MOHID Lagrangian

Introduction of v0.3

Ricardo Birjukovs Canelas

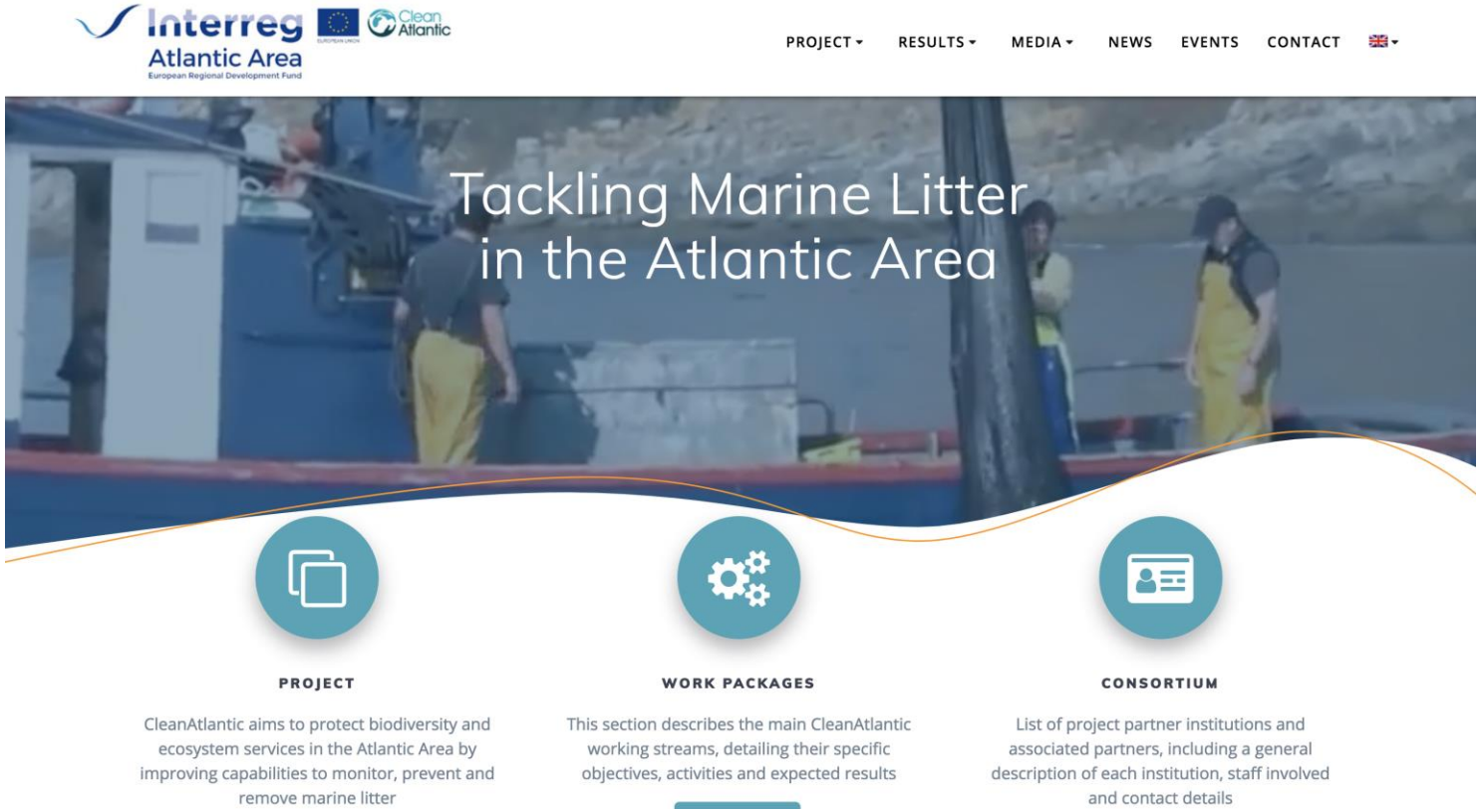
Daniel Garaboa Paz

Bentley Systems, Lisbon, Portugal

USC, GFNL, Santiago de Compostela

Immediate motivation – modelling litter at large scales

- Develop modelling methodologies and capabilities to tackle a domain such as the Atlantic ocean;
- Model **several types** of litter and their evolution in time (degradation, biofouling, aging, etc);
- Identify accumulation zones, account for main sources and predict trends.



What it *is* and *is not*

- Not a replacement for Lagrangian Modules (for now)
- Not an online tool
- Not full of physics for oil, larvae, water quality, outfalls, etc
- Not integrated in the MOHID system

- A high performance offline tool
- Medium independent – it should work in atmosphere, water, land,…
- Full of ideas on how to organize complex data and scenarios
- Easy to include new physics
- Built around standards – not a single made up format is used
- Model agnostic – whe have used data produced by Mohid, CMEMS, Delft3D and IBIS

I/O

- Case definitions input is done by .xml files
  - Exactly like the MOHID files, except everyone can read/write them
- Data input is exclusively done by NetCDF CF compliant files. MOHID has a good converter, so do most models. If it's on a THREDDS server, odds are you can use it directly

- Raw outpts are written to a vtk binary file
- One file has one time-step
- File timestamping is compliant with Copernicus
- Post-processed outputs are written to NetCDF CF compliant files. You can place them directly on a THREDDS server

Available functionalities

Sources

- Boxes, points, lines, polylines, spheres
- and polygons
- One source - one type of tracer
- Arbitrary lifespan intervals
- Arbitrary emission rate

Input files

- NetCDF CF
- Single or list of files
- Only structured mesh
- Regular/irregular mesh
- 2D/3D

Integrators

- Euler (1st order)
- Multi-step Euler (2nd order)
- Runge-Kutta 4 (4th Order)

Available functionalities

Masking

- Automatic land masks
- Detection of beaching zone
- Inclusion of inter-tidal areas
- Detection of bed interaction zones

Performance

- Depends on resolution of hydro files
- Tested to 30 million tracers
- Multi-threaded code, designed for
- shared and distributed memory
- machines
- Only shared memory is implemented
- (OpenMP)

Available functionalities

Kernels - the model is desiged around a dynamical systems approach. Having a State Vector (v,r,density,...), all you need to write is a routine that provides a derivative of that vector!

- Kinematic Lagrangian
- Isotropic diffusion
- Adaptation length diffusion
- Beaching
- Windage
- Stokes drift
- Buoyancy
- Linear degradation

Available functionalities

Pre-processor

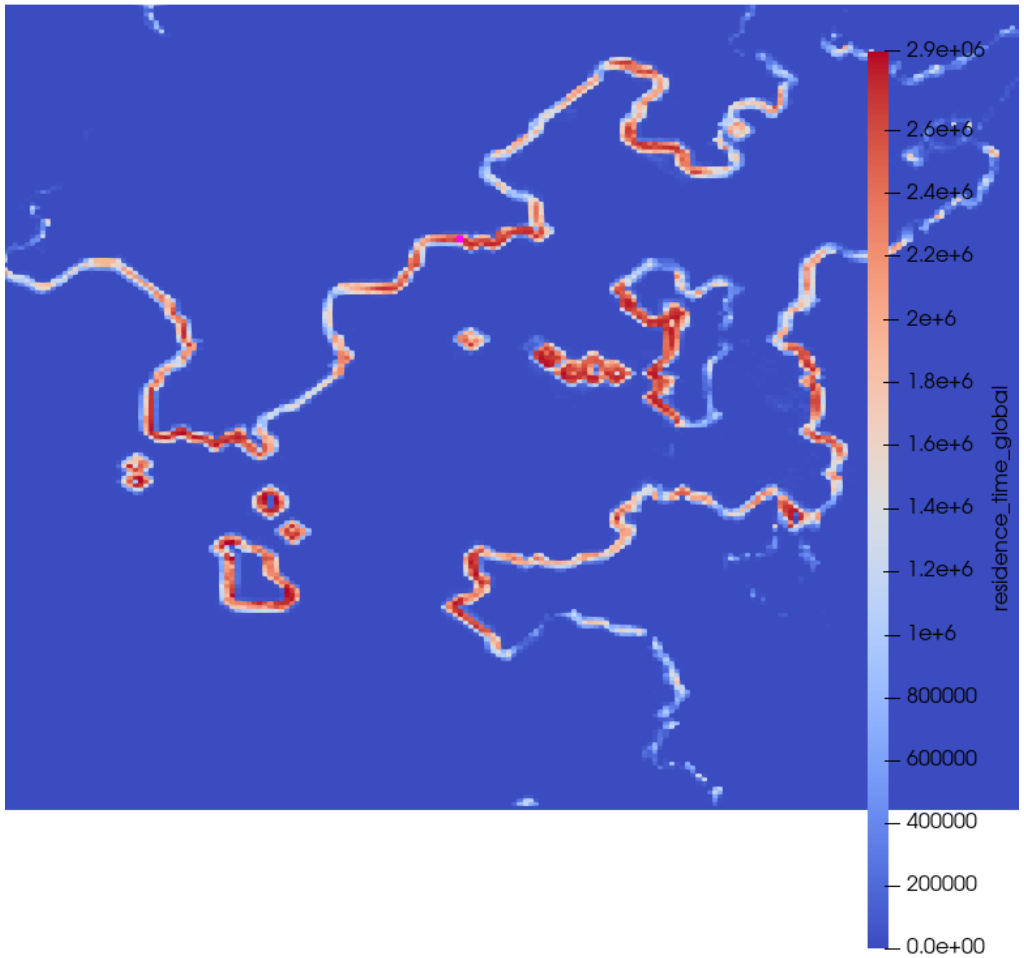
- Indexes collections of input (no more glues, please)
- files by groups
- Supports hydrodynamic,
- waves, winds and water
- property data
- Trivial to extend and maintain

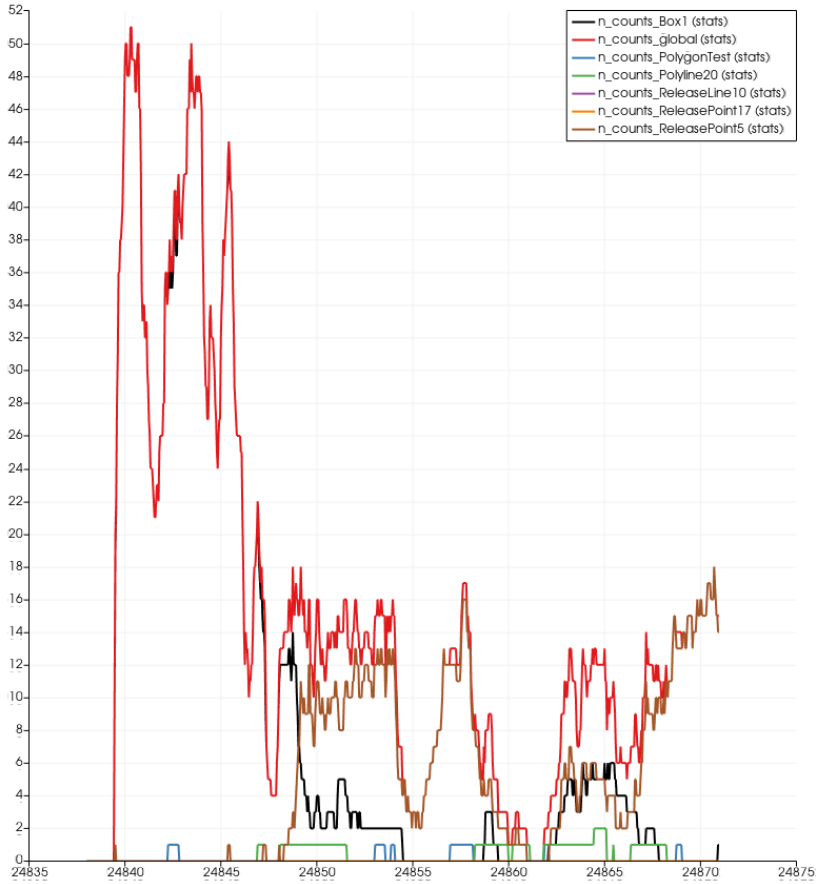
Post-processor

- Recipe based - one simulation, endless outputs
- Produces consumption ready NetCDF outputs
- Can section a solution in space and time
- Several sampling mesh options
- Integrates any available output variable
- Can compute new variables, such as accumulated concentrations and residence times
- Easy to tack on file converters for other formats - hdf5 already implemented
- Easy to extend and maintain

If we put everything in place

If we put everything in place





Modelling a mixed ocean, coastal and estuarine environment

Data downloaded from operational Tagus model, converted to NetCDF and used directly (didn’t even change directory structure, several daily files in several directories)

The code

- Mixed Fortran (core) and Python (pre and post processors)
- Standard OOP structure
- Auto documentation for developers (doxygen)
- Scripts for everything – preparing Windows Visual Studio solutions, Linux makefiles, compile external libraries and model, run the model, post-process the results
- Two manuals and 6 fully contained and click-run examples with all the functionalities

https://github.com/RBCanelas/MOHID-Lagrangian

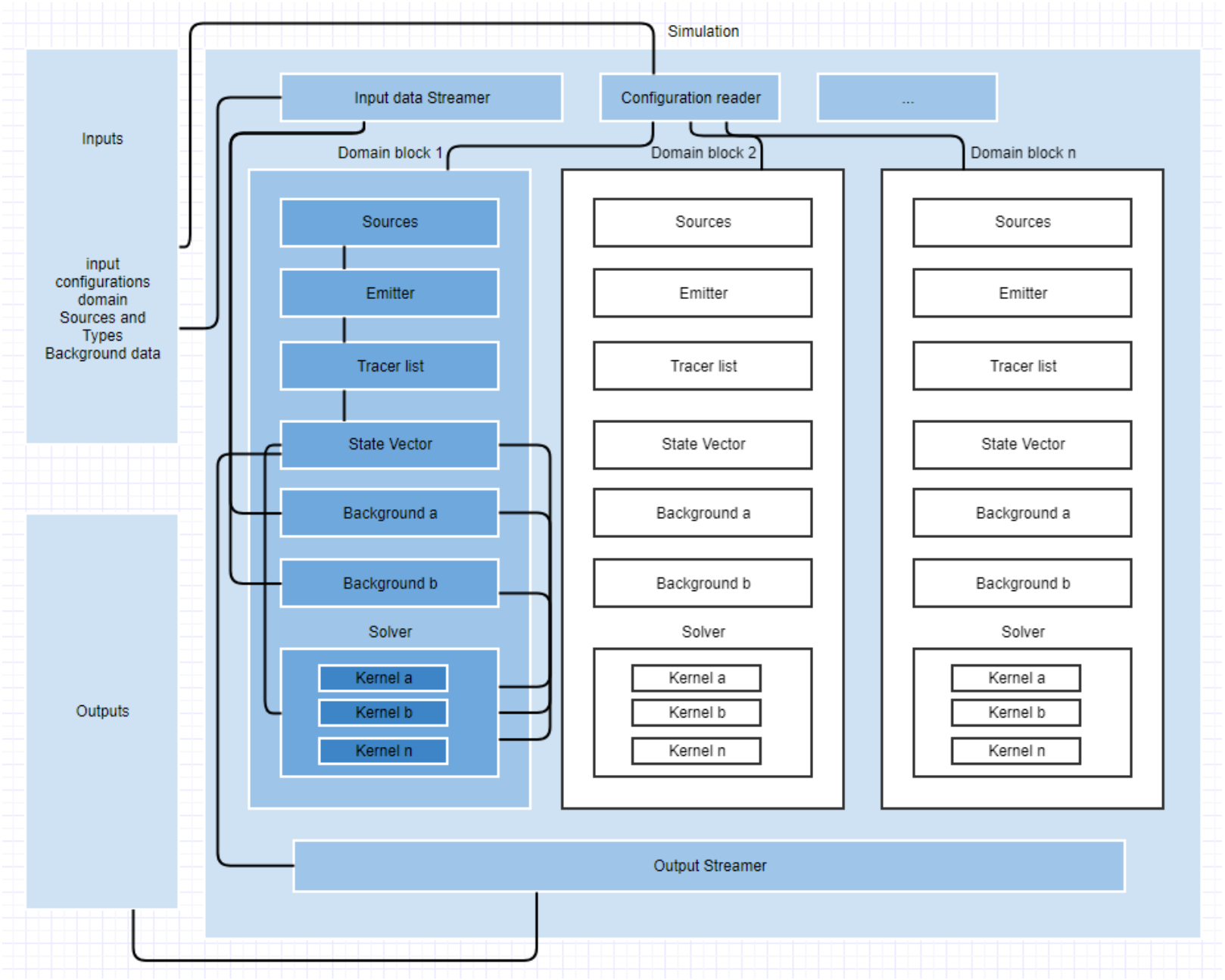
The code

- Fortran 2008+
  - Tested in intel and gnu compilers
- Python 3+

- Tested and tried in windows, linux and mac machines

- To compile you need
  - Cmake, VS in windows, m4 & autotools in Linux
  - Modern Fortran compiler - IFort 18+, GFortran 8+

The code – modern, no shame FORTRAN



Tomorrow – hackathon

- 1hr of practical session
- 1hr of code review

- Proposals
  - Import MOHID Land fields (runoff and porous media)
  - Import a variable beaching probability map
  - degradation parametrization based on object specific surface
  - wind drag depending on object exposed surface
  - Smagorinsky diffusion kernel
  - oil kernels
  - tracer type library brainstorm

- Needed software
  - Cmake
  - VS
  - Notepad++
  - Ifort or gFortran
  - Python 3+ with xarray, h5py, netcdf 4 & vtk
  - Paraview
  - MOHID Lagrangian dev branch clone

Thank you!