MOHID LAGRANGIAN

V0.1 - A QUICK REFERENCE GUIDE

Contents

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 V0.1

- Define tracer sources in space and time using basic shapes
- Import currents data from .nc or .nc4 files, using CF conventions
- Import 3D/2D/regular/irregular structured meshes
- Import data from a file series automatically (constant dt)
- One domain per simulation
- Provides 1st, 2nd and 4th order integrators
- Physics kernels for Lagrangian kinematics and basic isotropic diffusion (disabled)
- Automatic land, beaching and bed interaction masks (assuming CF compliant input files)

Features on V0.2 – ETA August 2019

- Basic litter modelling physics kernels (settling velocity, buoyancy, degradation)
- Advanced diffusion kernels (random walk based on mixing length and turbulent velocities)
- Beaching behaviors
- Windage and Stokes drift effects
- Basic, 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 and netcdf libs installed)
- text editor...

To process outputs

- Paraview

Input files

To set up a case:

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

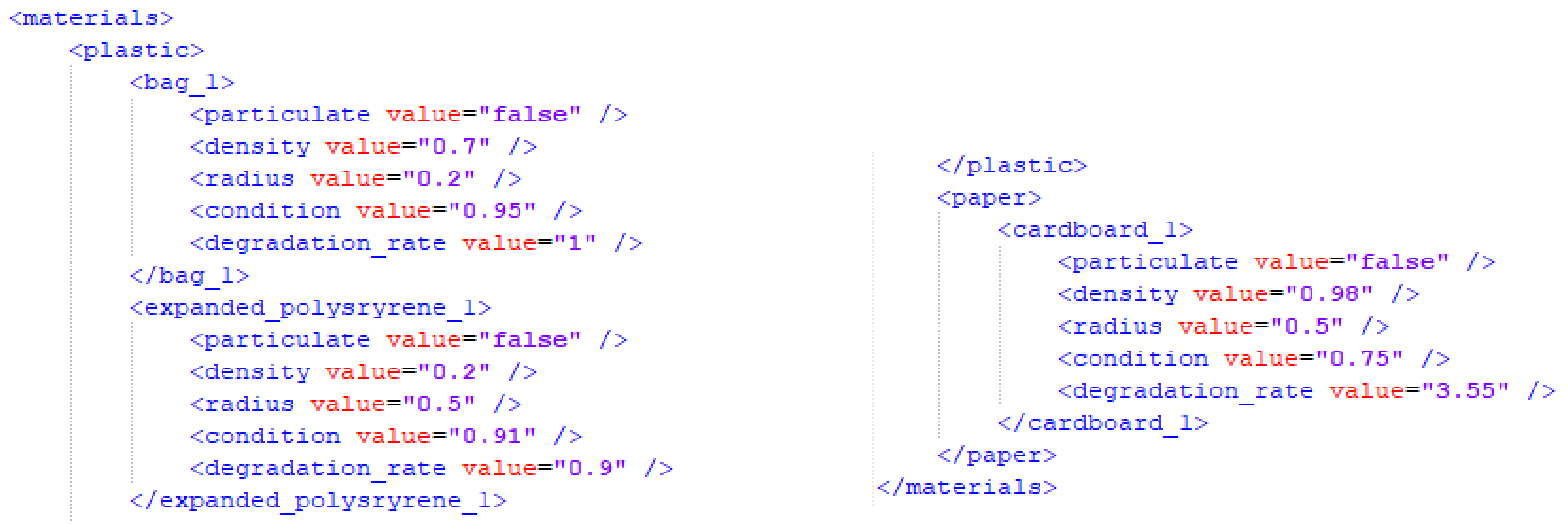
Input files – case definition

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

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

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.



```xml
<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>
    <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>
```

The structure of this file will suffer changes as new types and subtypes are added, for V0.1 this is just a placeholder as no tracer types are differentiated during a simulation.

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.

```xml
<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>
```

```xml
<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 V0.1 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 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. A restriction of V0.1 is that the time-step should be constant.

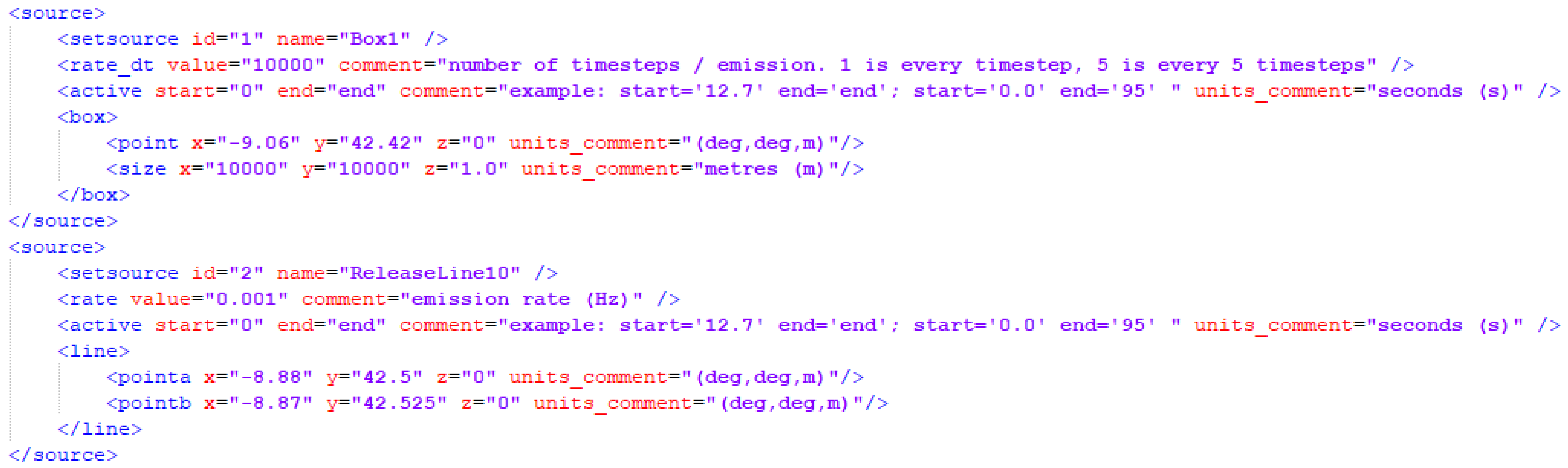
Case definition - example

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="auto" comment="Computation threads for shared memory computation (default=auto)" />
        <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>
</execution>
```

Case definition - example

Source definitions of the Arousa2D test case configuration file



```xml
<source>
    <setsource id="1" name="Box1" />
    <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="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>
```

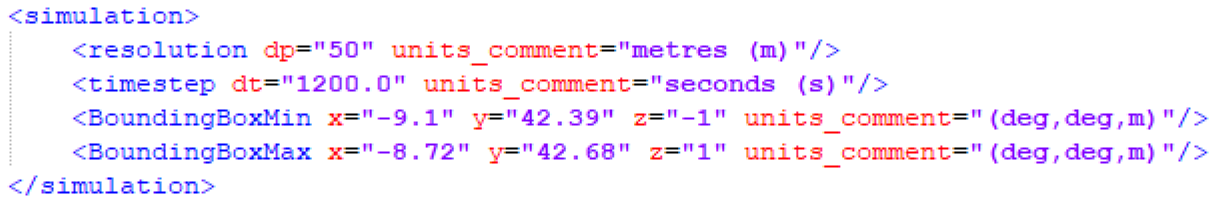
Case definition – useful pointers

- 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
- 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

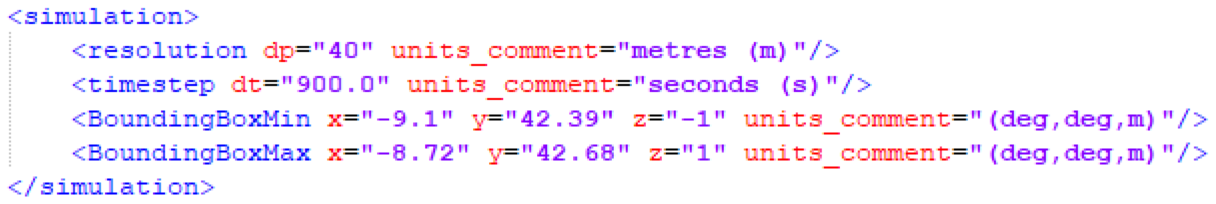
Case setup – workflow example

- 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 V0.1 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>
```

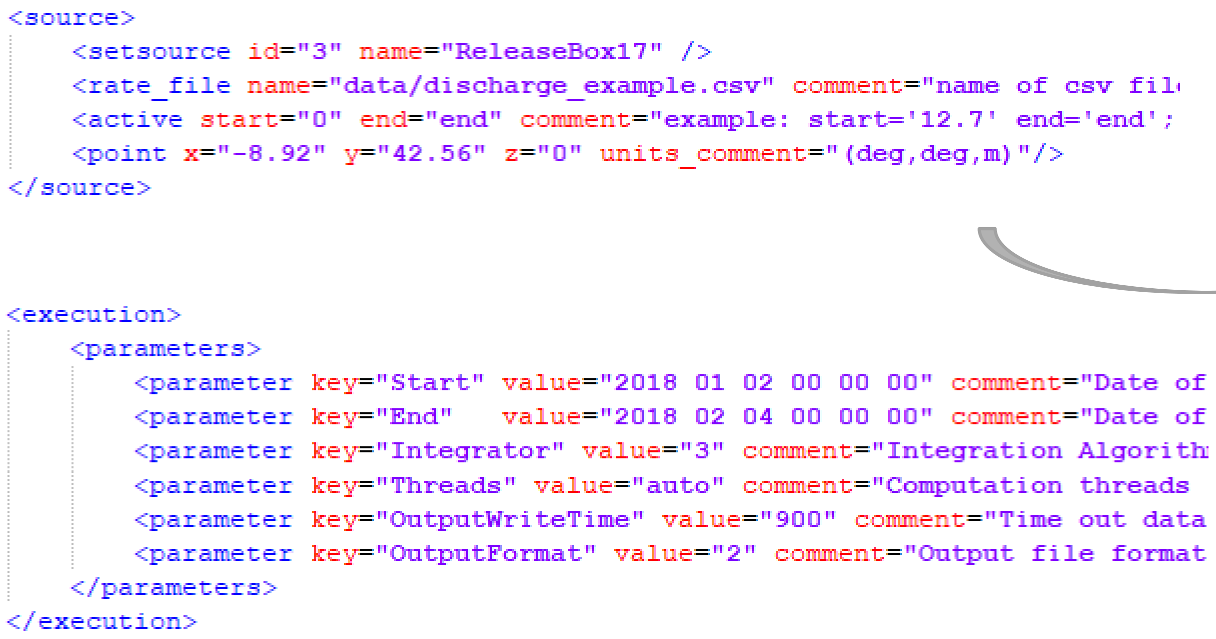


```
<simulation>
    <resolution dp="40" units_comment="metres (m)"/>
    <timestep dt="900.0" units_comment="seconds (s)"/>
    <BoundingBoxMin x="-9.1" y="42.39" z="-1" units_comment="(deg,deg,m)"/>
    <BoundingBoxMax x="-8.72" y="42.68" z="1" units_comment="(deg,deg,m)"/>
</simulation>
```

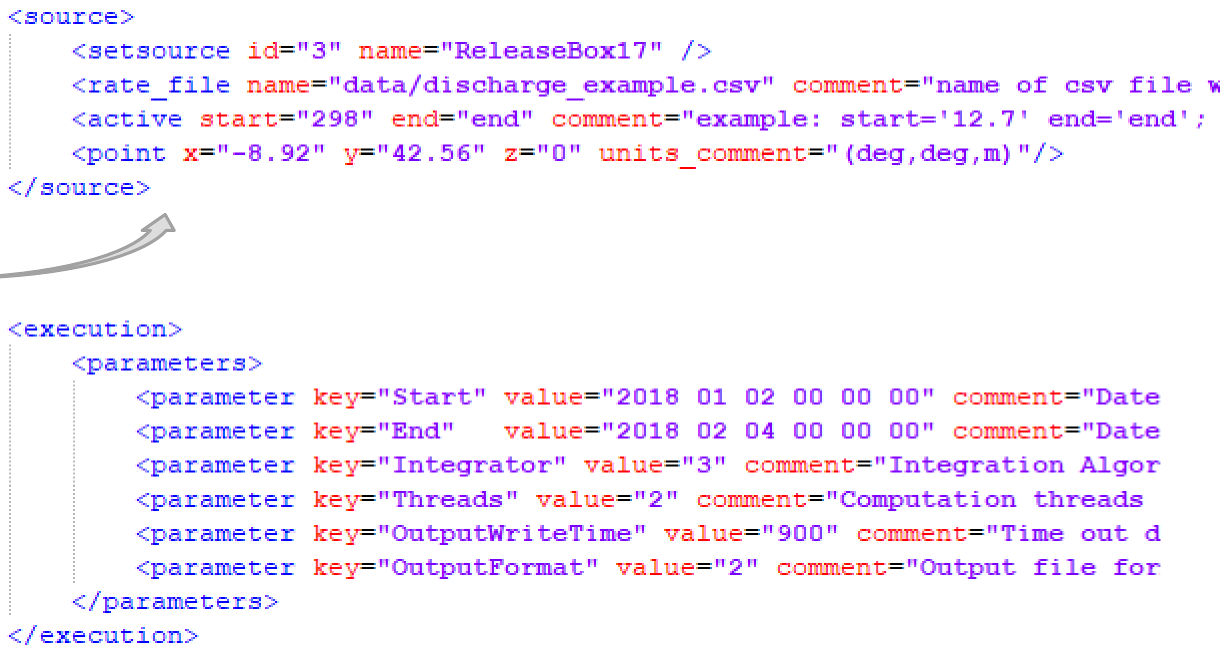
Change spatial and temporal resolutions

Case setup – workflow example

Make source id=3 active only after 298 seconds and change automatic use of CPU cores to using only 2



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

<execution>
    <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 Algorith
        <parameter key="Threads" value="auto" comment="Computation threads
        <parameter key="OutputWriteTime" value="900" comment="Time out data
        <parameter key="OutputFormat" value="2" comment="Output file format
    </parameters>
</execution>
```



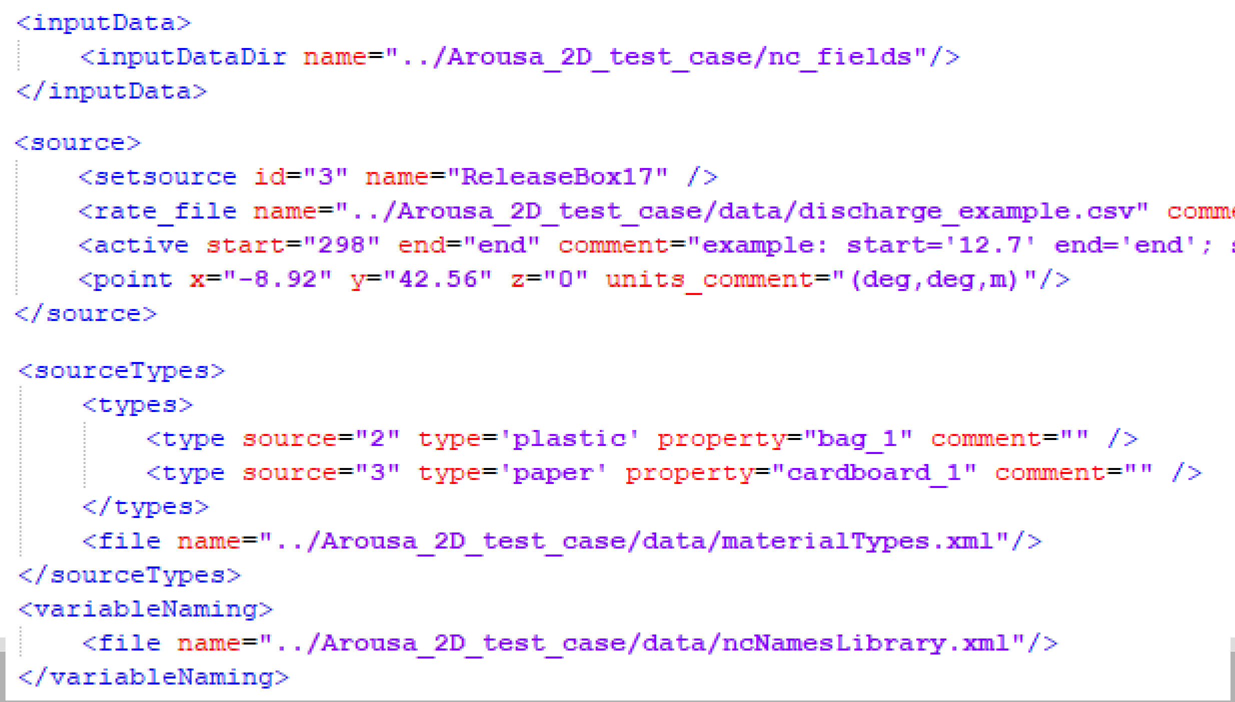
```
<source>
    <setsource id="3" name="ReleaseBox17" />
    <rate_file name="data/discharge_example.csv" comment="name of csv file w
    <active start="298" end="end" comment="example: start='12.7' end='end';
    <point x="-8.92" y="42.56" z="0" units_comment="(deg,deg,m)"/>
</source>

<execution>
    <parameters>
        <parameter key="Start" value="2018 01 02 00 00 00" comment="Date
        <parameter key="End"   value="2018 02 04 00 00 00" comment="Date
        <parameter key="Integrator" value="3" comment="Integration Algor
        <parameter key="Threads" value="2" comment="Computation threads
        <parameter key="OutputWriteTime" value="900" comment="Time out d
        <parameter key="OutputFormat" value="2" comment="Output file for
    </parameters>
</execution>
```

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.

I don’t want to copy files, so I just used relative paths to the old directory:



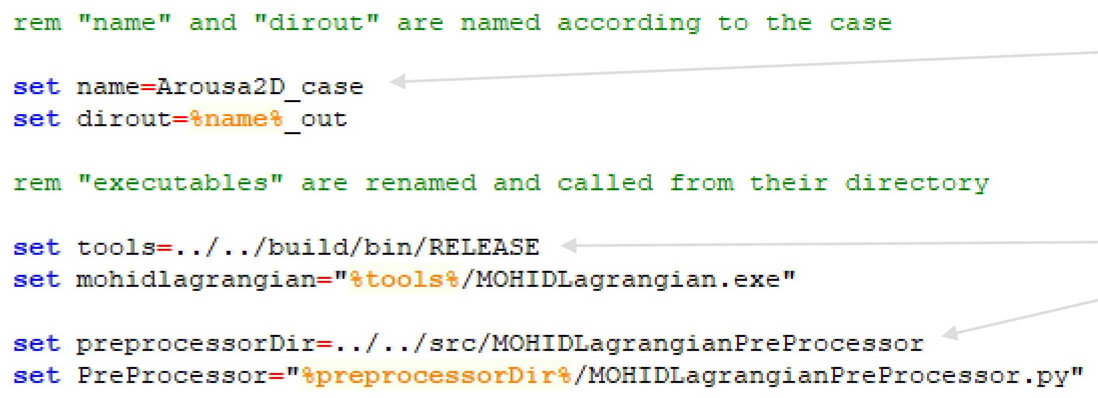
```xml
<inputData>
    <inputDataDir name="../Arousa_2D_test_case/nc_fields"/>
</inputData>

<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

How do I now run MOHID Lagrangian?

The typical chain is data – setup – preprocessor – MOHID Lagrangian.

In the examples there are windows (.bat) and unix (.sh) scripts to run the last two steps for your new case. All you need to do is 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

Path to the executables

Just run the script!

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)

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.

$$\begin{tikzcd}
 & & & \text{yes} \arrow[r] & \text{Open .pdv file} \\
\text{Open Paraview} \arrow[r] & \text{Open Files} \arrow[r] & \text{Simulation ended?} \arrow[ur] \arrow[dr] & & \\
 & & & \text{no} \arrow[r] & \text{Open series of .vu files}
\end{tikzcd}$$

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 in V0.2

- Ploting 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, 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