.

Let's go step by step how to prepare a working simulation:

1.1 Test working simulations or template folder

Inside the path /MOHID-Lagrangian/RUN_Cases, we provide different working examples:

- Arousa_2D_test_case
- PCOMS_test_case
- Tagus3D_case
- Vigo_3D_test_case
- Case_template (does not run)

The four first cases are working setups of MOHID-Lagrangian with the files configured to work properly and test how MOHID-Lagrangian works. If everything is fine and there are not problems at the installation stage each case should run fine just by running the scripts inside each test_case folder:

RunCase.sh (Linux)

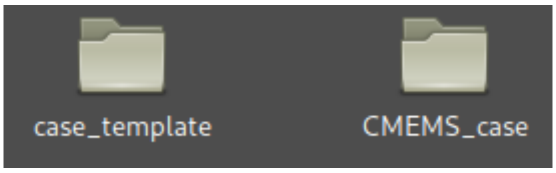
RunCase.bat (windows)

The Case_template, does not run a specified case, however it contains all possible options on the configuration files to setup the simulation at your needs. From here, we will use this folder as a template to create a new simulation.

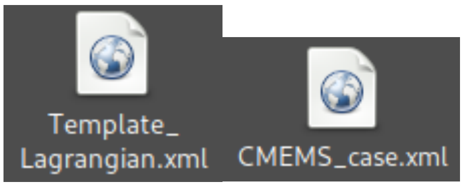
1.2 Create a new simulation from Case_template

Suppose we want to perform a new simulation using CMEMS surface current data for example. This netcdf hydrodynamic data contains the currents speeds u,v,w. The first step is to:

1) copy the **case_template** folder into a new one and rename to **CMEMS_case** for example:

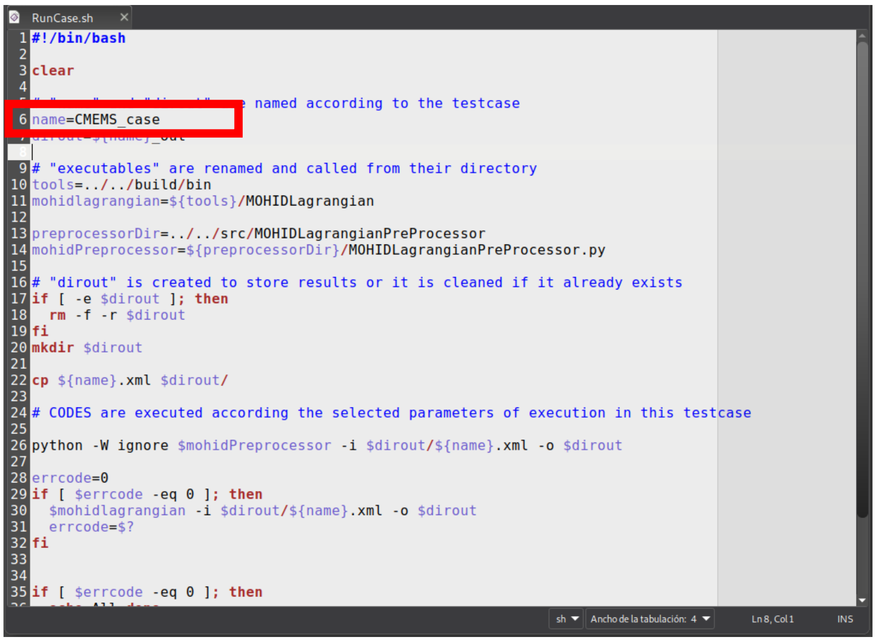


2) Then, to keep an naming agreement clear (optional but recommended) **rename the file Template_Lagrangian.xml** to **CMEMS_case.xml**



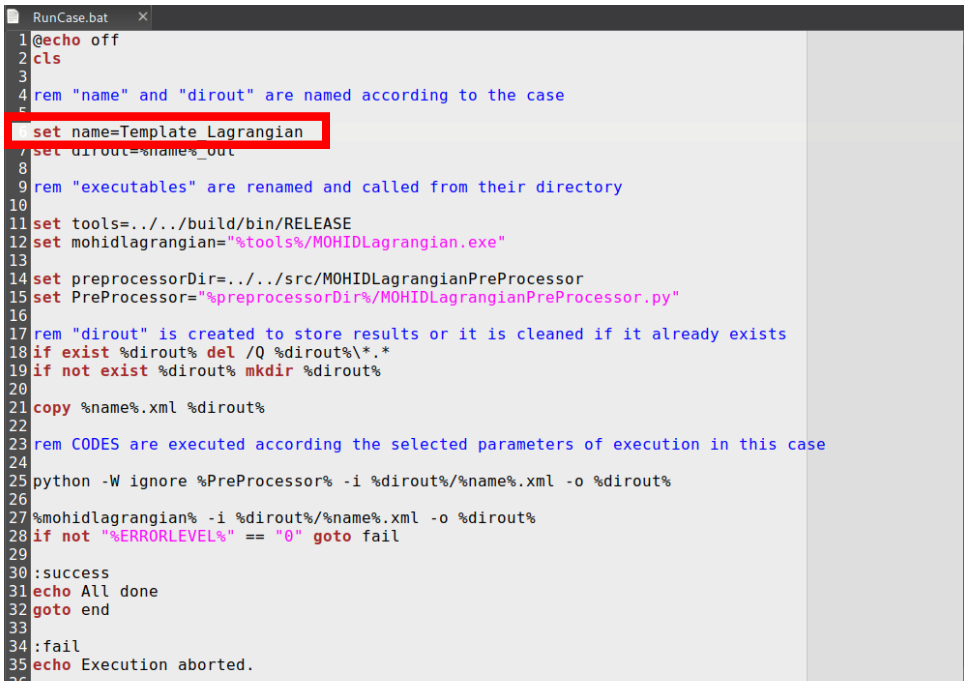
3) Open with your text editor the file **RunCase.sh (in Linux)** or **RunCase.bat (in windows)** and edit the line number 6) with:

- LINUX – (RunCase.sh) name=Template_Lagrangian -----> name=CMEMS_case



```
 1 #!/bin/bash
 2 
 3 clear
 4 
 5 # "name" and "dirout" are named according to the testcase
 6 name=CMEMS_case
 7 dirout=${name}_out
 8 
 9 # "executables" are renamed and called from their directory
10 tools=../../build/bin
11 mohidlagrangian=${tools}/MOHIDLagrangian
12 
13 preprocessorDir=../../src/MOHIDLagrangianPreProcessor
14 mohidPreprocessor=${preprocessorDir}/MOHIDLagrangianPreProcessor.py
15 
16 # "dirout" is created to store results or it is cleaned if it already exists
17 if [ -e $dirout ]; then
18   rm -f -r $dirout
19 fi
20 mkdir $dirout
21 
22 cp ${name}.xml $dirout/
23 
24 # CODES are executed according the selected parameters of execution in this testcase
25 
26 python -W ignore $mohidPreprocessor -i $dirout/${name}.xml -o $dirout
27 
28 errcode=0
29 if [ $errcode -eq 0 ]; then
30   $mohidlagrangian -i $dirout/${name}.xml -o $dirout
31   errcode=$?
32 fi
33 
34 
35 if [ $errcode -eq 0 ]; then
```

- WINDOWS – (RunCase.bat) set name= Template_Lagrangian -----> set name = CMEMS_case



RunCase.bat

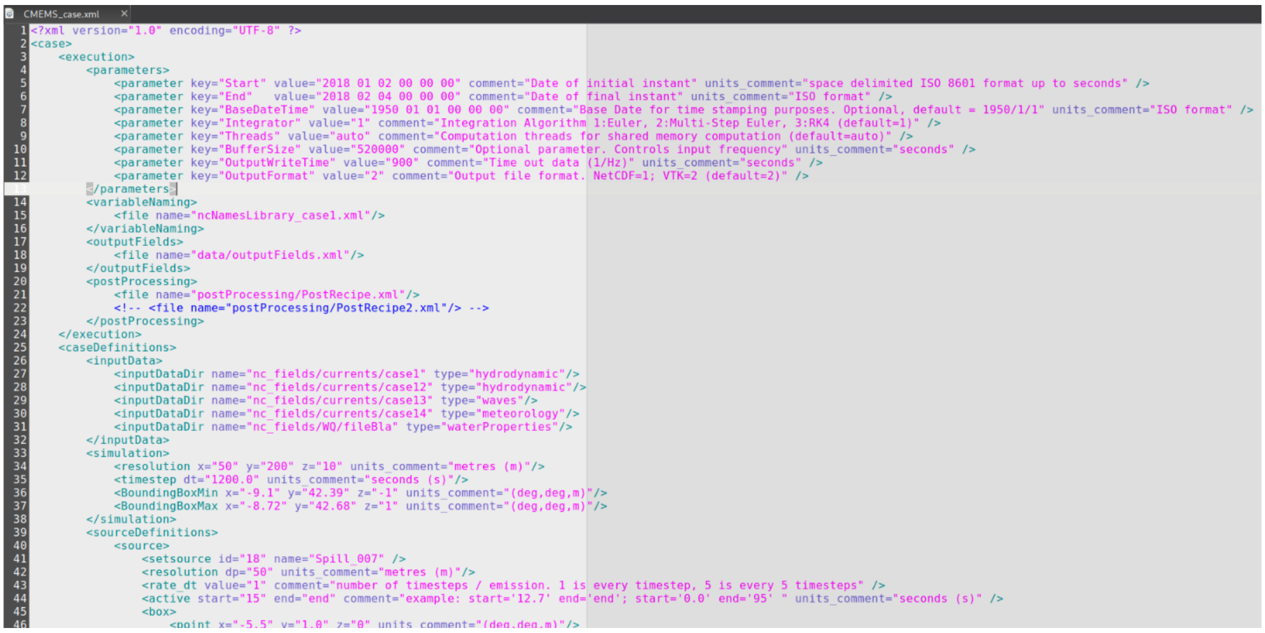
```
@echo off
cls

rem "name" and "dirout" are named according to the case

set name=Template_Lagrangian
set dirout=%name%_out

rem "executables" are renamed and called from their directory

set tools=../../build/bin/RELEASE
set mohidlagrangian="%tools%/MOHIDLagrangian.exe"

set preprocessorDir=../../src/MOHIDLagrangianPreProcessor
set PreProcessor="%preprocessorDir%/MOHIDLagrangianPreProcessor.py"

rem "dirout" is created to store results or it is cleaned if it already exists
if exist %dirout% del /Q %dirout%\*.*
if not exist %dirout% mkdir %dirout%

copy %name%.xml %dirout%

rem CODES are executed according the selected parameters of execution in this case

python -W ignore %PreProcessor% -i %dirout%/%name%.xml -o %dirout%

%mohidlagrangian% -i %dirout%/%name%.xml -o %dirout%
if not "%ERRORLEVEL%" == "0" goto fail

:success
echo All done
goto end

:fail
echo Execution aborted.
```

2. Setup the Main_Case.xml setup file.

Once we rename the executable files, then we start setup the main setup file of our simulation:

CMEMS_case.xml

When you open it you can observe the following text in xml format:



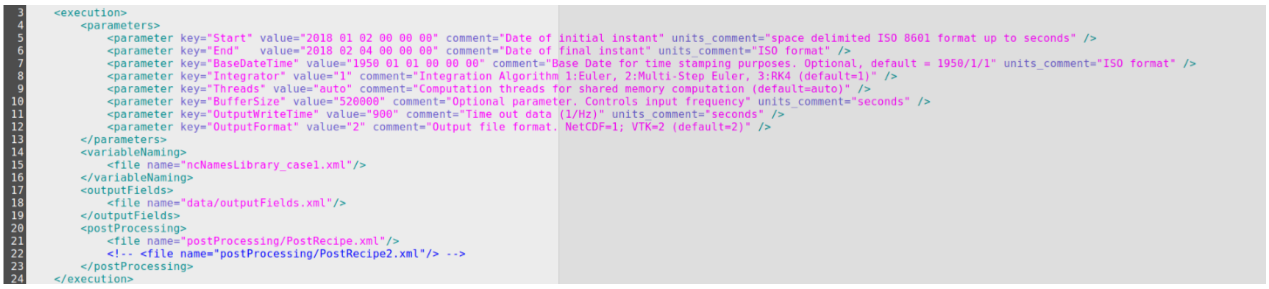
CMEMS_case.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<case>
    <execution>
        <parameters>
            <parameter key="Start" value="2018 01 02 00 00 00" comment="Date of initial instant" units_comment="space delimited ISO 8601 format up to seconds" />
            <parameter key="End"   value="2018 02 04 00 00 00" comment="Date of final instant" units_comment="ISO format" />
            <parameter key="BaseDateTime" value="1950 01 01 00 00 00" comment="Base Date for time stamping purposes. Optional, default = 1950/1/1" units_comment="ISO format" />
            <parameter key="Integrator" value="1" comment="Integration Algorithm 1:Euler, 2:Multi-Step Euler, 3:RK4 (default=1)" />
            <parameter key="Threads" value="auto" comment="Computation threads for shared memory computation (default=auto)" />
            <parameter key="BufferSize" value="520000" comment="Optional parameter. Controls input frequency" units_comment="seconds" />
            <parameter key="OutputWriteTime" value="900" comment="Time out data (1/Hz)" units_comment="seconds" />
            <parameter key="OutputFormat" value="2" comment="Output file format. NetCDF=1; VTK=2 (default=2)" />
        </parameters>
        <variableNaming>
            <file name="ncNamesLibrary_case1.xml"/>
        </variableNaming>
        <outputFields>
            <file name="data/outputFields.xml"/>
        </outputFields>
        <postProcessing>
            <file name="postProcessing/PostRecipe.xml"/>
            <!-- <file name="postProcessing/PostRecipe2.xml"/> -->
        </postProcessing>
    </execution>
    <caseDefinitions>
        <inputData>
            <inputDataDir name="nc_fields/currents/case1" type="hydrodynamic"/>
            <inputDataDir name="nc_fields/currents/case12" type="hydrodynamic"/>
            <inputDataDir name="nc_fields/currents/case13" type="waves"/>
            <inputDataDir name="nc_fields/currents/case14" type="meteorology"/>
            <inputDataDir name="nc_fields/WQ/fileBla" type="waterProperties"/>
        </inputData>
        <simulation>
            <resolution x="50" y="200" z="10" units_comment="metres (m)"/>
            <timestep dt="1200.0" units_comment="seconds (s)"/>
            <BoundingBoxMin x="-9.1" y="42.39" z="-1" units_comment="(deg,deg,m)"/>
            <BoundingBoxMax x="-8.72" y="42.68" z="1" units_comment="(deg,deg,m)"/>
        </simulation>
        <sourceDefinitions>
            <source>
                <setsource id="18" name="Spill_007" />
                <resolution dp="50" units_comment="metres (m)"/>
                <rate_dt value="1" comment="number of timesteps / emission. 1 is every timestep, 5 is every 5 timesteps" />
                <active start="15" end="end" comment="example: start='12.7' end='end'; start='0.0' end='95' " units_comment="seconds (s)" />
                <box>
                    <point x="-5.5" y="1.0" z="0" units_comment="(deg,deg,m)"/>
```

For those who have no idea about the xml format. The xml file has a tree structure with blocks or sections, with subblocks or subsections. Each section begins with <nameSection> and ends with </nameSection>. If it is an inline section (a section just in one line it is <namesection -------/>. Inside each section, there is always a pair or many pairs of "nameofsomething"="valueofsomething".

2.1 Setup the <Execution> block

We will start from top to setup each condition or parameters. The first main block you can observe is the <execution> block.



```
<execution>
    <parameters>
        <parameter key="Start" value="2018 01 02 00 00 00" comment="Date of initial instant" units_comment="space delimited ISO 8601 format up to seconds" />
        <parameter key="End"   value="2018 02 04 00 00 00" comment="Date of final instant" units_comment="ISO format" />
        <parameter key="BaseDateTime" value="1950 01 01 00 00 00" comment="Base Date for time stamping purposes. Optional, default = 1950/1/1" units_comment="ISO format" />
        <parameter key="Integrator" value="1" comment="Integration Algorithm 1:Euler, 2:Multi-Step Euler, 3:RK4 (default=1)" />
        <parameter key="Threads" value="auto" comment="Computation threads for shared memory computation (default=auto)" />
        <parameter key="BufferSize" value="520000" comment="Optional parameter. Controls input frequency" units_comment="seconds" />
        <parameter key="OutputWriteTime" value="900" comment="Time out data (1/Hz)" units_comment="seconds" />
        <parameter key="OutputFormat" value="2" comment="Output file format. NetCDF=1; VTK=2 (default=2)" />
    </parameters>
    <variableNaming>
        <file name="ncNamesLibrary_case1.xml"/>
    </variableNaming>
    <outputFields>
        <file name="data/outputFields.xml"/>
    </outputFields>
    <postProcessing>
        <file name="postProcessing/PostRecipe.xml"/>
        <!-- <file name="postProcessing/PostRecipe2.xml"/> -->
    </postProcessing>
</execution>
```

This block controls the execution parameters. In the *comment* field, there is a description about each entry. In the ***<parameters>*** section, there are sub inline section called **<parameter>** with the template:

> *<parameter key=" the name of the parameter" value="its value" comment="help comments-optional" units_comment="units if they exist - optional" />*

Using this structure, we start to setup the different <parameters> in this section

2.1.1 Change the Start time and end time

These two parameters control the date when the simulation starts and finishes.

> *<parameter key="Start" value="2018 01 02 00 00 00" comment="Date of initial instant" units_comment="space delimited ISO 8601 format up to seconds" />*
>
> *<parameter key="End"   value="2018 02 04 00 00 00" comment="Date of final instant" units_comment="ISO format" />*

Be sure that the time limits when your simulation start and/or ends fits in with your time range of netCDF input data (later, we will check this). An example: to perform a two-year simulation from 2017 to 2019 we have to change the time period from the previous one to:

> *<parameter key="Start" value="2017 01 01 00 00 00" comment="Date of initial instant" units_comment="space delimited ISO 8601 format up to seconds" />*
>
> *<parameter key="End"   value="2019 01 01 00 00 00" comment="Date of final instant" units_comment="ISO format" />*

2.1.2 Change the Integrator.

The next block controls the type of integrator to use:

> *<parameter key="Integrator" value="2" comment="Integration Algorithm 1:Euler, 2:Multi-Step Euler, 3:RK4 (default=1)" />*

In the comment, you can observe the different options:

1. Euler (faster, but less precise)
2. Muli-Step Euler (slower than Euler but more precise)
3. RK4 (slower than Muli-Step Euler but more precise)

Apart from purely academic or conceptual cases, option “2” is enough, especially if one considers the effects of diffusion.

2.1.3 Change the number of Threads

The next <parameter> controls the number of threads for shared memory computation (at this moment, MOHID-Lagrangian can only run in one machine).

> *<parameter key="Threads" value="2" comment="Computation threads for shared memory computation (default=auto)" />*

To get a better performance this value should be proportional to the number of cores available in your system. The auto option uses all cores, user doesn’t need to worry. If your machine has 8 physical cores but you want to use only 2 processes change the value from ”auto” to “2”. Recommended range is between 4 to 6 cores in most CPUs, not necessarily maximum.

2.1.4 Change the output frequency.

The parameter key=”OutputWriteTime” controls when the data is written to disk, effectively the sampling frequency of your solution. For example, if your solution is computed every hour (dt=3600s), you could have an <OutputWriteTime value=”10800”. This makes that every three time steps (10800/3600 = 3) of computation, the third one is written to disk.

2.1.5 Change the buffersize

The parameter with *key=”BufferSize”* is important to perform long integrations. For large hydrodynamic fields, it allows you to control the amount of data to store in RAM memory (you cannot load 30Gb of hydrodynamic fields if your computer has 8gb of RAM) before going to disk to read a new chunk of data and continue the integration. A buffer size of value=”520000” means that MOHID-Lagrangian is going to read a chunk of data of “520000” seconds and copy it into memory, use that data to simulate the particle motion and then if it requires data to continue, it add it while the previous data is released from ram memory. This value should be changed just if you have some errors related to “not enough RAM memory” and make it lower.

2.1.6 Change the output format.

This value controls the output format files to write on disk. At this moment, MOHID-Lagrangian just supports *vtk* output file format so the *value* must be kept at ”2”. NetCDF and hdf5 files are produced in Post-processing (see chapter 0)

2.2 Setup the netCDF variable names

The variable < *variableNaming* > section just points to an xml file containing the naming conventions library. This file can be anywhere in your machine, the path can be given in absolute of relative ways.

<variableNaming>

<file name="ncNamesLibrary_case1.xml"/>

</variableNaming>

See chapter 3 for details on this file.

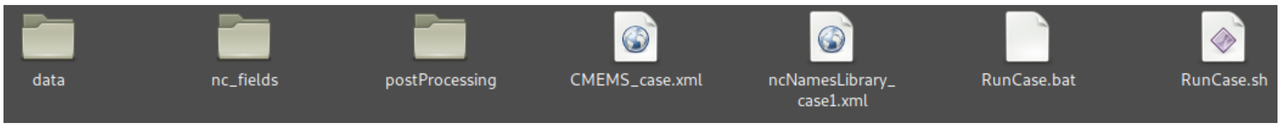
2.3 Setup the <caseDefinitions>

The caseDefinitions block controls the specific features for your scenario. In controls the path for input data, simulation domain limits (bounding boxes), timesteps and resolution and also the definition of sources together with other parameters to control some physical aspects.

2.3.1 netCDF input files

In order to move the particles in within the medium, netCDF files with velocity fields must be provided. To add them and to use it in our simulation, we propose to do in the following way:

1) Create a folder inside the CMEMS_case called for example: nc_fields



2) Inside the nc_fields, we create different subfolders, depending the type of the input that we are going to use. At this moment, we support 4 different input types.
   - **hydrodynamic**: velocities u,v,w, can also contain other variables
   - **waves**: read the vsdx, vsdy (velocity stokes drift x and y)
   - **meteorology**: read the wind surface or wind at 10 m u10,v10
   - **waterproperties**: read salinity and temperature: salt, temp

3) For each type we recommend creating a subfolder with this structure:

```
├── nc_fields
|   ├── currents – place the netCDFs fields with the currents velocity u,v and w .
|   ├── water_properties – place the netCDFs with temperature and salinity fields
|   ├── waves – place the netCDFs with stokes velocity drift
|   └── winds – place inside the netCDF files with wind speed at surface
```

4) **Inside each folder, place the netCDF data**. The name is not important.

The netCDF data inside each subfolder can be in one file, or in multiple netCDF files and they can be in further subdirectories. You do not have to worry about this. The MOHID-Lagrangian preprocessor analyzes the time dimension of the all netCDF files and provides a continuous stream for the integration between the different files in MOHID Lagrangian.

Once you placed the netCDFs files inside each folder, please, check that your variables and dimension names inside the files are names in the ncNamesLibrary_case1.xml, as described in chapter 3.

The input data section should look like:

```
    <inputData>
        <inputDataDir name="nc_fields/currents" type="hydrodynamic"/>
        <inputDataDir name="nc_fields/waves" type="waves"/>
        <inputDataDir name="nc_fields/winds" type="meteorology"/>
        <inputDataDir name="nc_fields/waterProperties" type="waterProperties"/>
    </inputData>
```

Notice that the names of the directories and files is arbitrary, only the 'type' keywords must be respected. You can also add several paths to the same 'type' of file:

```
<inputData>
        <inputDataDir name="nc_fields/currents" type="hydrodynamic"/>
        <inputDataDir name="D:myDrive/weid_data" type="hydrodynamic"/>
    </inputData>
```

In the simplest case, if we just want to perform a simulation just using the hydrodynamic currents to track the motion of water masses, we just have to set up as input data:

```
<inputData>
   <inputDataDir name="nc_fields/currents" type="hydrodynamic"/>
</inputData>
```

2.3.2 Setup the simulation section

The *<simulation>* controls some key parameters. The most important are the *timestep* and the *BoundingBox*

```
<simulation>
   <resolution x="50" y="200" z="10" units_comment="metres (m)"/>
   <timestep dt="1200.0" units_comment="seconds (s)"/>
   <BoundingBoxMin x="-9.1" y="42.39" z="-1" units_comment="(deg,deg,m)"/>
   <BoundingBoxMax x="-8.72" y="42.68" z="1" units_comment="(deg,deg,m)"/>
</simulation>
```

The *<timestep>,* controls when the solution is computed. It is your choice and depends on the spatial and time resolution of your data and the hydrodynamic type.

The *<BoundingBoxMin and Max>*, defines the corners of your simulation domain (simulation box) or in other words, defines the domain where your trajectories of the particles are computed. If a particle is outside this bounding box, it will be excluded from the simulation.

The BoundingBox, can be smaller than the field domain inside your netCDF files. If a particle reaches the BoundingBox limits despite there is data outside of it, it will be excluded from the domain.

The boundingBox can be bigger than your netCDF data, however, if the particles leave the data domain of your netcdf they will keep zero velocity and stay at the last computed position, because there is no data to integrate the particles. This can influence your results in post-processing.

The bounding box is not automatically set up. If you consider using the whole domain from your netCDF data, you must set it up correctly according to you netCDF data limits.

2.3.3 Setup the sources

Each source defined needs a block source. If you want to put two different sources, you require:

```
<source>
        Source definition 1
        Source rate
        Source geometry
</source>
<source>
        Source definition 2
        Source rate
        Source geometry
</source>
```

You can add as many sources as you want.

2.3.3.1 Setup the sources ID

All the sources have a common property:

```
<source>
        <setsource id="18" name="Spill_007" />
</source>
```

Each source requires an id and name to be identified. This is very important to specify the origin of the particles and type of the material/ particle that will be emitted. This must be set up later in the <sources type> definition block.

IMPORTANT: each source, can emit only one type of particles. If you want to emit two types of particles from one source, define two sources with two ids and two names to identify them, keeping their geometry the same, and add the corresponding material in the sources Types block later.

2.3.3.2 Setup the source rate

The rate of emission can be setup in two different ways, using a time series <rateTimeSeries> block or using a fixed rate. Using a time series approach, the user can specify:

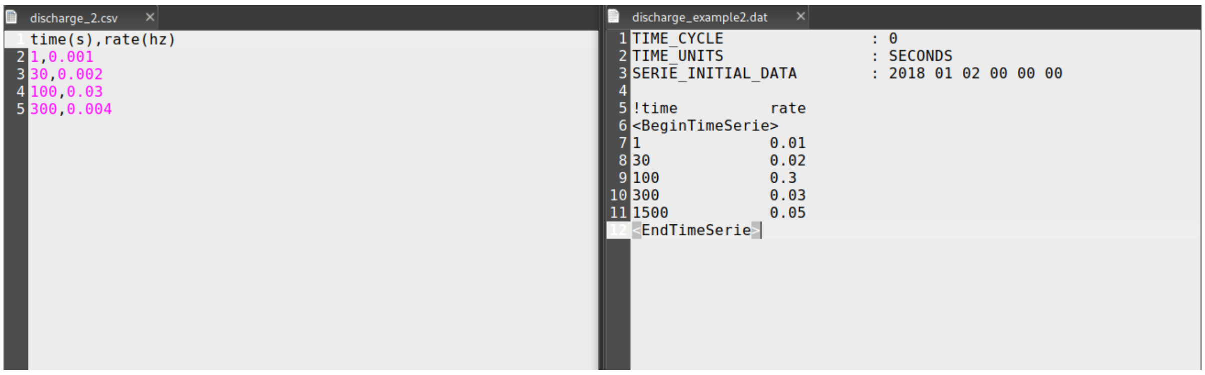
<rateTimeSeries>

<file name="data/discharge_example.csv" comment="name of csv file with discharge information (time and rate columns)"/>

<scale value="1.01" comment="scales the data on the file by this factor (not time)" />

</rateTimeSeries>

The time series files can be either be given in csv or dat (MOHID time series) files:


discharge_2.csv

```
time(s),rate(hz)
1,0.001
30,0.002
100,0.03
300,0.004
```

discharge_example2.dat

```
TIME_CYCLE                : 0
TIME_UNITS                : SECONDS
SERIE_INITIAL_DATA        : 2018 01 02 00 00 00

!time           rate
<BeginTimeSerie>
1               0.01
30              0.02
100             0.3
300             0.03
1500            0.05
<EndTimeSerie>
```

On the left, it is a csv file. In the left column, the time is in seconds from the beginning of the simulation specified in <execution><parameters><parameter key=”Start” value=”2017 01 01 00 00 00”/>

In the right hand, the initial time for emission is at the SERIE_INITIAL_DATA attribute.

To define a fixed emission rate there are two options:

<rate value="0.0333" comment="emission rate (Hz)" />

Or

<rate_dt value="1" comment="number of timesteps / emission. 1 is every timestep, 5 is every 5 timesteps" />

2.3.3.3 Setup the source geometry

MOHID Lagrangian supports the following types of source geometries:

- Box
- Sphere
- Point

- Line
- KMZ-polygon (example: area emissions)
- XY-Polygon (example: area emissions)

- Point is given in geographical coordinates

  ```
  <point x="2.5" y="5.5" z="0.75" units_comment="(deg,deg,m)"/>
  ```

- Box:

  ```
        <box>
          <point x="-5.5" y="1.0" z="0" units_comment="(deg,deg,m)"/>
            <size x="0.5" y="3" z="4.5" units_comment="metres (m)"/>
          </box>
  ```

  The point defines the lower left corner (in geographical coordinates) and the size of the box is given in meters.

- Line

  ```
   <line>
     <pointa x="1" y="2" z="-10" units_comment="(deg,deg,m)"/>
     <pointb x="1.001" y="2.00001" z="7" units_comment="(deg,deg,m)"/>
   </line>
  ```

- Sphere

  To setup a sphere, you need a center point and a radius.

  ```
   <sphere radius="0.95" units_comment="metres (m)">
     <point x="9.05" y="2.0" z="0" units_comment="(deg,deg,m)"/>
   </sphere>
  ```

- KMZ-Polygon and XY-polygons

  To use a KMZ or MOHID XY polygon add the file path where your polygon file is stored. You can use the verticalBoundingBox block to fill the vertical space inside the poligon.

  ```
  <polygon>
          <file name="data/polygon1.xy"/>
  ```

```
                    <verticalBoundingBox min="0.0" max="0.0"/>

            </polygon>
```

Each Source can specify it's own 'resolution' or inherit the global resolution set in the 'simulation' block. The resolution can be isotropic:

```
<resolution dp="50" units_comment="metres (m)"/>
```

Or vary in direction:

```
<resolution x="80" y="50" z="150" units_comment="metres (m)"/>
```

Each geometry can have a pre-described movement given by either a csv or dat file, as you can find in the examples. To consider a moving emission geometry just use the following block:

```
        <positionTimeSeries>

         <file name="data/spill_trajectory.csv" />

        </positionTimeSeries>
```

All these options can be combined with time intervals where the source is active:

```
        <active start="15" end="end" comment="example: start='12.7' end='end'; start='0.0'
end='95' " units_comment="seconds (s)" />
```

These intervals can be given in absolute dates, or in seconds (relative to the beginning of the simulation), and an arbitrary number of intervals can be set.

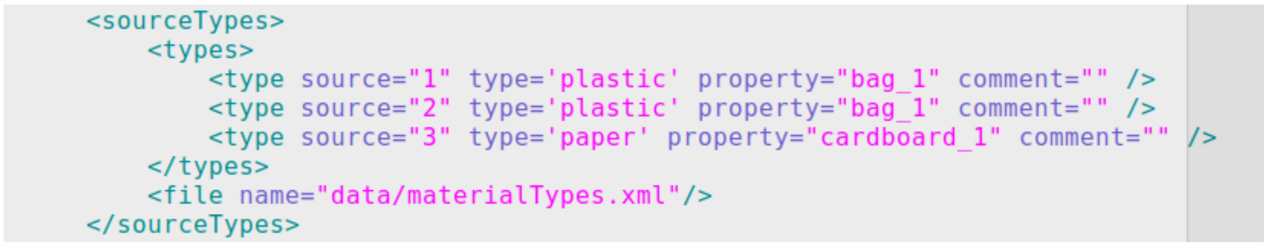
2.3.4 Setup the particle types

The <sourceTypes> contain the information about the particle type for each source. All the particle types must be placed in a xml file to act as a database for all simulations.

So the *<sourceTypes>* block has the following syntax:

```
   <sourceTypes>

     <types>

       <type source="1" type='plastic' property="bag_1" comment="" />

        <type source="2" type='plastic' property="bag_1" comment="" />

        <type source="3" type='paper' property="cardboard_1" comment="" />
```

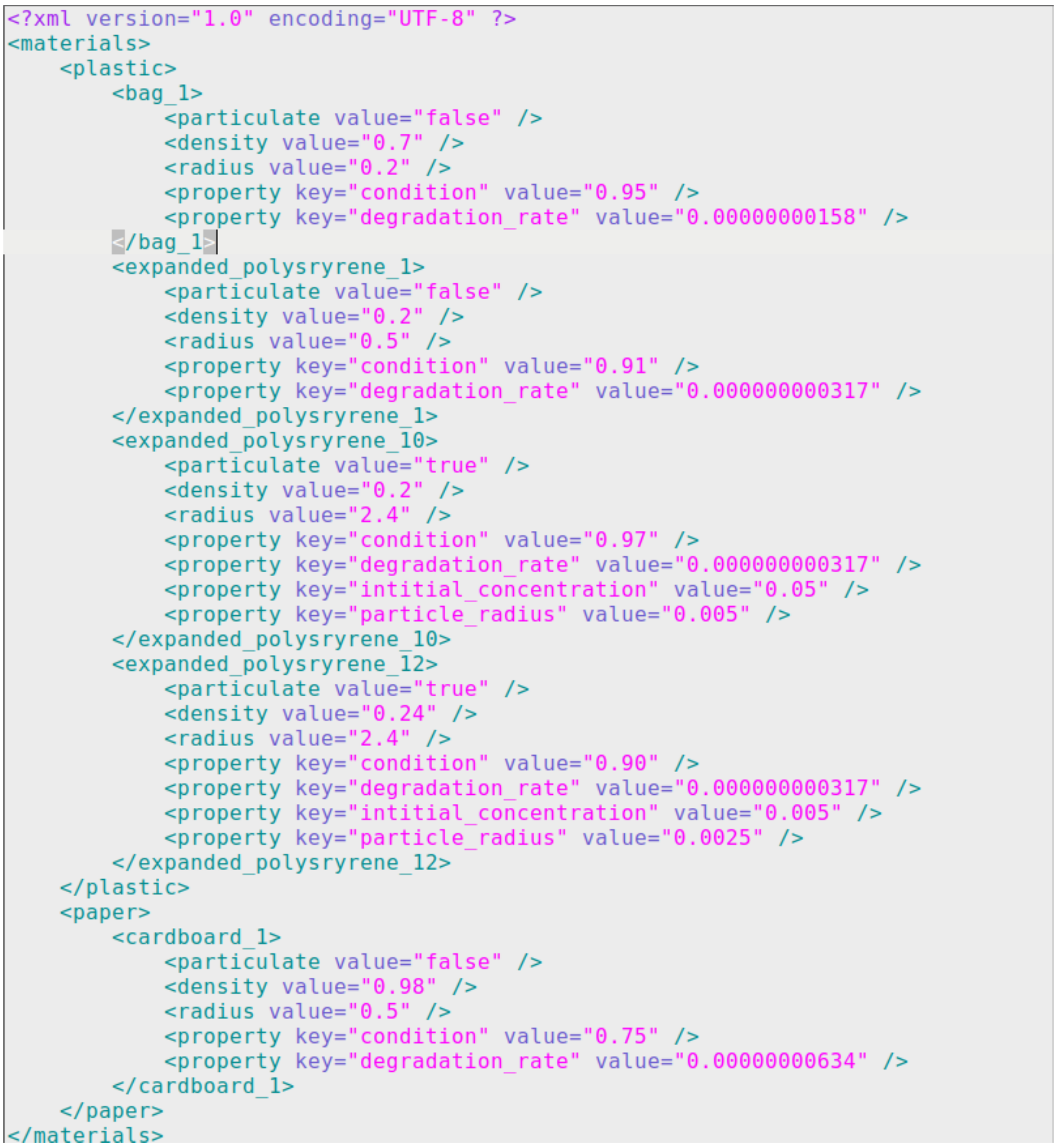
```
  </types>

  <file name="data/materialTypes_example.xml"/>

</sourceTypes>
```



```
<sourceTypes>
    <types>
        <type source="1" type='plastic' property="bag_1" comment="" />
        <type source="2" type='plastic' property="bag_1" comment="" />
        <type source="3" type='paper' property="cardboard_1" comment="" />
    </types>
    <file name="data/materialTypes.xml"/>
</sourceTypes>
```

The important key here is the *<file name="data/materialTypes_example.xml"/>.* It is used as a database for different types of particles. Each type has its own subtypes and corresponding properties, and these must be enumerated when saying that a source has a given type.

The library file looks like:



```xml
<?xml version="1.0" encoding="UTF-8" ?>
<materials>
    <plastic>
        <bag_1>
            <particulate value="false" />
            <density value="0.7" />
            <radius value="0.2" />
            <property key="condition" value="0.95" />
            <property key="degradation_rate" value="0.00000000158" />
        </bag_1>
        <expanded_polysryrene_1>
            <particulate value="false" />
            <density value="0.2" />
            <radius value="0.5" />
            <property key="condition" value="0.91" />
            <property key="degradation_rate" value="0.000000000317" />
        </expanded_polysryrene_1>
        <expanded_polysryrene_10>
            <particulate value="true" />
            <density value="0.2" />
            <radius value="2.4" />
            <property key="condition" value="0.97" />
            <property key="degradation_rate" value="0.000000000317" />
            <property key="intitial_concentration" value="0.05" />
            <property key="particle_radius" value="0.005" />
        </expanded_polysryrene_10>
        <expanded_polysryrene_12>
            <particulate value="true" />
            <density value="0.24" />
            <radius value="2.4" />
            <property key="condition" value="0.90" />
            <property key="degradation_rate" value="0.000000000317" />
            <property key="intitial_concentration" value="0.005" />
            <property key="particle_radius" value="0.0025" />
        </expanded_polysryrene_12>
    </plastic>
    <paper>
        <cardboard_1>
            <particulate value="false" />
            <density value="0.98" />
            <radius value="0.5" />
            <property key="condition" value="0.75" />
            <property key="degradation_rate" value="0.00000000634" />
        </cardboard_1>
    </paper>
</materials>
```

The different properties associated to each type of particle, are passed to the particle properties of the MOHID-Lagrangian code. To add new particles, just copy a block of code and rename it to fit in your needs. This part is still work in progress.

2.4 Setup the constants

The setup <constants> are global values affecting to the integration and should not be changed unless you need to adapt some values to your needs. The description in the comment section will guide you to make necessary changes.

For example, if you want to increase the diffusion effect change the:

<DiffusionCoeff value="0.75" comment="Horizontal diffusion coefficient. Default = 1.0" units_comment="m^2/s" />

3. Setup the Variable naming .xml library file.

This file controls the naming convention in your netCDF files. What is the role of this file? Imagine that you have a netCDF file, with a variable name for your hydrodynamic velocity fields or your depth dimension... How could MOHID-Lagrangian knows that your “strange_name_u_velocity_component” or your “z_evil_dimension” inside the netCDF file means “u_velocity_field” or simple “depth”?

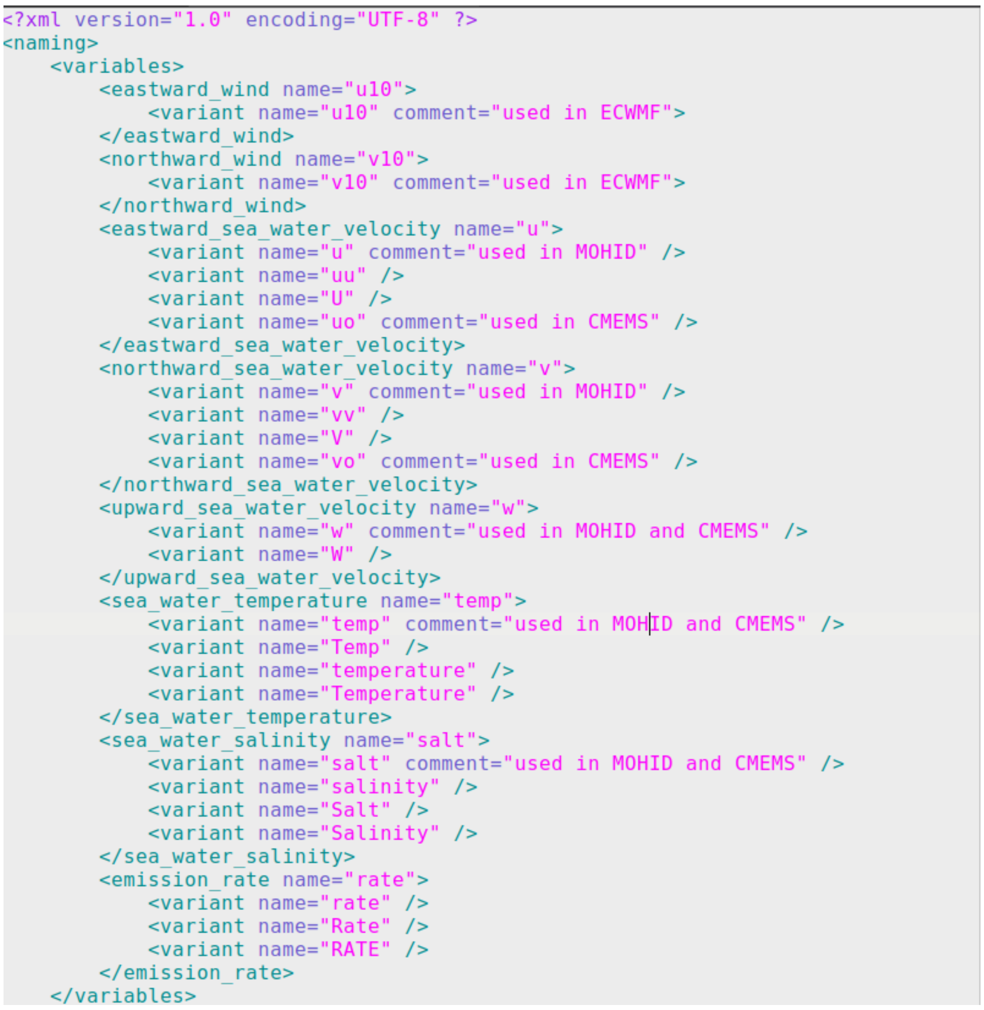
That is why this library file was created. **The xml file is dictionary/translator or dictionary between the CF compliant names for variables and dimensions and other variants for the variables that can appear for other netCDF outputs (which do not follow a common naming convention)**. The following pattern is used :

<standandard_cf_variable_name name=’variable_name’>

<variant name=’variable_name’>

</standandard_cf_variable_name >

Inside, it contains the following text:



```
<?xml version="1.0" encoding="UTF-8" ?>
<naming>
    <variables>
        <eastward_wind name="u10">
            <variant name="u10" comment="used in ECWMF">
        </eastward_wind>
        <northward_wind name="v10">
            <variant name="v10" comment="used in ECWMF">
        </northward_wind>
        <eastward_sea_water_velocity name="u">
            <variant name="u" comment="used in MOHID" />
            <variant name="uu" />
            <variant name="U" />
            <variant name="uo" comment="used in CMEMS" />
        </eastward_sea_water_velocity>
        <northward_sea_water_velocity name="v">
            <variant name="v" comment="used in MOHID" />
            <variant name="vv" />
            <variant name="V" />
            <variant name="vo" comment="used in CMEMS" />
        </northward_sea_water_velocity>
        <upward_sea_water_velocity name="w">
            <variant name="w" comment="used in MOHID and CMEMS" />
            <variant name="W" />
        </upward_sea_water_velocity>
        <sea_water_temperature name="temp">
            <variant name="temp" comment="used in MOHID and CMEMS" />
            <variant name="Temp" />
            <variant name="temperature" />
            <variant name="Temperature" />
        </sea_water_temperature>
        <sea_water_salinity name="salt">
            <variant name="salt" comment="used in MOHID and CMEMS" />
            <variant name="salinity" />
            <variant name="Salt" />
            <variant name="Salinity" />
        </sea_water_salinity>
        <emission_rate name="rate">
            <variant name="rate" />
            <variant name="Rate" />
            <variant name="RATE" />
        </emission_rate>
    </variables>
```

An example: Imagine that your future netCDF file/files contains the hydrodynamic fields and their names are **“utotal”** and **“vtotal”** to describe the CF compliant variables <eastward_sea_water_velocity> and <northward_sea_water_velocity> respectively. In that case, you should add the following to this file:

```
  <eastward_sea_water_velocity name="u">
    <variant name="u" comment="used in MOHID" />
    <variant name="uu" />
    <variant name="U" />
    <variant name="uo" comment="used in CMEMS" />
    <variant name="utotal">
  </eastward_sea_water_velocity>
  <northward_sea_water_velocity name="v">
    <variant name="v" comment="used in MOHID" />
    <variant name="vv" />
    <variant name="V" />
    <variant name="vo" comment="used in CMEMS" />
    <variant name="vtotal">
</northward_sea_water_velocity>
```

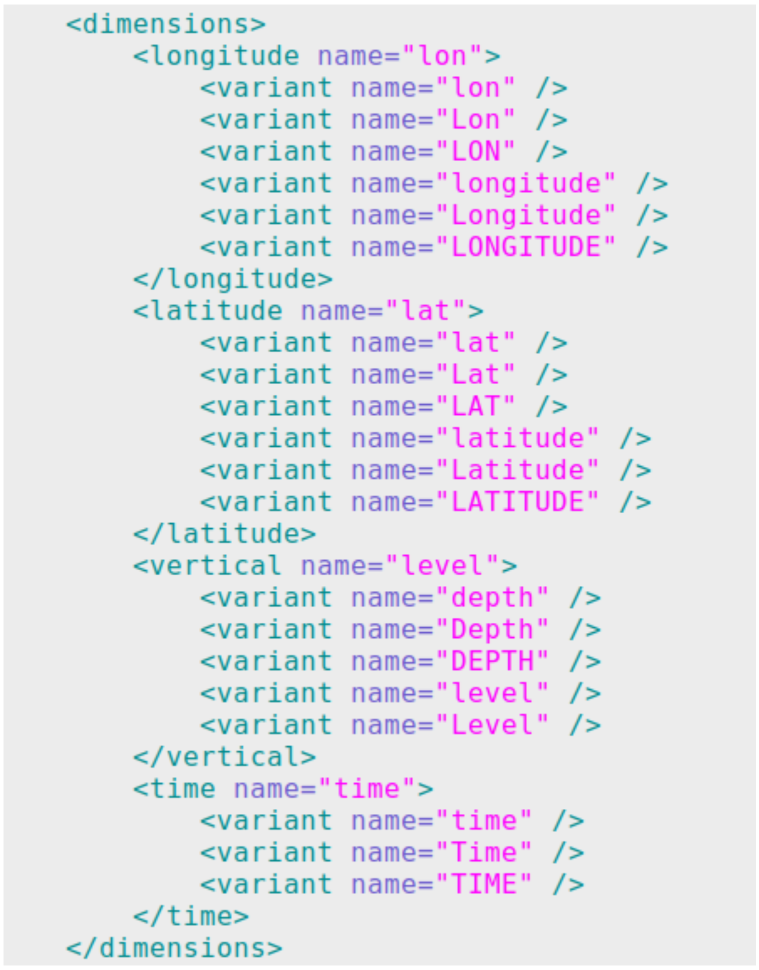
Another example:

If you want to add for example the waves effect through the *stokes velocity drift*, you should add the variant such as:

```
<stokes_velocity_drift_x name='vsdx'>
        <variant name='vsdx'
</stokes_velocity_drift_x name>
<stokes_velocity_drift_y name='vsdx'>
        <variant name='vsdy'
</stokes_velocity_drift_y name>
```

The variant name allows MOHID-Lagrangian to seek for those variant names of the standard name.

In the *<dimensions>,* it is the same case.



```
    <dimensions>
        <longitude name="lon">
            <variant name="lon" />
            <variant name="Lon" />
            <variant name="LON" />
            <variant name="longitude" />
            <variant name="Longitude" />
            <variant name="LONGITUDE" />
        </longitude>
        <latitude name="lat">
            <variant name="lat" />
            <variant name="Lat" />
            <variant name="LAT" />
            <variant name="latitude" />
            <variant name="Latitude" />
            <variant name="LATITUDE" />
        </latitude>
        <vertical name="level">
            <variant name="depth" />
            <variant name="Depth" />
            <variant name="DEPTH" />
            <variant name="level" />
            <variant name="Level" />
        </vertical>
        <time name="time">
            <variant name="time" />
            <variant name="Time" />
            <variant name="TIME" />
        </time>
    </dimensions>
```

Your depth dimension inside your netCDF file is *"z_depth".* You must add in the *<vertical name="level">* the following information:

```
 <vertical name="level">

        <variant name="depth" />

        <variant name="Depth" />

        <variant name="DEPTH" />

        <variant name="level" />

        <variant name="Level" />

          <variant name="z_depth" />

    </vertical>
```

4. Setup the output fields file

The block <outputFields> contains the file path of the xml file that controls the outputs of the variables to be written in the output files during a simulation run.

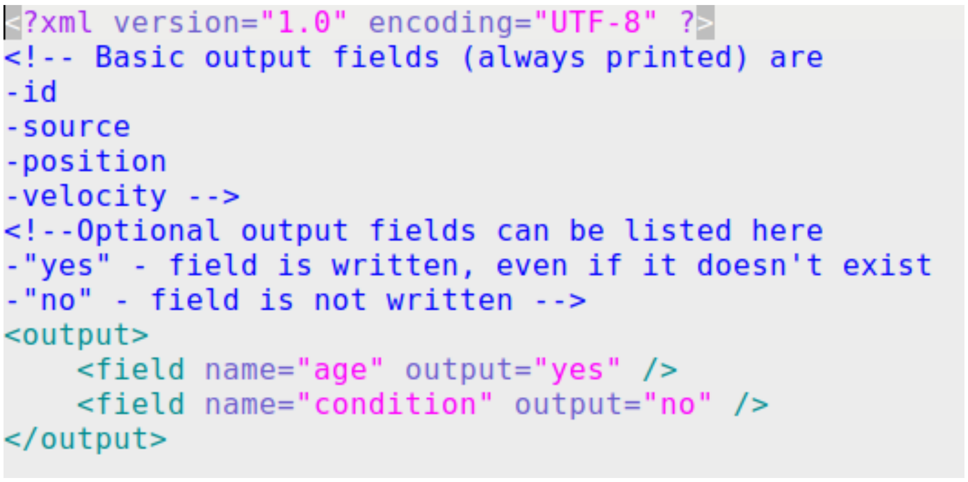
<outputFields>

<file name="data/outputFields.xml"/>

</outputFields>

If you change the path "data/outputFields.xml", please change the path/name.xml of the file outside the xml.

This file inside the path data contains:



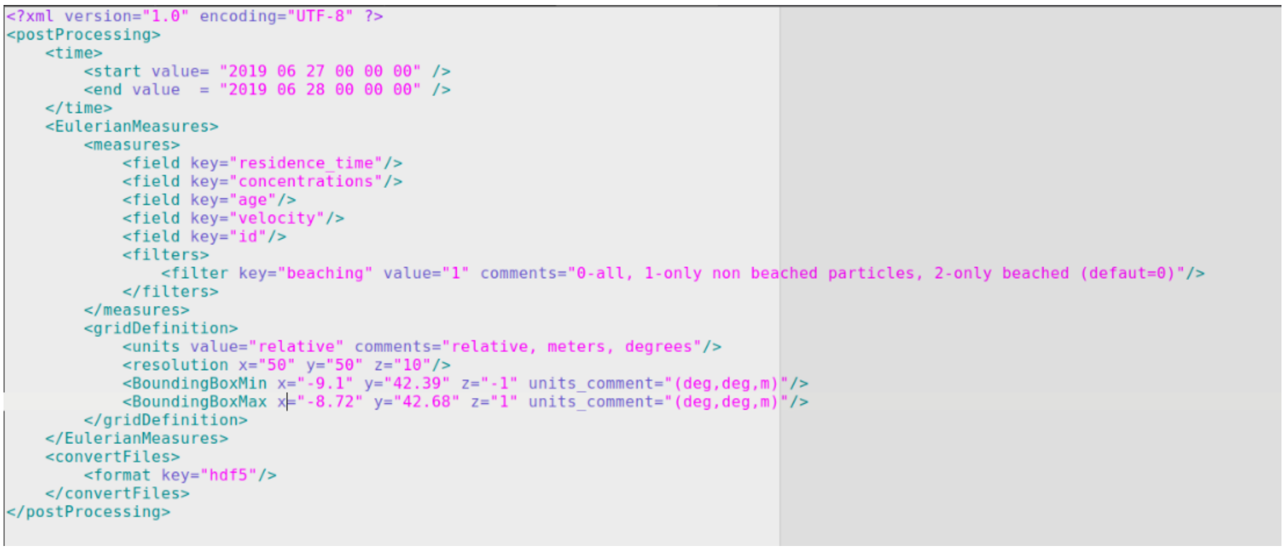
```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Basic output fields (always printed) are
-id
-source
-position
-velocity -->
<!--Optional output fields can be listed here
-"yes" - field is written, even if it doesn't exist
-"no" - field is not written -->
<output>
    <field name="age" output="yes" />
    <field name="condition" output="no" />
</output>
```

The *<output>* block controls when a variable is written to disk or not. As it is written in the xml comment section, the basic output fields are always written to disk.

5. Setup the post processor: recipes

Like the previous block, the *<postprocessing>* block controls the "recipes" file paths. The recipes are xml files controlling the post processing stage: the outputs fields produced, their type, and the conversion of the output files to other file formats. You can add many "recipes" with different setups. Each recipe produces one netCDF output field placed inside the output folder where the vtu files are stored and it contains the different fields specified in the xml recipe. The output will be written in a structured netCDF data with dimensions [time, depth, latitude, longitude] in a rectangular grid with boxes (depth, latitude, longitude) with a size specified inside the recipes.xml files.

5.1 Setup the recipes



```
<?xml version="1.0" encoding="UTF-8" ?>
<postProcessing>
    <time>
        <start value= "2019 06 27 00 00 00" />
        <end value  = "2019 06 28 00 00 00" />
    </time>
    <EulerianMeasures>
        <measures>
            <field key="residence_time"/>
            <field key="concentrations"/>
            <field key="age"/>
            <field key="velocity"/>
            <field key="id"/>
            <filters>
                <filter key="beaching" value="1" comments="0-all, 1-only non beached particles, 2-only beached (defaut=0)"/>
            </filters>
        </measures>
        <gridDefinition>
            <units value="relative" comments="relative, meters, degrees"/>
            <resolution x="50" y="50" z="10"/>
            <BoundingBoxMin x="-9.1" y="42.39" z="-1" units_comment="(deg,deg,m)"/>
            <BoundingBoxMax x="-8.72" y="42.68" z="1" units_comment="(deg,deg,m)"/>
        </gridDefinition>
    </EulerianMeasures>
    <convertFiles>
        <format key="hdf5"/>
    </convertFiles>
</postProcessing>
```

5.1.1 Recipe time range

The first part of the <postProcessing> block:

```
<time>
  <start value= "2018 01 01 00 00 00" />
  <end value  = "2018 02 01 00 00 00" />
</time>
```

Controls the time range of your simulation to send it to postprocessor. If you have a long simulation (one year for example), you can make a sub selection in time and focuses in one month (for example. Our simulation could go, in the CMEMS_case.xml, from *2017 01 01 00 00 00* to *2019 01 01 00 00 00* and here we decided to use particle trajectories in the date range 2018 01 01 to 2018 02 01). This block is optional.

5.1.2 Recipe output fields

The Eulerian measures controls the integration of particle measures/properties to instantaneous grid measures.

```
<measures>
  <field key="residence_time"/>
  <field key="concentrations"/>
  <field key="age"/>
  <field key="velocity"/>
```

```
      <field key="id"/>

      <filters>

        <filter key="beaching" value="1" comments="0-all, 1-only non beached particles, 2-only
beached (defaut=0)"/>

      </filters>

    </measures>
```

The field key="value" controls the variables to compute and add to the netCDF output file. If you include variables that are basics from *outputFields.xml* section, the postprocessor uses the particles each grid or cell and it makes an average to provide the average value inside the cell at each timestep. For *concentrations* and *residence_time* it will compute the number of particles per area/volume grid and the time that a cell is active by the presence of particles respectively.

If you do not want to compute a measure, you can delete the line. For example, if you do not want to compute the average ”id”, just delete the line from the xml.

```
    <measures>

      <field key="residence_time"/>

      <field key="concentrations"/>

      <field key="age"/>

      <field key="velocity"/>

      <filters>

        <filter key="beaching" value="1" comments="0-all, 1-only non beached particles, 2-only
beached (defaut=0)"/>

      </filters>

    </measures>
```

An important parameter here is the <filters> parameter. It allows you to control which particles are going to be used for postprocessing:

0) All particles (beached and non-beached).
1) Just particles that do not reach beach condition (non-beached).
2) Just particles that reach the beach condition (beached) .

For example, to measure the concentrations ONLY in beaches replace *value=”0”* with 2

```
      <filters>
```

```
        <filter key="beaching" value="2" comments="0-all, 1-only non beached particles, 2-only beached (default=0)"/>

    </filters>
```

5.1.3 Recipe grid region and resolution

The *<gridDefinition>* block controls how to slice your simulation domain of it into boxes or cells to count the particles and to obtain the associated quantities mentioned above.

```
  <gridDefinition>

    <units value="relative" comments="relative, meters, degrees"/>

    <resolution x="50" y="50" z="10"/>

    <BoundingBoxMin x="-9.1" y="42.39" z="-1" units_comment="(deg,deg,m)"/>

    <BoundingBoxMax x="-8.72" y="42.68" z="1" units_comment="(deg,deg,m)"/>

  </gridDefinition>
```

The <units> fields within the resolution, allows you to control how to split the domain.

1) “relative”: It slices the domain in 50 pieces in x direction, 50 pieces in y direction and 10 pieces in z direction using the bounding box domain limits.

2) “meters”: It slices the domain in steps of 50 meters in x direction, 50 meters in y direction and 10 meters in z direction

3) “degrees”: It slices the domain in steps of 50 degrees in x direction, 50 degrees in y direction and 10 meters in z direction.

The BoundingBoxMin and BoundingBoxMax, controls the domain where you want to perform the postprocess stage computation. If you do not provide a BoundingBox here, the MOHID-Lagrangian Postprocessor will take these limits from the *CMEMS_case.xml*. If your main boundingBox from CMEMS_case.xml takes the whole ocean, here you could select a box around the Azores island for example to compute concentrations around it.

5.1.4 Convert files to hdf5

This block controls the conversion of vtu files to MOHID compatible hdf5 formated files at the post processing stage.

```
 <convertFiles>

   <format key="hdf5"/>

 </convertFiles>
```

If you do not want to convert the vtu output files, just delete this block.