

MOHID LAGRANGIAN

V0.2 - A QUICK REFERENCE GUIDE

Contents

- Context & features
- Requirements
- Input files
- Case setup and example
- How to run a simulation
- Outputs
- Postprocessing

Context

MOHID Lagrangian is:

- A unidirectional Lagrangian particle simulator
- Made to easily extend the physical models acting on the particles
- Made to support large scale modelling in both space and time
- Made to support medium independent simulations
- Made to support large tracer numbers
- A cross platform, shared memory parallel tool

Features on V0.2

- Define tracer sources in space and time using basic shapes and file defined polygons
- Import currents data from .nc or .nc4 files, using CF conventions
- Import 3D/2D/regular/irregular structured meshes – not curvilinear
- Import data from a file series automatically
- One domain per simulation – no fitted or overlaid domains
- Provides 1st, 2nd and 4th order integrators
- Physics kernels for Lagrangian kinematics
- Automatic land, beaching and bed interaction masks (assuming CF compliant input files)

Features on V0.2

- Basic litter modelling physics kernels (degradation)
- Diffusion (random walk based on mixing length estimate)
- Beaching behaviour
- Windage and Stokes drift effects
- Python based, postprocessor suite in order to interpolate solutions and cast them on NetCDF CF compliant grids, so they can be published and explored

Requirements

To compile

- modern Fortran compiler - IFort 18+, GFortran 8+
- Cmake
- m4, autotools if using Linux
- Visual Studio if using windows

to Run

- python 3+ (with xarray, vtk and netcdf4 libs installed)
- text editor...

To process outputs

- Paraview

Input files

To set up a case:

- Case definition file (.xml)
- Tracer library file (optional, .xml)
- Variable output control file (optional, .xml)
- NetCDF variable library file (optional, .xml)
- Input data (.nc or .nc4 file/s)

Input files – case definition

A case definition file contains several sections and typically has the following structure

- Case
 - Execution
 - Parameters – dates, formats, numerical schemes, output frequency,...
 - Post processing cycles requests
 - Variable naming – file containing variable name correspondences in NetCDF files
 - Case definitions
 - Input data fields – directories where .nc files are present
 - Simulation parameters – spatial and temporal resolution, bounding box, ...
 - Source definitions – defining the name, emission rates, resolution, lifespan and geometry of a tracer source
 - Source type properties – linking a source to a specific type (plastic, paper, oil, ...)
 - Case constants – beaching level and intensity, diffusion coefficient, ...

Input files – tracer library

A tracer library file is a simplified tracer type database. The idea is that users can add tracer types and subtypes, each with their parameters and coefficients, to that library and use them on any case and share with other users.

```
<materials>
  <plastic>
    <bag_1>
      <particulate value="false" />
      <density value="0.7" />
      <radius value="0.2" />
      <condition value="0.95" />
      <degradation_rate value="1" />
    </bag_1>
    <expanded_polysryrene_1>
      <particulate value="false" />
      <density value="0.2" />
      <radius value="0.5" />
      <condition value="0.91" />
      <degradation_rate value="0.9" />
    </expanded_polysryrene_1>
  </plastic>
</materials>
```

```
</plastic>
<paper>
  <cardboard_1>
    <particulate value="false" />
    <density value="0.98" />
    <radius value="0.5" />
    <condition value="0.75" />
    <degradation_rate value="3.55" />
  </cardboard_1>
</paper>
</materials>
```

For v0.2, plastic and paper types are implemented, with simple, placeholder physical functions such as a degradation rate.

Input files – variable names library

A variable library file simply stores the match between the formal NetCDF CF name for a variable and the actual variants that different models write on their outputs. Any user can add to this file if they encounter outputs from different models, and as long as the output is CF compliant, no further pre-processing should be required.

```
<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>
```

```
<longitude name="lon">
  <variant name="lon" />
  <variant name="Lon" />
  <variant name="LON" />
  <variant name="longitude" />
  <variant name="Longitude" />
  <variant name="LONGITUDE" />
</longitude>
```

The name of the section is the CF long name, and the name field in front of it determines the name of that variable inside our simulation and its outputs.

The variants are a list of possible names for a given variable, that if found in a NetCDF file are used to read the fields.

Input files – .nc or .nc4 files

MOHID Lagrangian V0.2 consumes NetCDF CF files with arbitrary dimension fields. These are checked for correctness and processed if necessary during the input stages. The vertical dimension needs to be well formed however – MOHID was made to model any medium, in natural coordinates – if your ocean data vertical coordinate is written indistinguishably from an atmospheric data set, it will be imported as such. Use NCO or other tools to correct this in your files directly if necessary.

MOHID Lagrangian supports reading an arbitrary number of fields. If you have a collection of time steps spread across different files, these are ordered internally and read as required, automatically.

Case definition - example

Execution section of the Arousa2D test case configuration file

```
<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="Integrator" value="3" comment="Integration Algorithm 1:Euler, 2:Multi-Step Euler, 3:RK4 (default=1)" />
    <parameter key="Threads" value="4" comment="Computation threads for shared memory computation (default=auto)" />
    <parameter key="OutputWriteTime" value="3600" comment="Time out data (1/Hz)" units_comment="seconds" />
  </parameters>
  <outputFields>
    <file name="data/outputFields.xml"/>
  </outputFields>
  <variableNaming>
    <file name="data/NamesLibrary.xml"/>
  </variableNaming>
  <postProcessing>
    <file name="Post_scripts/PostRecipe_Arousa.xml"/>
    <file name="Post_scripts/PostRecipe_Arousa_2.xml"/>
  </postProcessing>
</execution>
```

File listing variables to output

List of post processing recipes to run (add here recipes to run them in batch at simulation end)

Case definition - example

Case definitions of the Arousa2D test case configuration file

```
<caseDefinitions>
  <inputData>
    <inputDataDir name="nc_fields" type="hydrodynamic"/>
  </inputData>
  <simulation>
    <resolution dp="50" units_comment="metres (m)"/>
    <timestep dt="1200.0" units_comment="seconds (s)"/>
    <BoundingBoxMin x="-9.1" y="42.40" z="-1" units_comment="(deg,deg,m)"/>
    <BoundingBoxMax x="-8.75" y="42.66" z="1" units_comment="(deg,deg,m)"/>
  </simulation>
  <sourceDefinitions>
    <source>
      <setsource id="1" name="Box1" />
      <resolution x="50" y="200" z="10" units_comment="metres (m)"/>
      <rate_dt value="10000" comment="number of timesteps / emission. 1 is every timestep, 5 is every 5 timesteps" />
      <active start="0" end="end" comment="example: start='12.7' end='end'; start='0.0' end='95' " units_comment="seconds (s)" />
      <box>
        <point x="-9.06" y="42.42" z="0" units_comment="(deg,deg,m)"/>
        <size x="10000" y="10000" z="1.0" units_comment="metres (m)"/>
      </box>
    </source>
  </sourceDefinitions>
</caseDefinitions>
```

Input data directory and type

Global initial resolution (uniform or not)

Global bounding box – any tracer crossing this will be excluded from the simulation

Case definition - example

Source definitions of the Arousa2D test case configuration file

```
<source>
  <setsource id="1" name="Box1" />
  <resolution x="50" y="200" z="10" units_comment="metres (m)" />
  <rate_dt value="10000" comment="number of timesteps / emission. 1 is every timestep, 5 is every 5 timesteps" />
  <active start="0" end="end" comment="example: start='12.7' end='end'; start='0.0' end='95' " units_comment="seconds (s)" />
  <box>
    <point x="-9.06" y="42.42" z="0" units_comment="(deg,deg,m)" />
    <size x="10000" y="10000" z="1.0" units_comment="metres (m)" />
  </box>
</source>
<source>
  <setsource id="2" name="ReleaseLine10" />
  <rate value="0.001" comment="emission rate (Hz)" />
  <active start="0" end="5000" comment="example: start='12.7' end='end'; start='0.0' end='95' " units_comment="seconds (s)" />
  <active start="10000" end="20000" comment="example: start='12.7' end='end'; start='0.0' end='95' " units_comment="seconds (s)" />
  <active start="2018 01 12 00 00 00" end="2018 02 25 00 00 00" comment="example: start='12.7' end='end'; start='0.0' end='95' " units_comment="seconds (s)" />
  <active start="2018 02 02 00 00 00" end="end" comment="example: start='12.7' end='end'; start='0.0' end='95' " units_comment="seconds (s)" />
  <line>
    <pointa x="-8.88" y="42.5" z="0" units_comment="(deg,deg,m)" />
    <pointb x="-8.87" y="42.525" z="0" units_comment="(deg,deg,m)" />
  </line>
</source>
```

Override of global resolution for this source

Distinct types of emission rate definitions

Series of on/off cycles for this source

Case definition - example

Source definitions of the Arousa2D test case configuration file

```
<source>
  <setsource id="3" name="ReleasePoint17" />
  <rateTimeSeries>
    <file name="data/discharge_example.csv" comment="name of csv file with discharge information (time and rate columns)"/>
  </rateTimeSeries>
  <point x="-8.92" y="42.56" z="0" units_comment="(deg,deg,m)"/>
</source>
<source>
  <setsource id="5" name="ReleasePoint5" />
  <rateTimeSeries>
    <file name="data/discharge_example2.dat" />
    <scale value="0.1" comment="scales the data on the file by this factor (not time)" />
  </rateTimeSeries>
  <point x="-8.94" y="42.56" z="0" units_comment="(deg,deg,m)"/>
  <positionTimeSeries>
    <file name="data/spill_trajectory.dat" />
  </positionTimeSeries>
</source>
```

Emission rate mediated by a csv file (time(s), rate(Hz))

Emission rate mediated by a MOHID time series file.
You can add the name of the variable you want to import as rate to the naming .xml file

Optional scale to multiply the imported variable with (transforming river discharge into rate for example)

Position time series (either .csv or MOHID time series) for this source

Case definition - example

Source type definitions of the Arousa2D test case configuration file – optional section

Source 3 will imprint on emitted tracers the properties of 'bag_1', of type 'plastic'. This will lead to differentiated behavior with other types (different processes) and subtypes (different parameters)

```
<sourceTypes>
  <types>
    <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>
```

Type library file

Case definition – useful pointers

- File names and directories can be put in as absolute or relative paths
- The naming conventions can be imported from several files, just add another to the section to append more options
- If you get an error saying a .xml file doesn't exist while running MOHID Lagrangian, it's probably not correctly formatted. Make sure all sections are properly open and closed (</>)
- Comments and units are there for user convenience, they don't interact with the simulation

Case setup – workflow example

- Create a directory to run your case – it will typically hold configuration, input and output files, but this is entirely up to the user. The examples directory in V0.2 is 'RUN_Cases'. Create a 'Test' directory there.
- Create a configuration .xml file inside 'Test'. You can copy the one from 'Arousa_2D' and modify it:

```
<simulation>
  <resolution dp="50" 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>
```

```
<simulation>
  <resolution x="50" y="100" z="10" units_comment="metres (m)"/>
  <timestep dt="900.0" units_comment="seconds (s)"/>
  <BoundingBoxMin x="-9.1" y="42.40" z="-1" units_comment="(deg,deg,m)"/>
  <BoundingBoxMax x="-8.75" y="42.66" z="1" units_comment="(deg,deg,m)"/>
</simulation>
```



Change spatial and temporal resolutions – notice resolution can vary by direction or be uniform

Case setup – workflow example

Make source id=3 active only after 298 seconds and change using 4 threads to automatic use of CPU cores

```
<source>
  <setsource id="3" name="ReleasePoint17" />
  <rateTimeSeries>
    <file name="data/discharge_example.csv" comment="name of csv file with
  </rateTimeSeries>
  <point x="-8.92" y="42.56" z="0" units_comment="(deg,deg,m)"/>
</source>
```

```
<source>
  <setsource id="3" name="ReleasePoint17" />
  <active start="298" end="end" comment="example: start='12.7' end='end';
  <rateTimeSeries>
    <file name="data/discharge_example.csv" comment="name of csv file w
  </rateTimeSeries>
  <point x="-8.92" y="42.56" z="0" units_comment="(deg,deg,m)"/>
</source>
```



```
<parameters>
  <parameter key="Start" value="2018 01 02 00 00 00" comment="Date of :
  <parameter key="End" value="2018 02 04 00 00 00" comment="Date of :
  <parameter key="Integrator" value="3" comment="Integration Algorithm
  <parameter key="Threads" value="4" comment="Computation threads for :
  <parameter key="OutputWriteTime" value="3600" comment="Time out data
</parameters>
```

```
<parameters>
  <parameter key="Start" value="2018 01 02 00 00 00" comment="Date of initial ins
  <parameter key="End" value="2018 02 04 00 00 00" comment="Date of final insta
  <parameter key="Integrator" value="3" comment="Integration Algorithm 1:Euler, 2
  <parameter key="Threads" value="auto" comment="Computation threads for shared m
  <parameter key="OutputWriteTime" value="3600" comment="Time out data (1/Hz)" ur
</parameters>
```

Case setup – workflow example

Notice you can set an arbitrary amount of active intervals, both is relative time (seconds from the beginning of the simulation) or absolute with dates

```
<source>
  <setsource id="2" name="ReleaseLine10" />
  <rate value="0.001" comment="emission rate (Hz)" />
  <active start="0" end="5000" comment="example: start='12.7' end='end'; start='0.0' end='95' " units_comment="seconds (s)" />
  <active start="10000" end="20000" comment="example: start='12.7' end='end'; start='0.0' end='95' " units_comment="seconds (s)" />
  <active start="2018 01 12 00 00 00" end="2018 02 25 00 00 00" comment="example: start='12.7' end='end'; start='0.0' end='95' " units_comment="seconds (s)" />
  <active start="2018 02 02 00 00 00" end="end" comment="example: start='12.7' end='end'; start='0.0' end='95' " units_comment="seconds (s)" />
  <line>
    <pointa x="-8.88" y="42.5" z="0" units_comment="(deg,deg,m)" />
    <pointb x="-8.87" y="42.525" z="0" units_comment="(deg,deg,m)" />
  </line>
</source>
<source>
```

Case setup – workflow example

What about input files?

You can either copy all of the files to the new directory, or put the paths of where they already are in the configuration file.

Using relative paths to the old directory:

```
<inputData>
  <inputDataDir name="../../Arousa_2D_test_case/nc_fields"/> type="hydrodynamic"/>
</inputData>

<source>
  <setsource id="3" name="ReleaseBox17" />
  <rate_file name="../../Arousa_2D_test_case/data/discharge_example.csv" comment="name
  <active start="298" end="end" comment="example: start='12.7' end='end'; start='0.0
  <point x="-8.92" y="42.56" z="0" units_comment="(deg,deg,m)"/>
</source>

<sourceTypes>
  <types>
    <type source="2" type='plastic' property="bag_1" comment="" />
    <type source="3" type='paper' property="cardboard_1" comment="" />
  </types>
  <file name="../../Arousa_2D_test_case/data/materialTypes.xml"/>
</sourceTypes>
<variableNaming>
  <file name="../../Arousa_2D_test_case/data/ncNamesLibrary.xml"/>
</variableNaming>
```

Case setup – workflow example

What about different input files?

You must tag the directory containing the files with the type – hydrodynamic, waves, meteorology, water properties...

This is so they are correctly imported and used accordingly. You can set an arbitrary number of input directories, of any type. All valid .nc or .nc4 files under those (or under subdirectories) will be listed for import and used if needed.

```
<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>
```

V0.2 does NOT support multiple domains (either fitted or overlapping). All files of the same type must be from the same mesh. You can use different spatial and temporal discretizations across file types (the hydrodynamic solution can be 3D full north Atlantic and the meteorology is 2D and cover only a section of it, for example).

How to run a simulation

How do I now run MOHID Lagrangian?

The typical chain is data – setup – preprocessor – MOHID Lagrangian - postprocessor

In the examples there are windows (.bat) and unix (.sh) scripts to run the these steps for a given case, as well as separate ones to run post processing, that can be used while a simulation is running.

You need to make sure your configuration file name is correct and the directory to the executables are well set.

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

set name=Arousa2D_case
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"
```

← Name of the case and configuration file (the name of your main .xml file)

← Path to the executables

How to run a simulation

```
set name=Arousa2D_case
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"

set postProcessorDir=../../src/MOHIDLagrangianPostProcessor
set postProcessor="%postProcessorDir%/MOHIDLagrangianPostprocessor.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 %PreProcessor% -i %dirout%/ %name%.xml -o %dirout%

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

python -W ignore %postProcessor% -i %name%.xml -o %dirout%
```

Name of the case and configuration file (the name of your main .xml file, all you need to change in this script to run another case)

Path to the executables – change this to point to the executables if you want to run your cases in some other directory tree

Creates the output directory – deletes old results if they are present!

Running pre processor, MOHID Lagrangian and post processor


Outputs

An output directory will be created (or emptied if it already exists), with the same name as the case, appended with '_out'.

In there several files will be created:

- A copy of your case configuration file
- A 'casename'.log file with all of the console output
- A preprocessor output called 'casename'_inputs.xml
- A Bounding box and blocks .vtu files (these are just for visualization)
- A series of .vtu files with simulation heavy data (all the particles and variables)
- A .pdv file, with simulation light data (indexes and time-stamps heavy data)
- A directory for each post processing request, with processed data

This lists and sorts all of the input NetCDF files so MOHID Lagrangian know what to read. It also shows the dates of the files, so you can correct your configuration start and end dates based on this output.

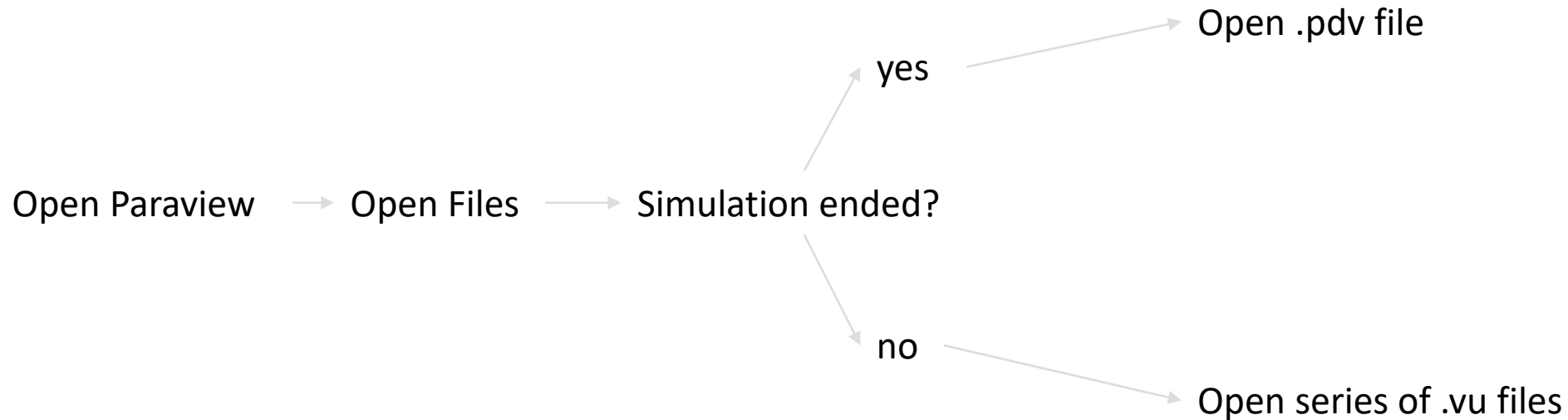


You need Paraview to plot this



Outputs – Using Paraview

Plenty of Paraview tutorials exist online, it is a mature software and is used by many to view, post-process, explore and render scientific spatial data. It is useful in our context – it reads NetCDF CF files and can read our output files (.vtu) with millions of particles on screen.



Outputs – Using Paraview - tips

- Paraview is coordinate system agnostic – This means that our horizontal units (degrees) are not compatible with our vertical units (m) – apply a transform filter and scale your data on the horizontal dimension so it looks appropriate.
- At the moment animating currents and particles simultaneously (synchronized in time) is not trivial – Our time stamp is fixed (seconds since 1950-01-01) and the time stamp of the input NetCDF may be different, or with different units (hours instead of seconds). Use NCO to change the input to match or wait for further post-processing tools
- Plotting the bathymetry in 3D (if available on a NetCDF file): read the file, set dimensions to the 2D plane (lon, lat typically), not as spherical coordinates. Use the *warp by scalar* filter to deform the bathymetry field using a given scale.

Post processing

- At the moment, to explore raw data Paraview is the gate for post-processing. Explore the filters and settings, it is a very complete suite.
- Paraview supports exporting data to other formats. Using routines you already have may require exporting particle data as a .csv file and using it with your other codes
- Paraview supports python scripting: a suite of scripts is being prepared to simplify and automate typical workflows

V0.2 introduces our post processor

- Interpolates raw data to a grid (Lagrangian to Eulerian results), both globally and by source
- Produces cell averages of any output variable and creates synthetic variables (concentration, residence time)
- Writes the data in OpenDAP, Thredds ready netCDF CF – results are immediately publishable and importable in any compliant framework
- Converts raw data to MOHID formatted .hdf5 files

Post processing

Post processing works by 'scripting' a recipe to follow, and the user can design an arbitrary number of recipes and request them to be run automatically after the simulation. These can also be run during runtime, on the available files.

This is an example recipe file.

It interpolates velocity magnitudes, on a mesh defined by the bounding box of the simulation it is attached to, discretized in 100x100 cells.

```
<postProcessing>
  <EulerianMeasures>
    <measures>
      <field key = "velocity"/>
    </measures>
    <gridDefinition>
      <units value= "relative" comments="relative, meters, degrees"/>
      <resolution x="100" y="100" z="1"/>
    </gridDefinition>
  </EulerianMeasures>
</postProcessing>
```

Post processing

```
<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 (default=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>
```

→ Selecting a time interval to process

→ Selecting the variables to process

→ Filtering the results by beached status

→ Selecting a space interval to process

→ Converting all the files to MOHID .hdf5 Lagrangian format