

---

**ERSEM**

***Release 2021***

**The PML Modelling Group**

**Nov 07, 2022**



**CONTENTS:**

<b>1</b>	<b>Support</b>	<b>3</b>
<b>2</b>	<b>How to cite</b>	<b>5</b>
2.1	The ERSEM Model . . . . .	5
2.2	ERSEM tutorials . . . . .	9
2.3	Tips and tricks/troubleshooting . . . . .	53
2.4	Developers: installation from source . . . . .	55
2.5	ERSEM Workflow . . . . .	57
2.6	ERSEM module index . . . . .	59
2.7	License . . . . .	73
2.8	Acknowledgements . . . . .	73
2.9	Bibliography . . . . .	73
	<b>Bibliography</b>	<b>75</b>



ERSEM is a marine biogeochemical and ecosystem model. It describes the cycling of carbon, nitrogen, phosphorus, silicon, oxygen and iron through the lower trophic level pelagic and benthic ecosystems.

- **Installation Instructions:** conda: <https://ersem.readthedocs.io/en/latest/tutorials/index.html#conda-installation>, source: <https://ersem.readthedocs.io/en/latest/developers/index.html>
- **User Documentation & Example Usage:** <https://ersem.readthedocs.io/en/latest/>
- **Automated Tests:** Run via GitHub Actions <https://github.com/pmlmodelling/ersem/actions>
- **Acknowledgements:** <https://ersem.readthedocs.io/en/latest/acknowledgements.html>
- **License:** <https://ersem.readthedocs.io/en/latest/license.html>



**SUPPORT**

We strongly encourage everyone using the ERSEM code to register as a user by filling a [short registration form](#). We'd love to get a better understanding of who is using the ERSEM code and for what applications, as it will help us to provide the best support to the user community. It will also allow you to receive information and news on the latest model developments.





## HOW TO CITE

To refer to ERSEM in publications, please cite:

Butenschön, M., Clark, J., Aldridge, J.N., Allen, J.I., Artioli, Y., Blackford, J., Bruggeman, J., Cazenave, P., Ciavatta, S., Kay, S., Lessin, G., van Leeuwen, S., van der Molen, J., de Mora, L., Polimene, L., Sailley, S., Stephens, N., Torres, R. (2016). ERSEM 15.06: a generic model for marine biogeochemistry and the ecosystem dynamics of the lower trophic levels. *Geoscientific Model Development*, 9(4), 1293–1339. doi: [10.5194/gmd-9-1293-2016](https://doi.org/10.5194/gmd-9-1293-2016).

To refer specifically to the ERSEM source code, you may use its Zenodo DOI:

## 2.1 The ERSEM Model

Here we outline the features of the ERSEM model.

### 2.1.1 What is ERSEM?

#### Background

PML is a charity, undertaking innovative, cutting-edge marine research, allowing society to benefit from clean, productive, biologically diverse seas, now and for future generations.

#### Ecosystem

The ecosystem in ERSEM is divided into functional types, which are further subdivided by traits such as size. In the pelagic, ERSEM by default distinguishes:

- 4 types of phytoplankton: diatoms, picophytoplankton, nanophytoplankton, microphytoplankton
- 3 types of zooplankton: nanoflagellates, microzooplankton, mesozooplankton
- bacteria

The benthic system includes:

- 3 types of infauna: meiofauna, suspension feeders, deposit feeders
- 2 types of bacteria: aerobic and anaerobic

In addition, ERSEM tracks the concentrations of phosphate, nitrate, ammonium, silicate, iron, oxygen, dissolved inorganic carbon and alkalinity in the pelagic and in sediment porewaters. It includes several classes of particulate and dissolved organic matter in pelagic and sediment. A carbonate system module calculates pH and calcium carbonate saturation.

FABM also makes it easy to customise the default set of functional types described above, and to combine ERSEM with modules representing other parts of the ecosystem, including fish communities, shellfish, seagrass meadows and spectrally resolved irradiance.

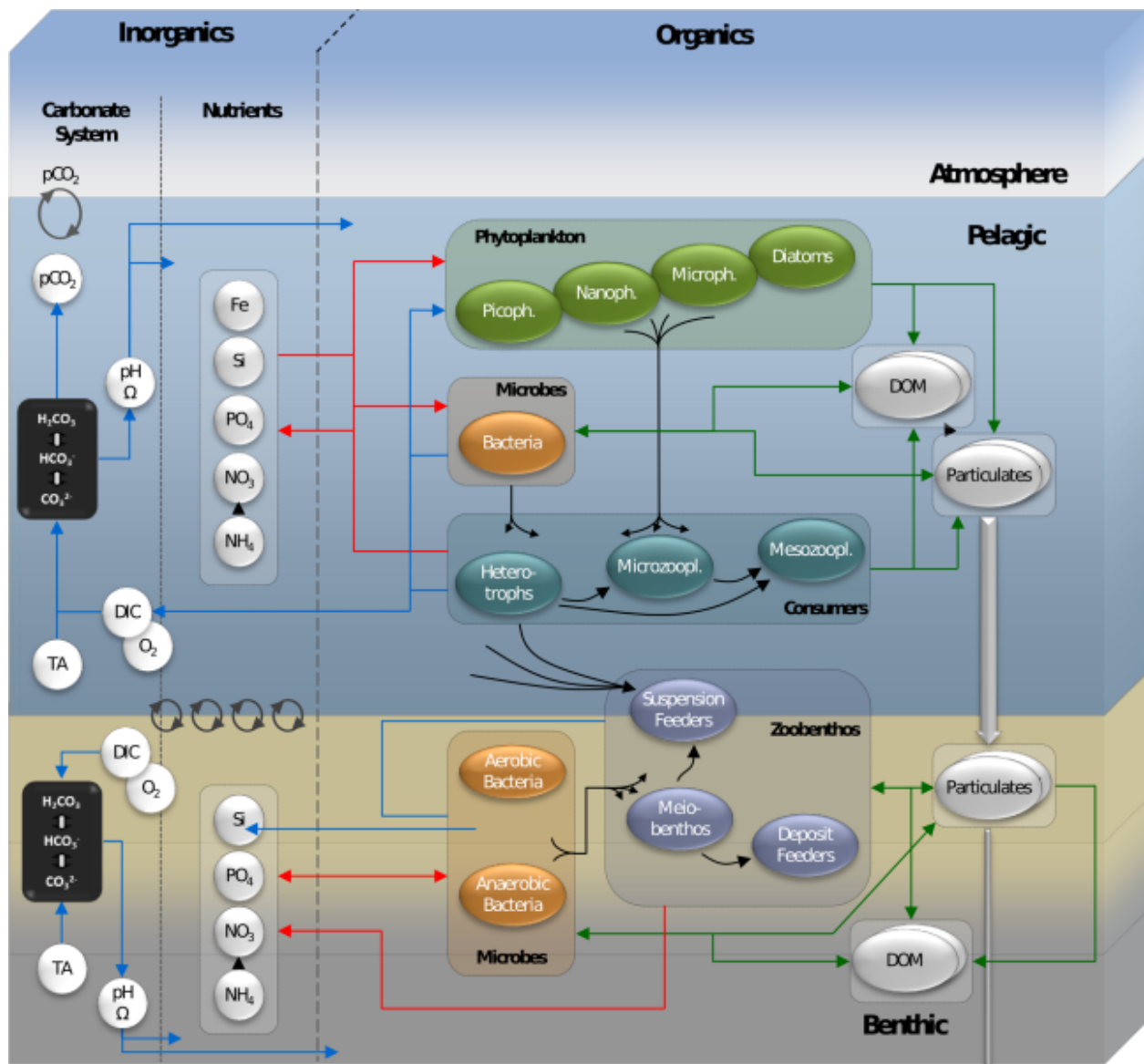


Fig. 1: ERSEM schematic

### 2.1.2 The History of ERSEM

The European Regional Seas Ecosystem Model, ERSEM, was initially developed in two projects, funded by the EU Marine Science and Technology programme (ERSEM, 1990-1993; ERSEM II, 1993-1996) both under the leadership of Job Baretta at NIOZ (NL). In addition to NIOZ, these initial projects involved Plymouth Marine Laboratory (UK), the Universities of Oldenburg (DE), Hamburg (DE), Strathclyde (UK) and Aberdeen (UK), Marine Laboratory Aberdeen, Ecological Modelling Centre (DK) and CEAB (ES).

The earlier ambitions of the program were significant, not only coupling benthic and pelagic processes in a physical context and developing a prognostic model, but extending up the food chain to age-structured mesozooplankton models, pelagic and demersal fish and even seabirds. As a general rule, the higher the trophic level the less successful the models as 20+ years ago understanding, data and computational power were far inferior to that which we currently enjoy.

Although ERSEM itself had roots in earlier modelling initiatives (the Ems-Dollard and Gembase models) it pioneered a community approach to model development. At the heart of this was a set of scripts (SESAME) which controlled compilation and output generation, data transfer by COMMON BLOCK, a system of code nomenclature ([1], if you are wondering why variables and parameters have strange names) and a structure of almost independent modules. This meant that for the majority of developers, with minimal practice, the code was accessible with limited computational expertise. Development of modules could occur in parallel without worrying about interfaces between functional groups. This accessibility did not lead to the most efficient coding but did allow a range of computationally naive experts the unfettered ability to develop the model. A six monthly development cycle, whereby new modules were submitted for assessment and incorporation ahead of a project meeting to evaluate results ensured rapid model development. Computational An early schematic of ERSEM incompatibility was avoided by providing each partner with the same UNIX computer system, the then eye-wateringly expensive SUN SPARC desktops retailing at £20K apiece. Just a few years previously, before the internet, email and processing capability exceeding 100 MHz ERSEM would have been inconceivable.

Each of the initial ERSEM projects produced a special issue describing model systems and results, the Netherlands Journal of Sea Research V33 (3/4) in 1995 and the by then renamed Journal of Sea Research V38 (3/4) in 1997. After cessation of the ERSEM projects model development occurred more independently in three main centres, NIOZ and PML producing variants using the ERSEM name and at Bologna where ERSEM formed the basis of the Biogeochemical Flux Model (BFM). We suspect that in excess of 200 papers have been published using ERSEM since 1995.

The ERSEM model now aspires to reach beyond its name, addressing biogeochemical and ecological systems in many applications in global regional seas and more recently the global ocean, engaging in a range of heuristic, predictive and impact studies.

### 2.1.3 ERSEM model domains

ERSEM is used in a variety of geographical regions, coupled to different physical (hydrodynamic) models. The table below outlines several of these setups.

Area	Spatial Scale		Quality (data used)
	Domain	Res.(km)	
Water column (GOTM-ERSEM)	Various locations	N/A	4,6,8,14,16,22
Global (NEMO)	Global	111	8,11,19,20,21
North Atlantic (NEMO)	20N-80N	28	8,11,19,20,21
North Atlantic (NEMO shelf)	20N-80N	7	8,11,19,20,21
Irish and Celtic Seas (FVCOM)	41.8-56.8N, 9.6-2.5W	3.5	7,23,24,25
NW European shelf (NEMO)	40N-65N, 20W-13E	7	3,8,11,12,13,14,19
NW European shelf (POLCOMS)	40N-65N, 20W-13E	10	3,8,9,10,11,12,13,14,15,16,17,19
NW European shelf (POLCOMS + data assimilation)	40N-65N, 20W-13E	10	2,3,8,9,10,12,13,14,15,17
AMM7NW European-Baltic-GCOMS	46.4-63N, 17.5W-13E	10	3,8,11,14,17,18
Western European Channel (WEC)	48.5-50.9N, 7.6-1.25W	1.9	2,4,8,12,13,14
WEC (data assimilation)	48.5-50.9N, 7.6-1.25W	7	1,2,4,8,12,13,14

## Key

- 1. GlobColour ocean colour chlorophyll
- 2. ESA CCI ocean colour
- 3. North Sea Project cruises
- 4. Western Channel Observatory
- 5. Tide gauges
- 6. Smart Buoy data
- 7. NCEP reanalysis
- 8. World Ocean Atlas
- 9. ICES temperature and salinity
- 10. ICES nutrients and chlorophyll
- 11. IPCC climate forcing (HADGEM, IPSL, ECHAM)
- 12. EA riverine nutrients data
- 13. European river data
- 14. ECMWF reanalysis meteorology
- 15. DMI reanalysis meteorology
- 16. ESSC ocean reanalysis
- 17. GLORYS reanalysis
- 18. GLOBAL NEWS river data
- 19. GLODAP
- 20. DFS atmospheric reanalysis
- 21. GEMS-GLORI river data
- 22. BATS data
- 23. HF Radar and CTD Data
- 24. NTSLF data
- 25. CEH river discharge and temperature

**Note:** The table of regional setups is not yet complete, please stay tuned for updates.

## Existing uses:

- **Natural resources:** understanding biogeochemical cycles, biodiversity and valuation of ecosystem services.
- **Resilience to environmental hazards:** eutrophication, microplastics, fishing pressure, invasive species and harmful algal blooms. Impacts of offshore renewable energy and ecological risks of carbon capture and storage.
- **Environmental change:** climate change impacts, ocean acidification, multiple stressors, and sustainable fisheries.
- Run operationally by the UK Met Office to predict water quality.
- Estimation of the carbon and nutrient budgets of the UK shelf.

**Potential new uses:**

- Understanding shelf seas carbon (“blue carbon”) and nutrient budgets (past and future climate).
- Expansion to represent biodiversity-relevant processes over a range of spatial and temporal scales, and simulate changes in function in the context of ecosystem services.
- Implementation and testing scalable models of differing complexity.

**Key modelling issues:**

- Setting inputs (parameterisation) and testing outputs against real data (calibration) is an essential, but resource-intensive and on-going process to ensure quality and improve predictions. Understanding the impact of changing inputs on the outputs from the models (sensitivity) and the effect of uncertainty in model parameters on robustness of model predictions.
- Challenge to assess model capability with respect to seasonal variability, long-term changes, regime shift and tipping points due to limitations of the data available.
- Complexity of model leads to a need for significant interpretation and explanation for stakeholders.
- Potential mismatch between scales of model output and data sets.
- Significant expertise needed to operate system and high performance parallel computing facility required for three-dimensional full scale simulations.

## 2.2 ERSEM tutorials

ERSEM is mainly run with 3D hydrodynamic models such as [NEMO](#) and [FVCOM](#). However, due to its flexibility, it can also be run as a 0D box model or within a 1D water column, via [GOTM](#). Parts of the library can also be called directly from Python through the `pyfabm` interface. Here we describe several tutorials for the common use cases for ERSEM.

### 2.2.1 Python front-end (pyfabm)

FABM comes with a `pyfabm` package for the python programming language. This package enables you to access [FABM's](#) biogeochemical models directly from python. For instance, you can enumerate a model's parameters and variables, or obtain temporal derivatives and diagnostics for any given environment and model state. In combination with a python-based time integration scheme (e.g., `scipy.integrate.odeint`), this also allows you to perform model simulations. More information about `pyfabm` can be found on the [FABM wiki page](#). Below you can find brief instructions on how to use `pyfabm` with ERSEM.

#### Running pyfabm tutorial

To demonstrate how `pyfabm` can be used with ERSEM, we will calculate the oxygen saturation concentration as a function of temperature and salinity using the method implemented in ERSEM, which is taken from [3].

---

**Note:** The following script requires `matplotlib` to be installed. This can easily be done via `pip` in the following way:

```
python -m pip install matplotlib
```

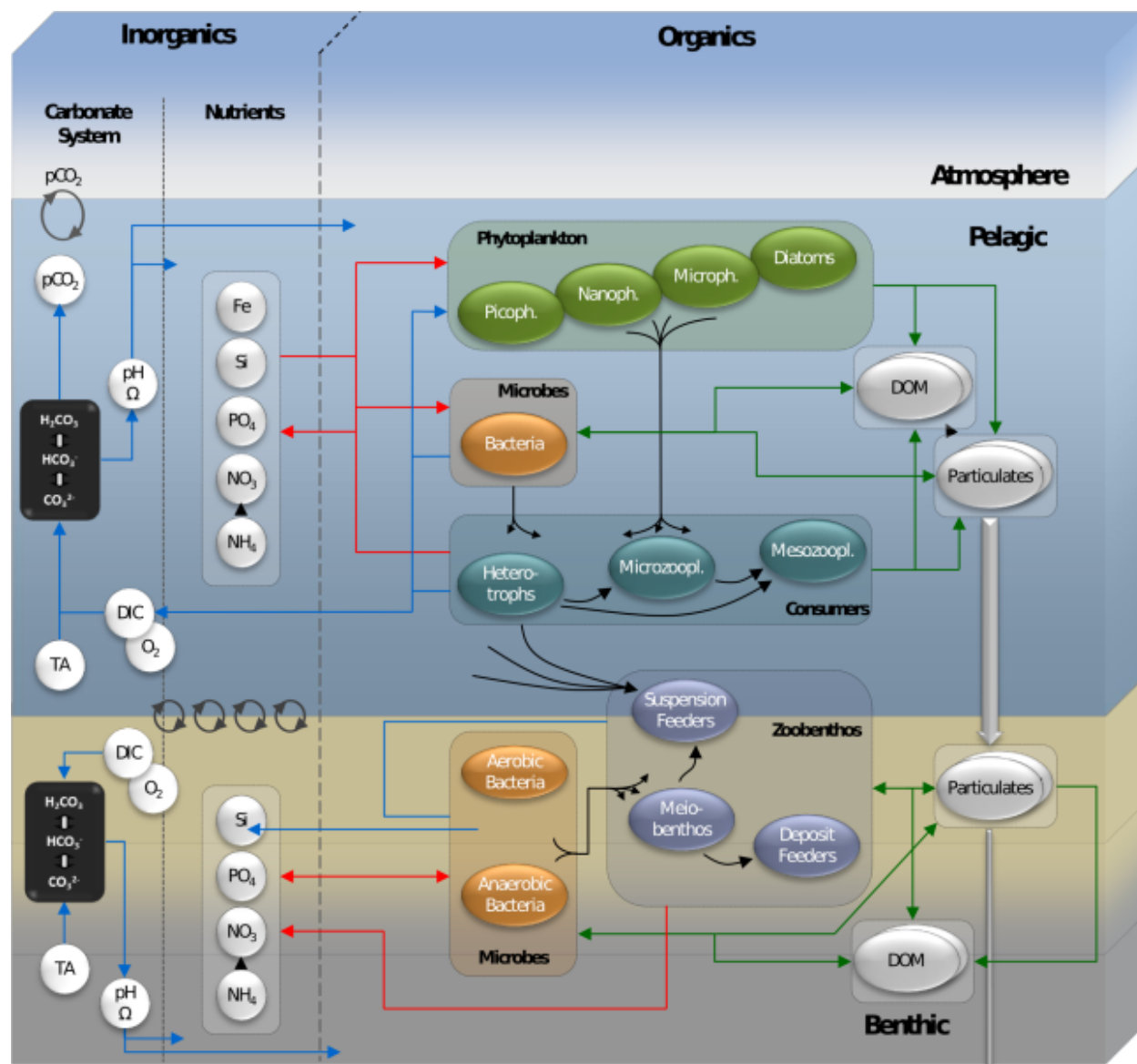


Fig. 2: ERSEM schematic

With pyfabm installed, this can be achieved using the following code:

```

1 from matplotlib import cm
2 import matplotlib.pyplot as plt
3 import numpy as np
4 import os
5 import pyfabm
6
7 def main():
8     """
9     Run pyfabm-ERSEM tutorial
10
11     :param model_path: Full path to netCDF model output
12     :type model_path: str
13
14     :return: list of oxygen saturation concentrations
15     :rtype: list
16     """
17     dir_path = os.path.dirname(os.path.realpath(__file__))
18     ersem_dir = os.path.dirname(os.path.dirname(dir_path))
19
20     # Path to ERSEM yaml file
21     ersem_yaml_file = os.path.join(ersem_dir,
22                                     'testcases',
23                                     'fabm-ersem-15.06-L4-noben-docdyn-iop.yaml')
24
25     if not os.path.isfile(ersem_yaml_file):
26         raise RuntimeError("Could not find Ersem yaml file with the "
27                             "{}".format(ersem_yaml_file))
28
29     # Create model
30     model = pyfabm.Model(ersem_yaml_file)
31
32     # Configure the environment
33     model.findDependency('longitude').value = -4.15
34     model.findDependency('latitude').value = 50.25
35     model.findDependency('number_of_days_since_start_of_the_year').value = 0.
36     model.findDependency('temperature').value = 10.
37     model.findDependency('wind_speed').value = 1.
38     model.findDependency('surface_downwelling_shortwave_flux').value = 50.
39     model.findDependency('practical_salinity').value = 35.
40     model.findDependency('pressure').value = 10.
41     model.findDependency('density').value = 1035.
42     model.findDependency('mole_fraction_of_carbon_dioxide_in_air').value = 280.
43     model.findDependency('absorption_of_silt').value = 0.07
44     model.findDependency('bottom_stress').value = 0.
45     model.findDependency('cell_thickness').value = 1.
46     model.setCellThickness(1)
47
48     # Verify the model is ready to be used
49     assert model.checkReady(), 'One or more model dependencies have not been fulfilled.'
50

```

(continues on next page)

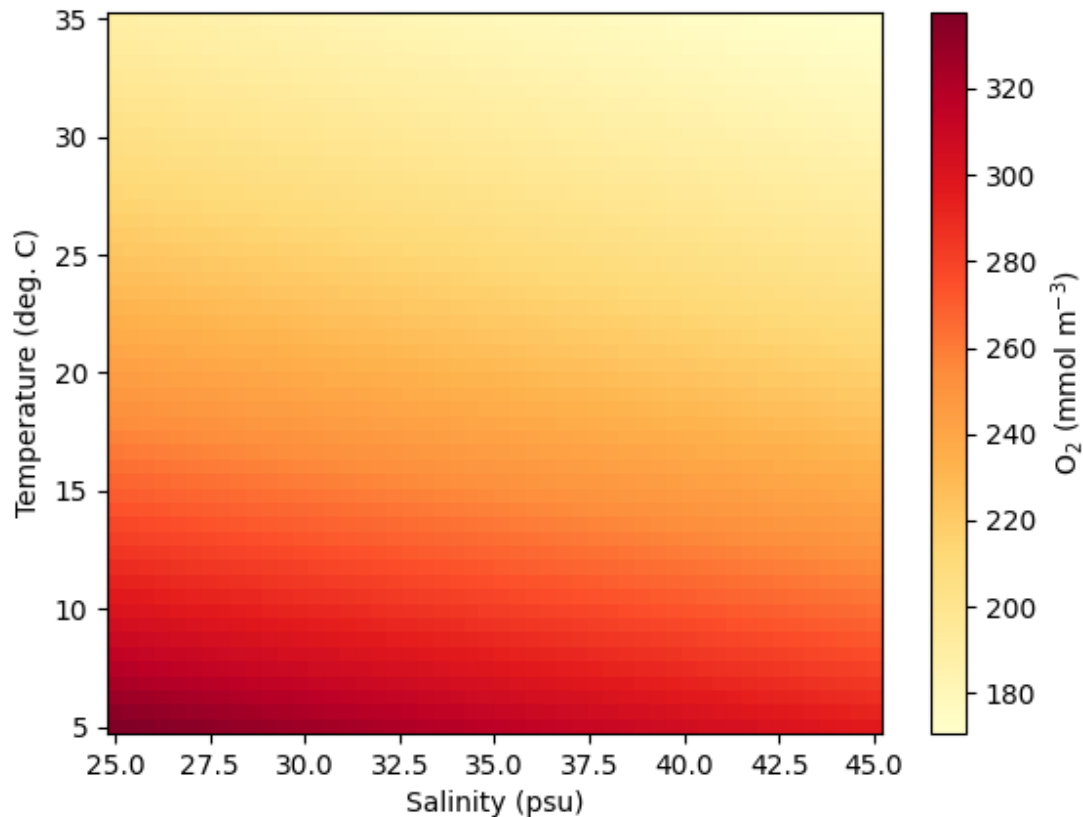
(continued from previous page)

```

51  # Define ranges over which temperature and salinity will be varied
52  n_points = 50
53  temperature_array = np.linspace(5, 35, n_points)
54  salinity_array = np.linspace(25, 45, n_points)
55
56  # Create an array in which to store oxygen_saturation_concentrations
57  oxygen_saturation_concentration = np.empty((n_points, n_points), dtype=float)
58
59  # Calculate oxygen saturation concentrations using ERSEM
60  for t_idx, t in enumerate(temperature_array):
61      model.findDependency('temperature').value = t
62      for s_idx, s in enumerate(salinity_array):
63          model.findDependency('practical_salinity').value = s
64          _ = model.getRates()
65          oxygen_saturation_concentration[t_idx, s_idx] = \
66              model.findDiagnosticVariable('O2/osat').value
67
68  # Create the figure
69  figure = plt.figure()
70  axes = plt.gca()
71
72  # Set color map
73  cmap = cm.get_cmap('YlOrRd')
74
75  # Plot
76  plot = axes.pcolormesh(salinity_array,
77                        temperature_array,
78                        oxygen_saturation_concentration,
79                        shading='auto',
80                        cmap=cmap)
81
82  axes.set_xlabel('Salinity (psu)')
83  axes.set_ylabel('Temperature (deg. C)')
84
85
86
87  # Add colour bar
88  cbar = figure.colorbar(plot)
89  cbar.set_label('O2 (mmol m-3)')
90
91  plt.show()
92
93  return oxygen_saturation_concentration
94
95  if __name__ == "__main__":
96      main()

```





In additions, example Jupyter notebooks that use the Python front-end can be found in <FABMDIR>/testcases/python and more examples can be found on [the FABM wiki](#).

## 2.2.2 fabm0d: ERSEM in an aquarium

FABM's 0d driver allows you to run biogeochemical models in a “well-mixed box” under arbitrary (time-varying) environmental forcing.

### Running FABM0d tutorial

To run the model, you must first obtain or generate a set of input or forcing files. Setups for the L4 site are stored under version control and can be accessed [here](#).

To run the tutorial, you will first need to clone the configuration files and then run *FABM0d*. This is done with the following:

```

1  #!/bin/bash
2
3  echo "Cloning config repo"
4  git clone https://github.com/pmlmodelling/ersem-setups.git
5
6  cd ersem-setups/0d-aquarium

```

(continues on next page)

(continued from previous page)

```

7 echo "Running FABM with repo configuration"
8 ~/local/fabm/0d/bin/fabm0d -y ../L4/fabm.yaml
9

```

**Note:** The following script requires matplotlib, netCDF and numpy to be installed. This can easily be done via pip in the following way:

```
python -m pip install matplotlib numpy netCDF
```

To visualise the results, we again use a python script. Running the script you will need to add a commandline argument `--model-path` which is the path to the output from the FABM0d run.

```

1 import argparse
2 import matplotlib.pyplot as plt
3 from matplotlib import ticker
4 import numpy as np
5 import datetime
6 import re
7 import sys
8
9
10 try:
11     import netCDF4 as nc
12 except ImportError:
13     print("Please install the Python interface to netCDF")
14     sys.exit()
15
16
17 def main(model_path):
18     """
19     Run FABM0D-ERSEM tutorial
20
21     :param model_path: Full path to netCDF model output
22     :type model_path: str
23
24     :return: Dictionary containing output model variables for systests
25     :rtype: dict
26     """
27     # Dictionary used to save input and output variables for testing
28     test_vars = {}
29     def plot_var(axes, x, y, label):
30         # sets the maximum number of labels on the x axis to be 10
31         xticks = ticker.MaxNLocator(10)
32
33         var = data.variables[label]
34         y = var[:].squeeze()
35
36         long_label = '{} ({}).format(var.long_name, var.units)
37         y_label = re.sub(r'(\s\S*?)\s', r'\1\n', long_label)
38

```

(continues on next page)

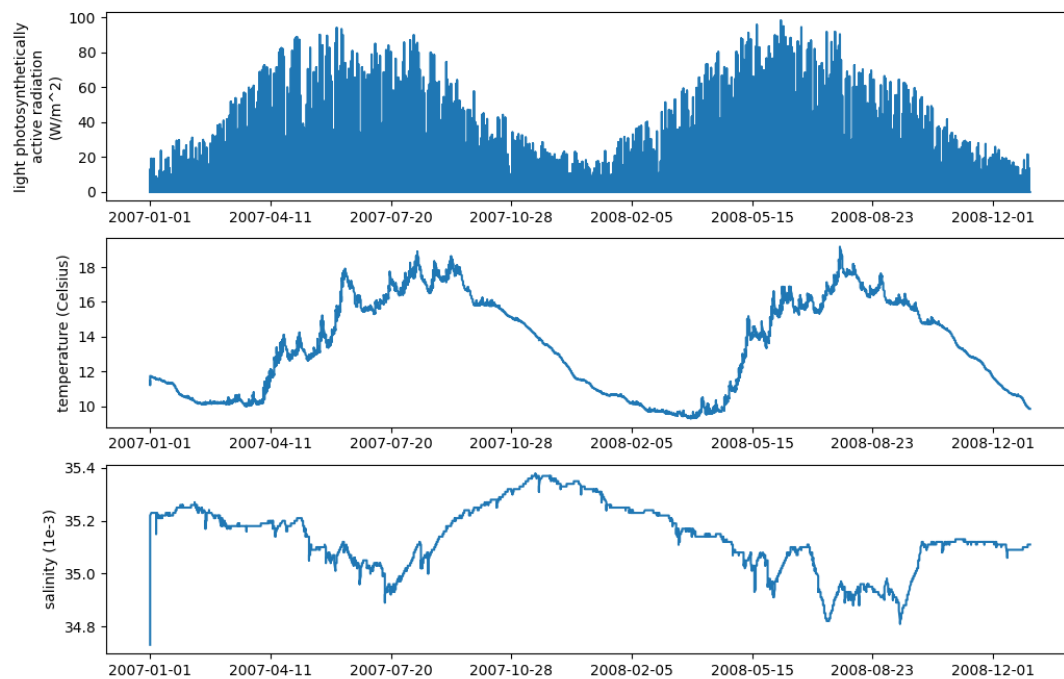
(continued from previous page)

```

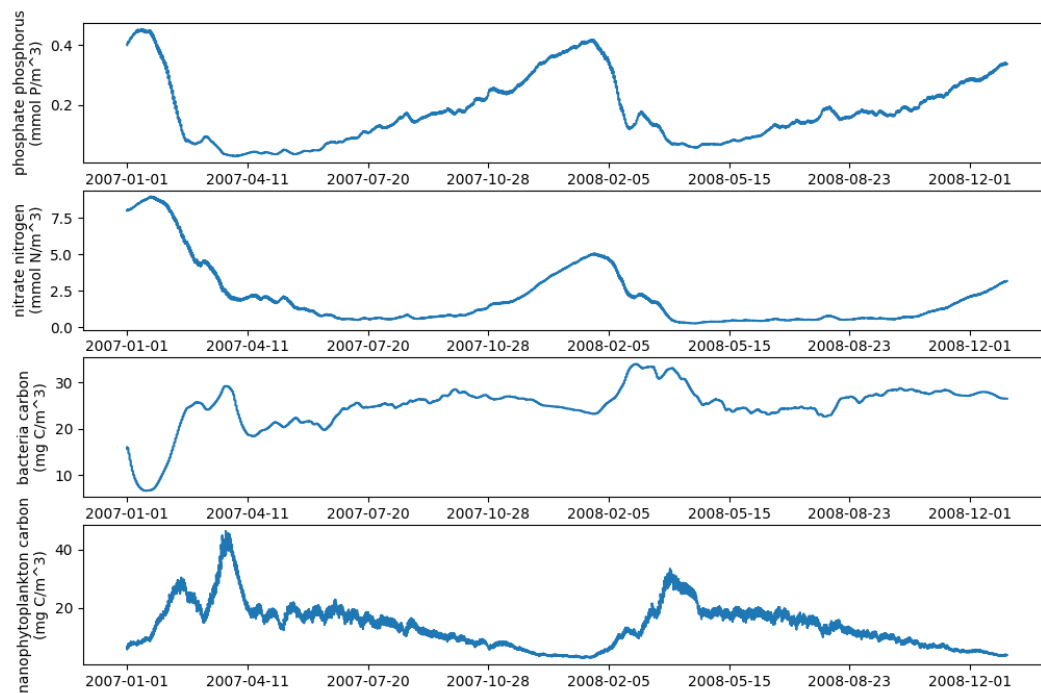
39     axes.plot(x, y)
40     axes.set_ylabel(y_label)
41     axes.xaxis.set_major_locator(xticks)
42     test_vars[label] = y
43
44
45     data = nc.Dataset(model_path, 'r')
46
47     times = data.variables['time']
48     dates = nc.num2date(times[:,
49                         units=times.units,
50                         calendar=times.calendar)
51     dates = [str(d).split(" ")[0] for d in dates]
52     test_vars["dates"] = dates
53
54     # Plotting input data
55     input_data = ["light_parEIR", "temp", "salt"]
56     fig, ax_arr = plt.subplots(len(input_data), 1)
57     DPI = fig.get_dpi()
58     fig.set_size_inches(1200.0/float(DPI), 800.0/float(DPI))
59     for ax, label in zip(ax_arr, input_data):
60         plot_var(ax, dates, data, label)
61     fig.suptitle("Input data")
62
63     # Plotting output data
64     output_data = ["N1_p", "N3_n", "B1_c", "P2_c"]
65     fig, ax_arr = plt.subplots(len(input_data), 1)
66     DPI = fig.get_dpi()
67     fig.set_size_inches(1200.0/float(DPI), 800.0/float(DPI))
68     for ax, label in zip(ax_arr, output_data):
69         plot_var(ax, dates, data, label)
70     fig.suptitle("Output data")
71     plt.show()
72
73     return test_vars
74
75 if __name__ == "__main__":
76
77     parser = argparse.ArgumentParser()
78     parser.add_argument('-p', '--model-path', type=str, required=True,
79                         help='Path to FABMOD-ERSEM output')
80     args, _ = parser.parse_known_args()
81     model_path = args.model_path
82
83     main(model_path)

```

Input data



Output data



### 2.2.3 GOTM: ERSEM in a water column

To run the model, you must first obtain or generate a set of input or forcing files. Setups for the L4 and BATS sites which were used in [2] are stored under version control and can be accessed here. Forcing data for the SSB sampling sites can be accessed here. The following example uses the L4 setup described in [2].

#### Running GOTM-FABM-ERSEM

The L4 sampling site is one part of the [Western Channel Observatory \(WCO\)](#), and is located at (50°15.0'N; 4°13.0'W).

To run the tutorial, you will first need to clone the configuration files and then run *gotm*. This is done with the following:

```

1  #!/bin/bash
2
3  echo "Cloning config repo"
4  git clone https://github.com/pmlmodelling/ersem-setups.git
5
6  cd ersem-setups/L4
7
8  echo "Running GOTM with repo configuration"
9  ~/local/gotm/bin/gotm

```

**Note:** The following script requires `matplotlib`, `netCDF4` and `numpy` to be installed. This can easily be done via `pip` in the following way:

```
python -m pip install matplotlib numpy netCDF4
```

To visualise the results, we again use a python script. Running the script you will need to add a commandline argument `--model-path` which is the path to the output from the GOTM-FABM-ERSEM run.

```

1  import argparse
2  import matplotlib.pyplot as plt
3  from matplotlib import ticker
4  import numpy as np
5  import os
6  import sys
7  import re
8
9  try:
10     import netCDF4 as nc
11 except ImportError:
12     print("Please install the Python interface to netCDF")
13     sys.exit()
14
15
16 def main(model_path):
17     """
18     Run GOTM-ERSEM tutorial
19
20     :param model_path: Full path to netCDF model output
21     :type model_path: str
22

```

(continues on next page)

(continued from previous page)

```

23 :return: Dictionary containing output model variables for systests
24 :rtype: dict
25 """
26
27 # Dictionary used to save output variables for testing
28 output_vars = {}
29 def plot_time_series(axes, data, depth, model_var_name):
30     # sets the maximum number of labels on the x axis to be 10
31     xticks = ticker.MaxNLocator(10)
32
33     times = data.variables['time']
34     dates = nc.num2date(times[:],
35                         units=times.units,
36                         calendar=times.calendar)
37
38     z = data.variables['z'][:].squeeze()
39     zi = data.variables['zi'][:].squeeze()
40     var = data.variables[model_var_name]
41
42     long_label = '{} {}'.format(var.long_name, var.units)
43     y_label = re.sub(r'(\s\S*?)\s', r'\1\n', long_label)
44
45     # Interpolate variable data to the given depth below the moving free surface
46     var_time_series = []
47     for i in range(var.shape[0]):
48         # Remove offset introduced by the moving free surface
49         depth_offset = depth + zi[i, -1]
50
51         var_time_series.append(np.interp(depth_offset, z[i, :], var[i, :].squeeze()))
52
53     dates = [str(d).split(" ")[0] for d in dates]
54     axes.plot(dates, var_time_series)
55     axes.set_ylabel(y_label)
56     axes.xaxis.set_major_locator(xticks)
57     output_vars[model_var_name] = var_time_series
58     output_vars['dates'] = dates
59
60     data = nc.Dataset(model_path, 'r')
61     data.set_auto_maskandscale(True)
62
63     # Depth at which to compare the model and data, given as relative to the moving free_
64     ↪ surface.
65     depth = 0.0
66
67     fig, ax_arr = plt.subplots(3,1)
68     DPI = fig.get_dpi()
69     fig.set_size_inches(1200.0/float(DPI), 800.0/float(DPI))
70
71     names = ['N1_p', 'N3_n', 'N5_s']
72     for ax, name in zip(ax_arr, names):
73         plot_time_series(ax, data, depth, name)

```

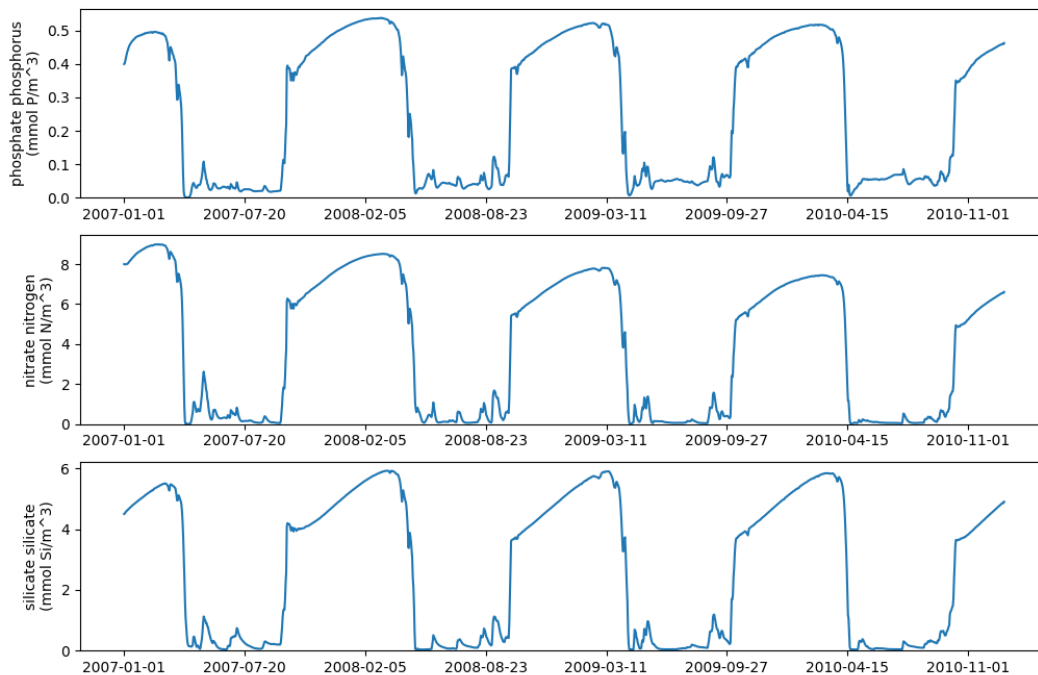
(continues on next page)

(continued from previous page)

```

74     ax.tick_params(axis='both', which='major')
75     ax.tick_params(axis='both', which='minor')
76     ax.set_ylim(0)
77
78     plt.show()
79
80     return output_vars
81
82 if __name__ == "__main__":
83     parser = argparse.ArgumentParser()
84     parser.add_argument('-p', '--model-path', type=str, required=True,
85                         help='Path to GOTM-FABM-ERSEM output')
86     args, _ = parser.parse_known_args()
87     model_path = args.model_path
88
89     main(model_path)

```



## 2.2.4 FVCOM: ideal estuary model

This tutorial gives an end to end example how to install and then use FVCOM-ERSEM with an ideal estuary model on a high performance computing (HPC) machine. We have used PML's in house machine, [CETO](#). Unlike the other tutorials in this section we go through setting up FVCOM-ERSEM on the HPC machine and then running and plotting the results.

This tutorial is based on the scripts found in the [ERSEM's setups repository](#). The individual scripts can be found in the *ideal\_estuary* folder.

---

**Note:** You will need to get access to the [UK-FVCOM GitLab repo](#).

---

### Building and running FVCOM-ERSEM

The three key packages required to run this tutorial are:

- [FVCOM](#) (UK version – see note)
- [FABM](#)
- [ERSEM](#)

Both ERSEM and FABM are freely available on GitHub, however, for UK-FVCOM you will have to register for the code – see note above.

An example build script is as follows:

```
1  #!/usr/bin/env bash
2
3  module load intel-mpi/5.1.2 netcdf-intelmpi/default intel/intel-2016 hdf5-intelmpi/
4  ↪default
5
6  SCRIPT_DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" &> /dev/null && pwd )"
7  CODE_DIR=$SCRIPT_DIR/code
8  INSTALL_DIR=$SCRIPT_DIR/model
9
10 num_cpu=$(nproc)
11
12 website=("git@github.com:UK-FVCOM-Usergroup/uk-fvcom.git" "git@github.com:pmlmodelling/
13 ↪ersem.git" "git@github.com:fabm-model/fabm.git")
14 name=("uk-fvcom" "ersem" "fabm")
15 branch=("FVCOM-FABM" "master" "master")
16
17 mkdir $CODE_DIR
18 cd $CODE_DIR
19
20 echo "Obtaining source code"
21 for i in 0 1 2
22 do
23   git clone ${website[i]} ${name[i]}
24   cd ${name[i]}
25   git checkout ${branch[i]}
26   cd $CODE_DIR
27 done
```

(continues on next page)



(continued from previous page)

```

26
27 cd $SCRIPT_DIR
28
29 FABM=$CODE_DIR/fabm/src
30 ERSEM=$CODE_DIR/ersem
31 FABM_INSTALL=$INSTALL_DIR/FABM-ERSEM
32 FC=$(which mpiifort)
33
34 mkdir -p $FABM_INSTALL
35
36 cd $FABM
37 mkdir build
38 cd build
39 # Production config:
40 cmake $FABM -DFABM_HOST=fvcom -DFABM_ERSEM_BASE=$ERSEM -DCMAKE_Fortran_COMPILER=$FC -
41 ↪DCMAKE_INSTALL_PREFIX=$FABM_INSTALL
42 make install -j $num_cpu
43
44 cd $SCRIPT_DIR
45
46 sed -i 's|BASE_SETUP_DIR|"$SCRIPT_DIR"|g' make_ideal_estuary.inc
47 ln -s $SCRIPT_DIR/make_ideal_estuary.inc $SCRIPT_DIR/code/uk-fvcom/FVCOM_source/make.inc
48
49 # Installing FVCOM additional packages (METIS, Proj, etc)
50 cd $SCRIPT_DIR/code/uk-fvcom/FVCOM_source/libs
51 mv makefile makefile_
52 ln -s makefile.CETO makefile
53 make -j $num_cpu
54
55 # Building FVCOM
56 cd ..
57 make -j $num_cpu

```

Here you will have to adapt the script to ensure you are using the right HPC modules and the corresponding compilers. The important compiler is the fortran one which is set with the variable *FC*.

Another key file to change is the `make.inc` file, here again you will need to change the compilers to reflect the modules you are using on the HPC machine. For example, on lines 74 and 75 `IOLIBS` and `IOINCS` are set, these would need to be change to reflect the modules on the HPC machine you are using.

`make.inc` file

```

1  #/=====/
2  #
3  # PML cet0 makeinc for the ideal estuary model
4  #
5  #/=====/
6
7  #=====
8  #
9  #   WELCOME TO FVCOM
10 #
11 #   TO BUILD FVCOM, PLEASE SET THE FOLLOWING IN THIS FILE:

```

(continues on next page)

(continued from previous page)

```

12 #      TOPDIR - the directory this make.inc file is in
13 #      LIBDIR - the directroy where libraries are installed
14 #      INCIDR - the directroy where include files are installed
15 #
16 #      CHOOSE YOUR MODEL OPTIONAS - The Def Flags!
17 #
18 #      CHOOSE A COMPILER FROM THE LIST OR CREATE YOUR OWN
19 #
20 #      You can also use the makefile to build and install some of the libraries
21 #      used by fvcom. Set the INSTALLDIR and choose the 'LOCAL INSTALL' option
22 #      below. Select an non-mpi compiler from the list and run 'make libs_ext'
23 #
24 #      Good Luck!
25 #
26
27 #===== TOPDIR =====
28 # TOPDIR is the directory in which this make file and the fvcom source reside
29
30         TOPDIR      = BASE_SETUP_DIR/code/uk-fvcom/FVCOM_source
31 # TOPDIR must be set!
32
33 #===== INSTALLDIR =====
34 # INSTALLDIR is the directory where you wish to install external libraries
35 # The default is in the $(TOPDIR)/libs/install, but you can choose...
36         INSTALLDIR  = $(TOPDIR)/libs/install
37 #=====
38
39 # PREPROCESSOR OPTIONS FOR CPP
40         DEF_FLAGS    = -P -traditional
41 #=====
42
43 ##### MEDM ENVIRONMENT #####
44 # Use the environmental variables, LIBPATH and INCLUDEPATH, set by MODULE
45 # to find the packages used by this build.
46 colon=:
47 empty=
48 dashI= $(empty) -I
49 dashL= $(empty) -L
50 # ### UNCOMMENT HERE!
51 #         LIBDIR      = -L$(subst $(colon),$(dashL),$(LIBPATH))
52 #         INCIDR      = -I$(subst $(colon),$(dashI),$(INCLUDEPATH))
53
54 #####
55
56 # LOCAL INSTAL
57         LIBDIR        = -L$(INSTALLDIR)/lib
58         INCIDR        = -I$(INSTALLDIR)/include
59
60 #-----
61 # STANDARD LIBRARIES FOR DATA AND TIME IN fvcom:
62 #
63         DTLIBS        = -ljulian

```

(continues on next page)

(continued from previous page)

```

64      DTINCS      =
65  #
66  #-----
67  #      NETCDF OUTPUT      NETCDF IS NOW REQUIRED TO COMPILE FVCOM
68  #                          DUMP OUTPUT INTO NETCDF FILES (yes/no)
69  #                          REQUIRES SYSTEM DEPENDENT NETCDF LIBRARIES
70  #                          COMPILED WITH SAME F90 COMPILER
71  #                          SET PATH TO LIBRARIES WITH IOLIBS
72  #                          SET PATH TO INCLUDE FILES (netcdf.mod) WITH IOINCS
73  #-----
74      IOLIBS      = -L/gpfs1/apps/netcdf/intelmpi/lib -lnetcdff -lnetcdf -lhdf5_
    ↪ hl -lhdf5 -lz -lm
75      IOINCS      = -I/gpfs1/apps/netcdf/intelmpi/include -I/gpfs1/apps/hdf5/
    ↪ intelmpi/include
76  #-----
77
78
79  #=====
80  # BEGIN USER DEFINITION SECTION
81  #=====
82  #      SELECT MODEL OPTIONS
83  #      SELECT FROM THE FOLLOWING OPTIONS BEFORE COMPILING CODE
84  #      SELECT/UNSELECT BY COMMENTING/UNCOMMENTING LINE (#)
85  #      CODE MUST BE CLEANED (with "make clean") AND
86  #      RECOMPILED IF NEW SET OF OPTIONS IS DESIRED
87  #-----
88
89
90  #-----
91  #      PRECISION          DEFAULT PRECISION: SINGLE
92  #                          UNCOMMENT TO SELECT DOUBLE PRECISION
93  #-----
94
95  #      FLAG_1 = -DDOUBLE_PRECISION
96
97  ## SINGLE PRECISION OUTPUT FOR VISIT
98      FLAG_1 = -DDOUBLE_PRECISION -DSINGLE_OUTPUT -DNETCDF4_COMPRESSION
99  #      FLAG_1 = -DSINGLE_OUTPUT -DNETCDF4_COMPRESSION
100
101  #-----
102  #      SPHERICAL          SELECT SPHERICAL COORDINATES FOR INTEGRATION
103  #                          DEFAULT: CARTESIAN
104  #                          UNCOMMENT TO SELECT SPHERICAL COORDINATES
105  #-----
106
107  #      FLAG_2 = -DSPHERICAL
108
109  #-----
110  #      FLOODYING/DRYING   INCLUDE WET/DRY TREATMENT OF DOMAIN
111  #                          CAN BE ACTIVATED/DEACTIVATED AT RUN TIME WITH
112  #                          INPUT FILE CONTROL. (SEE exa_run.dat) FILE
113  #                          DEFAULT: NO FLOODYING/DRYING INCLUDED

```

(continues on next page)

(continued from previous page)

```

114 #                                UNCOMMENT TO INCLUDE FLOODYING/DRYING
115 #-----
116
117 FLAG_3 = -DWET_DRY
118
119 #-----
120 #      MULTI_PROCESSOR      INCLUDES PARALLELIZATION WITH MPI
121 #                           REQUIRES LINKING MPI LIBRARIES OR COMPILING
122 #                           WITH A PRELINKED SCRIPT (mpif90/mpf90/etc)
123 #                           DEFAULT: NO PARALLEL CAPABILITY
124 #                           UNCOMMENT TO INCLUDE MPI PARALLEL CAPABILITY
125 #-----
126
127 FLAG_4 = -DMULTIPROCESSOR
128 PARLIB = -lmetis -L$(LIBDIR)
129
130 #-----
131 #      WATER_QUALITY      INCLUDE EPA WATER QUALITY MOD
132 #                           CAN BE ACTIVATED/DEACTIVATED AT RUN TIME WITH
133 #                           VARIABLE WQM_ON IN INPUT FILE
134 #                           DEFAULT: NO WATER QUALITY MODEL
135 #                           UNCOMMENT TO INCLUDE WATER QUALITY MODEL
136 #-----
137
138 #      FLAG_5 = -DWATER_QUALITY
139
140 #-----
141 #      PROJECTION          A Fortran90 wrapper for the Cartographic projection
142 #                           Software, proj4.
143 #                           Proj can be obtained from:
144 #                           http://www.remotesensing.org/proj/
145 #                           Thanks to: USGS
146 #
147 #                           The Proj fortran bindings can be obtained from:
148 #                           http://forge.nesc.ac.uk/projects/glimmer/
149 #                           Thanks to: Magnus Hagdorn (Magnus.Hagdorn@ed.ac.uk)
150 #
151 #                           !! NOTE THAT THE PROJ 4 LIBRARY MUST BE IN YOUR
152 #                           LD_LIBRARY_PATH FOR DYNAMIC LOADING!!
153 #
154 #-----
155
156 FLAG_6 = -DPROJ
157
158 PROJLIBS = -L$(LIBDIR) -lproj4 -lproj -lm
159 PROJINCS = -I$(INCDIR)
160
161 #-----
162 #      DATA_ASSIMILATION  INCLUDE NUDGING BASED DATA ASSIMILATION FOR
163 #                           CURRENT/TEMP/SALINITY/SST
164 #                           CAN BE ACTIVATED/DEACTIVATED AT RUN TIME WITH
165 #                           INPUT FILE CONTROL. (SEE exa_run.dat) FILE

```

(continues on next page)

(continued from previous page)

```

166 #           DEFAULT: NO DATA ASSIMILATION INCLUDED
167 #           UNCOMMENT TO INCLUDE DATA ASSIMILATION
168 #-----
169
170 #           FLAG_7 = -DDATA_ASSIM
171 #           include ${PETSC_DIR}/bmake/common/variables
172 #           OILIB = -lmkl_lapack -lmkl_em64t -lguide -lpthread
173
174 #           OILIB = -L/gpfs1/apps/intel/mkl/lib/em64t/ -lmkl_lapack95_ilp64 -lmkl_
175 ↪intel_ilp64 -lmkl_core -lmkl_sequential
176
177 # For Build on em64t computer (Guppy)
178 #           LIBDIR = $(LIBDIR) -L/usr/local/lib64
179 # For Build on Cluster (Typhoeus and Hydra)
180 #           LIBDIR = $(LIBDIR) -L/usr/local/lib/em64t
181 # For i386 computers at SMAST (salmon and minke)
182 # NO NEED TO ADD ANYTHING LIBS ARE IN THE DEFAULT PATH
183
184 #-----
185 #           IN UPWIND LEAST SQUARE SCHEME:
186 #           LIMITED_NO: NO LIMITATION
187 #           LIMITED_1 : FIRST ORDER LIMITATION
188 #           LIMITED_2 : SECOND ORDER LIMITATION( )
189 #           !!!!! ONLY ONE OF THE FLAGS BELOW CAN BE AND MUST BE CHOSEN
190 #-----
191
192 #           FLAG_8 = -DLIMITED_NO
193
194 #-----
195 #           Semi-Implicit time stepping method
196 #-----
197
198 #           FLAG_9 = -DSEMI_IMPLICIT
199 #           include ${PETSC_DIR}/bmake/common/variables
200
201 #-----
202 #           SOLID BOUNDARY      IF GCN, NO GHOST CELL
203 #                               IF GCY1, GHOST CELL IS SYMMETRIC RELATIVE TO BOUNDARY
204 #                               CELL EDGE
205 #                               IF GCY2, GHOST CELL IS SYMMETRIC RELATIVE TO MIDDLE
206 #                               POINT OF THE BOUNDARY CELL EDGE
207 #-----
208
209 #           FLAG_10 = -DGCN
210
211 #-----
212 #           TURBULENCE MODEL    USE GOTM TURBULENCE MODEL INSTEAD OF THE ORIGINAL
213 #                               FVCOM MELLOR-YAMADA 2.5 IMPLEMENTATION
214 #                               UNCOMMENT TO USE GOTM TURBULENCE MODEL
215 #           NOTE: You Must Build GOTM 4.x, GOTM 3.x used a different
216 #           do_turbulence interface and will not work.

```

(continues on next page)

(continued from previous page)

```

217 #-----
218 #
219 #     FLAG_11 = -DGOTM
220 #     GOTMLIB      = -L../GOTM_source/ -lturbulence -lutil -lmeanflow
221 #     GOTMINCS     = -I../GOTM_source/
222 #-----
223 #
224 #     EQUILIBRIUM TIDE
225 #-----
226 #
227 #     FLAG_12 = -DEQUI_TIDE
228 #-----
229 #
230 #     ATMOSPHERIC TIDE
231 #-----
232 #
233 #     FLAG_13 = -DATMO_TIDE
234 #-----
235 #
236 #     RIVER DISTRIBUTION OPTION:
237 #     THE STANDARD NAME LIST USES A CHARACTER STRING TO SET A FUNCION
238 #     DISTROBUTION. YOU CAN OPTIONALLY SPECIFY TO USE THE OLD STYLE,
239 #     FLOATING POINT DISTROBUTION HERE. USE THIS WHEN CONVERTING OLD-STYLE
240 #     RIVER INPUT FILES!
241 #-----
242 #
243 #     FLAG_14 = -DRIVER_FLOAT
244 #-----
245 #
246 #     Using A fully multidimensional positive definite advection
247 #     transport algorithm with small implicit diffusion.
248 #     Based on Smolarkiewicz, P. K; Journal of Computational
249 #     Physics, 54, 325-362, 1984
250 #-----
251 #
252 #     FLAG_15 = -DMPDATA -DTVD
253 #-----
254 #
255 #     Run Two-D Barotropic Mode Only
256 #-----
257 #
258 #     FLAG_16 = -DTWO_D_MODEL
259 #-----
260 #
261 #     Output 2-D Momentum Balance Checking
262 #-----
263 #
264 #     FLAG_17 = -DBALANCE_2D
265 #-----
266 #
267 #     OPEN BOUNDARY FORCING TYPE
268 #     DEFAULT: OPEN BOUNDARY NODE WATER ELEVATION FORCING

```

(continues on next page)

(continued from previous page)

```

269 #          UNCOMMENT TO SELECT BOTH OPEN BOUNDARY NODE WATER ELEVATION
270 #          FORCING AND OPEN BOUNDARY VOLUME TRANSPORT FORCING
271 #-----
272
273 #          FLAG_18 = -DMEAN_FLOW
274
275 #-----
276 #          OUTPUT TIDAL INFORMATION AT NTIDENODE and NTIDECCELL
277 #          FOR MEANFLOW CALCULATION.
278 #-----
279
280 #          FLAG_19 = -DTIDE_OUTPUT
281
282 #-----
283 #          dye release
284 #-----
285
286 #          FLAG_20 = -DDYE_RELEASE
287
288 #-----
289 #          SUSPENDED SEDIMENT MODEL:  UNCOMMENT TO INCLUDE MODEL
290 # ORIG : the sediment transport model developed by Geoffey Cowles in v3.1-v3.2
291 # CSTMS: Community Sediment Transport Modeling System with cohesive model
292 # DELFT: Sediment modeling system as Delft Flow (not included)
293 #
294 # note: only one model should be chosen for modeling.
295 #
296 # Utilities:
297 #   OFFLINE_SEDIMENT : run sediment with offline hydrodynamic forcing
298 #   FLUID_MUD : activate the 2-D fluid mud at bed-water interface
299 #-----
300
301 #          FLAG_21 = -DSEDIMENT
302 #          FLAG_211 = -DORIG_SED
303 #          FLAG_211 = -DCSTMS_SED
304 #
305 #          FLAG_22 = -DOFFLINE_SEDIMENT
306 #          FLAG_43 = -DFLUID_MUD
307 #
308 #-----
309 #          KALMAN FILTERS
310 #-----
311
312 #          FLAG_23 = -DRRKf
313 #          FLAG_23 = -DENKF
314 #          include ${PETSC_DIR}/bmake/common/variables
315 #          KFLIB = -lmkl_lapack -lmkl_em64t -lguide -llapack -lblas
316
317 # For Build on em64t computer (Guppy)
318 #          LIBDIR = $(LIBDIR) -L/usr/local/lib64
319 # For Build on Cluster (Typhoeus and Hydra)
320 #          LIBDIR = $(LIBDIR) -L/usr/local/lib/em64t

```

(continues on next page)

(continued from previous page)

```

321 # For i386 computers at SMAST (salmon and minke)
322 #   NO NEED TO ADD ANYTHING LIBS ARE IN THE DEFAULT PATH
323
324 #-----
325 #       Run One-D Mode with Biological Model
326 #-----
327
328 #       FLAG_24 = -DONE_D_MODEL
329
330 #-----
331 #       GENERAL BIOLOGICAL MODEL:   UNCOMMENT TO INCLUDE MODEL
332 #-----
333 #       FLAG_25 = -DFABM  #-DOFFLINE_FABM  # -DFABM_DIAG_OUT
334 #       BIOLIB      = -LBASE_SETUP_DIR/model/FABM-ERSEM/lib -lfabm
335 #       BIOINCS     = -IBASE_SETUP_DIR/model/FABM-ERSEM/include -IBASE_SETUP_DIR/
336 ↪model/FABM-ERSEM/include/yaml
337
338 #-----
339 #       Dynamic/Thermodynamic Ice
340 #-----
341 # NOTE: Must use -DSPHERICAL and -DHEAT_FLUX ----- this note only for old version v2.7
342 #       ICE_EMBEDDING must with SEMI_IMPLICIT
343 #       FLAG_26 = -DICE
344 #       FLAG_261 = -DICE_EMBEDDING
345
346 #-----
347 #       CALCULATE THE NET HEAT FLUX IN MODEL (THREE CHOICES):
348 #       1. CALCULATE THE NET HEAT FLUX USING COARE26Z
349 #       2. CALCULATE THE NET HEAT FLUX USING COARE26Z for Great Lake
350 #       3. CALCULATE THE NET HEAT FLUX USING SOLAR HEATING MODULE
351 ↪
352 #-----
353 #       FLAG_27 = -DHEATING_CALCULATED
354 #       FLAG_27 = -DHEATING_CALCULATED_GL
355 #       FLAG_27 = -DHEATING_SOLAR
356
357 #-----
358 #       AIR_PRESSURE FROM SURFACE FORCING
359 #-----
360
361 #       FLAG_28 = -DAIR_PRESSURE
362
363 #-----
364 # Visit online simulation mode
365 #-----
366
367 #       FLAG_29 = -DVISIT
368
369 #       VISITLIB      = -lm -ldl -lsimf -lsim -lpthread
370 #       VISITLIBPATH = $(LIBDIR)

```

(continues on next page)



(continued from previous page)

```

371 #      VISITINCPATH = $(INCDIR)
372
373
374 # USE DEVELOPER INSTALL VISIT
375 #      VISITLIBPATH =
376 #      VISITLIB      = -lm -ldl -lsimf -lsim -lpthread
377 #      VISITINC      =
378
379 #-----
380 #      NON-HYDROSTATIC MODEL:
381 #-----
382
383 #      FLAG_30 = -DNH
384 #      include ${PETSC_DIR}/bmake/common/variables
385
386 #-----
387 #      PARTICLE TRACKING
388 #-----
389
390 #      FLAG_31 = -DLAG_PARTICLE
391
392 #-----
393 #      WAVE-CURRENT INTERACTION
394 #-----
395 #      FLAG_32 = -DWAVE_CURRENT_INTERACTION
396 #      FLAG_33 = -DPLBC
397 #      NOTE! This option is for wave code
398 #      FLAG_34 = -DEXPLICIT
399 #      WAVE ONLY
400 #      FLAG_35 = -DWAVE_ONLY
401 # Svendsen Roller contribution
402 #      FLAG_36 = -DWAVE_ROLLER
403 #      FLAG_37 = -DWAVE_OFFLINE
404 #      include ${PETSC_DIR}/bmake/common/variables
405 #-----
406 #      THIN-DAM MODEL
407 #-----
408 #      FLAG_38 = -DTHIN_DAM
409
410 #-----
411 #      PWP MIXED LAYER MODEL:
412 #-----
413
414 #      FLAG_39 = -DPWP
415
416 #-----
417 #      VERTICAL ADVECTION LIMITER:
418 #      FOR S-COORDINATES, DON'T USE THIS FLAG
419 #-----
420
421 #      FLAG_40 = -DLIMITER_VER_ADV
422

```

(continues on next page)

(continued from previous page)

```

423 #-----
424 #     PETSC Version
425 #     If your PETSc is 2.3.2 or older, uncomment this flag
426 #-----
427 #     FLAG_41 = -DOLD_PETSC
428
429 #-----
430 #     SPECIAL PARTITION
431 #     This flag can make sure the identical repeat run for same amount of CPUs
432 #-----
433 #     FLAG_42 = -DPARTITION_SPECIAL
434
435 #-----
436 #     DEVELOPMENT FLAGS
437 #     FOR BETA WORK ONLY
438 #-----
439
440 #     FLAG_101 = -DDEVELOP1
441 #     FLAG_102 = -DDEVELOP2
442 #     FLAG_103 = -DDEVELOP3
443 #     FLAG_104 = -DDEVELOP4
444 #     FLAG_105 = -DDEVELOP5
445
446 #-----
447 #     SELECT COMPILER/PLATFORM SPECIFIC DEFINITIONS
448 #     SELECT FROM THE FOLLOWING PLATFORMS OR USE "OTHER" TO DEFINE
449 #     THE FOLLOWING VARIABLES:
450 #     CPP:  PATH TO C PREPROCESSOR
451 #     FC:   PATH TO FORTRAN COMPILER (OR MPI COMPILE SCRIPT)
452 #     OPT:  COMPILER OPTIONS
453 #     MPILIB: PATH TO MPI LIBRARIES (IF NOT LINKED THROUGH COMPILE SCRIPT)
454 #-----
455 #-----
456 # Intel/MPI Compiler Definitions (PML)
457 #-----
458 #     CPP      = mpiicc -E
459 #     COMPILER = -DINTEL
460 #     FC       = mpiifort
461 #     DEBFLGS  = #-check all
462 #     OPT      = -O3 -L/gpfs1/apps/intel/compilers_and_libraries/linux/mpi/intel64/
463 #     lib      -I/gpfs1/apps/intel/compilers_and_libraries/linux/mpi/intel64/include/ -xHost #-
464 #     init=zero -init=arrays -ftrapuv
465 #     CLIB     =
466 #     CC       = mpiicc
467 #     CFLAGS   =
468 #-----
469 # Intel/MPI Compiler Definitions (PML) Debugging
470 #-----
471 #     COMPILER = -DIFORT
472 #     CPP      = /lib/cpp
473 #     CPPFLAGS = $(DEF_FLAGS) -P -traditional -DINTEL CPPMACH=-DNOGUI -I/opt/mpi/
474 #     mpibull2-current/include

```

(continues on next page)

(continued from previous page)

```

472 #      FC      = ifort
473 #      DEBFLGS = #-check all
474 #      OPT      = -I/opt/mpi/mpibull2-current/include -g -traceback -warn -nofor_
↳main -fp-model precise -traceback -fpe0 -keep
475 #      OILIB    = -L/opt/intel/cmkl/10.0.1.014/lib/em64t -Wl,-rpath=/opt/intel/cmkl/10.
↳0.1.014/lib/em64t -i-static -L/opt/mpi/mpibull2-current/lib -lmpi -L/usr/lib64 -
↳libverbs -L/opt/mpi/mpibull2-current/lib/pmi -lpmi
476 #      CC      = icc
477 #      CFLAGS   = -g -I/opt/mpi/mpibull2-current/include -traceback
478 #-----
479 #      COMPAQ/ALPHA Definitions
480 #-----
481 #      COMPILER = -DCOMPAQ
482 #      CPP      = /bin/cpp
483 #      FC      = f90
484 #      DEBFLGS = # -check bounds -check overflow -g
485 #      OPT      = -fast -arch ev6 -fpe1
486 #-----
487 #      CRAY Definitions
488 #-----
489 #      COMPILER = -DCRAY
490 #      CPP      = /opt/ctl/bin/cpp
491 #      FC      = f90
492 #      DEBFLGS =
493 #      OPT      =
494 #-----
495 #      Linux/Portland Group Definitions
496 #-----
497 #      CPP      = /usr/bin/cpp
498 #      COMPILER =
499 #      FC      = pgf90
500 #      DEBFLGS = -Mbounds -g -Mprof=func
501 #      OPT      = #-fast -Mvect=assoc,cachesize:512000,sse
502 #-----
503 #      Intel Compiler Definitions
504 #-----
505 #      CPP      = /usr/bin/cpp
506 #      COMPILER = -DIFORT
507 #      FC      = ifort
508 #      CC      = icc
509 #      CXX      = icc
510 #      CFLAGS   = -O3
511 #      DEBFLGS = #-check all
512 # Use 'OPT = -O0 -g' for fast compile toBASE_SETUP_DIRthe make
513 # Use 'OPT = -xP' for fast run on em64t (Hydra and Guppy)
514 # Use 'OPT = -xN' for fast run on ia32 (Salmon and Minke)
515 #      OPT      = -O0 -g
516 #      OPT      = -xP
517 # Do not set static for use with visit!
518 #      VISOPT    = -Wl,--export-dynamic
519 #      LDFLAGS   = $(VISITLIBPATH)
520 #-----

```

(continues on next page)

(continued from previous page)

```

521 # Intel/MPI Compiler Definitions (SMAST)
522 #-----
523 #      CPP      = /usr/bin/cpp
524 #      COMPILER = -DIFORT
525 #      CC       = mpicc
526 #      CXX      = mpicxx
527 #      CFLAGS   = -O3
528 #      FC       = mpif90
529 #      DEBFLGS  = -check all -traceback
530 # Use 'OPT = -O0 -g' for fast compile toBASE_SETUP_DIRthe make
531 # Use 'OPT = -xP' for fast run on em64t (Hydra and Guppy)
532 # Use 'OPT = -xN' for fast run on ia32 (Salmon and Minke)
533 #      OPT      = -O0 -g
534 #      OPT      = -axN -xN
535 #      OPT      = -O3
536 # Do not set static for use with visit!
537 #      VISOPT   = -Wl,--export-dynamic
538 #      LDFLAGS  = $(VISITLIBPATH)
539 #-----
540 # gfortran defs
541 #-----
542 #      CPP      = /usr/bin/cpp
543 #      COMPILER = -DGFORTRAN
544 #      FC       = gfortran -O3
545 #      DEBFLGS  =
546 #      OPT      =
547 #      CLIB     =
548 #-----
549 # absoft / mac os x defs
550 #-----
551 #      CPP      = /usr/bin/cpp
552 #      COMPILER = -DABSOFT
553 #      FC       = f90 -O3 -lU77
554 #      DEBFLGS  =
555 #      OPT      =
556 #      CLIB     =
557 #-----
558 # IBM/AIX Definitions
559 #-----
560 #      COMPILER = -DAIX
561 #      CPP      = /usr/local/bin/cpp
562 #      FC       = mpixlf90 -qsuffix=f=f90
563 #      DEBFLGS  = # -qcheck -C -g
564 #      OPT      = -O -qarch=pwr4 -qtune=pwr4 -bmaxdata:0x80000000 -qhot -qmaxmem=8096
565 #-----
566 # APPLE OS X/XLF Definitions (G5)
567 #-----
568 #      COMPILER = -DAIX
569 #      CPP      = /usr/bin/cpp
570 #      FC       = /opt/ibmcmp/xlf/8.1/bin/xlf90 -qsuffix=f=f90
571 #      DEBFLGS  = # -qcheck -C -g
572 #      OPT      = -O5 -qarch=g5 -qtune=g5 -qhot -qmaxmem=8096 -qunroll=yes -Wl,-
↪ stack_size,10000000

```

(continues on next page)

(continued from previous page)

```

573 #-----
574 # ARCHER Intel/MPI Compiler
575 #-----
576 #      CPP      = /usr/bin/cpp
577 #      COMPILER = -DIFORT
578 #      CC       = cc
579 #      CXX      = CC
580 #      CFLAGS   = -O3
581 #      FC       = ftn
582 ##      DEBFLGS = -check all
583 #      OPT      = -O3
584 #      COPTIONS = -c89
585 #-----
586 #=====
587 # END USER DEFINITION SECTION
588 #=====
589      CPPFLAGS = $(DEF_FLAGS) $(COMPILER)
590      FFLAGS   = $(DEBFLGS) $(OPT)
591      MDEPFLAGS = --cpp --fext=f90 --file=-
592      RANLIB    = ranlib
593      AR        = ar rc
594 #-----
595 # CAT Preprocessing Flags
596 #-----
597      CPPARGS = $(CPPFLAGS) $(DEF_FLAGS) $(FLAG_1) $(FLAG_2) \
598 $(FLAG_3) $(FLAG_4) $(FLAG_5) $(FLAG_6) \
599 $(FLAG_7) $(FLAG_8) $(FLAG_9) $(FLAG_10) \
600 $(FLAG_11) $(FLAG_12) $(FLAG_13) $(FLAG_14) \
601 $(FLAG_15) $(FLAG_16) $(FLAG_17) $(FLAG_18) \
602 $(FLAG_19) $(FLAG_20) $(FLAG_21) $(FLAG_22) \
603 $(FLAG_23) $(FLAG_24) $(FLAG_25) $(FLAG_26) \
604 $(FLAG_27) $(FLAG_28) $(FLAG_29) $(FLAG_30) \
605 $(FLAG_31) $(FLAG_32) $(FLAG_33) $(FLAG_34) \
606 $(FLAG_35) $(FLAG_36) $(FLAG_37) $(FLAG_38) \
607 $(FLAG_39) $(FLAG_40) $(FLAG_41) $(FLAG_42) \
608 $(FLAG_43) $(FLAG_LAM)\
609 $(FLAG_101) $(FLAG_102) $(FLAG_103) $(FLAG_104) $(FLAG_105)\
610 $(FLAG_211) $(FLAG_212) $(FLAG_213) $(FLAG_251) $(FLAG_261)
611 #-----
612 # Libraries
613 #-----
614
615      LIBS = $(LIBDIR) $(CLIB) $(PARLIB) $(IOLIBS) $(DTLIBS)\
616 $(MPILIB) $(GOTMLIB) $(KFLIB) $(BIOLIB) \
617 $(OILIB) $(VISITLIB) $(PROJLIBS) $(PETSC_LIB)
618
619      INCS = $(INCDIR) $(IOINCS) $(GOTMINCS) $(BIOINCS)\
620 $(VISITINCPATH) $(PROJINCS) $(DTINCS) \
621 $(PETSC_FC_INCLUDES)

```

The key lines to change are:

- 74–75

- 174
- 458–465

After building FVCOM-ERSEM, we suggest you use a HPC scheduler, for example, [SLURM](#) to run example. An example of the SLURM script used here is given below:

```
1  #!/bin/bash --login
2
3  #SBATCH --nodes=4
4  #SBATCH --ntasks-per-node=20
5  #SBATCH --threads-per-core=1
6  #SBATCH --job-name=estuary
7  #SBATCH --partition=all
8  #SBATCH --time=48:00:00
9  ##SBATCH --mail-type=ALL
10 ##SBATCH --mail-user=your_mail@pml.ac.uk
11
12 # Set the number of processes based on the number of nodes we have `select'ed.
13 np=$SLURM_NTASKS
14
15 # Export the libraries to LD_LIBRARY_PATH
16 export LD_LIBRARY_PATH=$(readlink -f $WORKDIR/install/lib):$LD_LIBRARY_PATH
17
18 set -eu
19 ulimit -s unlimited
20
21 # Number of months to skip
22 skip=0
23
24 # d53f5083 = FVCOM v3.2.2, M-Y, HEATING_ON, ceto
25 binary=bin/fvcom
26 grid=${grid:-"estuary"}
27 casename="${grid}"
28
29 # Make sure any symbolic links are resolved to absolute path
30 export WORKDIR=$(readlink -f $(pwd))
31
32 # Set the number of threads to 1
33 # This prevents any system libraries from automatically
34 # using threading.
35 export OMP_NUM_THREADS=1
36
37 # Magic stuff from the Atos scripts.
38 export I_MPI_PIN_PROCS=0-19
39 export I_MPI_EXTRA_FILESYSTEM=on
40 export I_MPI_EXTRA_FILESYSTEM_LIST=gpfs
41 export I_MPI_PMI_LIBRARY=/usr/lib64/libpmi.so
42
43 # Change to the directory from which the job was submitted. This should be the
44 # project "run" directory as all the paths are assumed relative to that.
45 cd $WORKDIR
46
47 if [ ! -d ./logs/slurm ]; then
```

(continues on next page)

(continued from previous page)

```

48     mkdir -p ./logs/slurm
49 fi
50 mv *.out logs/slurm || true
51 if [ -f ./core ]; then
52     rm core
53 fi
54
55 if [ ! -d ./output ]; then
56     mkdir -p ./output
57 fi
58
59 # Iterate over the months in the year
60
61 # Launch the parallel job
62 srun -n $np $binary --casename=$casename --dbg=0 > logs/${casename}-${SLURM_JOBID}.log
63
64
65 # Check if we crashed and if so, exit the script, bypassing the restart
66 # file creation.
67 if grep -q NaN logs/${casename}-${SLURM_JOBID}.log; then
68     echo "NaNs in the output. Halting run."
69     exit 2
70 fi

```

### Example output from FVCOM-ERSEM

We provide two python scripts to demonstrate how to visualise both the input files and the output files. The plotting uses PyFVCOM, we suggest you ask for access [here](#), however, a version of the code is available on [GitHub](#) as well as it being [pip installable](#).

python input plot script

```

1  import os
2  import numpy as np
3  from netCDF4 import Dataset, num2date
4  from PyFVCOM.read import MFileReader
5  import matplotlib.pyplot as plt
6
7  # simple grid
8  dir_path = os.path.dirname(os.path.realpath(__file__))
9  estuary_path = os.path.join('model',
10                             'estuary')
11  figure_path = os.path.join(estuary_path,
12                             "figures")
13  fvcom_files = os.path.join(estuary_path,
14                             'output',
15                             'estuary_avg_0001.nc')
16
17
18  if not os.path.isdir(figure_path):
19      os.makedirs(figure_path)

```

(continues on next page)

(continued from previous page)

```

20
21 print("Plotting mesh")
22 fig = plt.figure(figsize=(8, 6))
23 ax = fig.add_subplot(111)
24 dims = {'siglay': [0]}
25 fvcom = MFileReader(fvcom_files, dims=dims)
26 DATA = fvcom.grid.h
27 p1 = ax.tripcolor(fvcom.grid.x, fvcom.grid.y,
28                 fvcom.grid.triangles, DATA, shading='flat', edgecolors='k')
29 plt.colorbar(p1)
30 ax.set_aspect('equal', 'box')
31 name = os.path.join(fvcom_path, 'bathymetry')
32 fig.savefig(name, bbox_inches='tight', pad_inches=0.2, dpi=300)
33 plt.close()
34
35 print("Plotting wind forcing")
36 # check forcing
37 idtime = 100
38 fvcom_files = os.path.join(fvcom_path,
39                             'input',
40                             '2D_forcing_50_years_20180927_new_grd.nc')
41 fvcom_frc = MFileReader(fvcom_files, variables=['uwind_speed',
42                                                 'vwind_speed',
43                                                 'short_wave',
44                                                 'net_heat_flux'])
45 uwind = np.squeeze(fvcom_frc.data.uwind_speed[idtime, :])
46 vwind = np.squeeze(fvcom_frc.data.vwind_speed[idtime, :])
47
48 # wind speed
49 fig = plt.figure(figsize=(8, 6))
50 ax = fig.add_subplot(111)
51 scale = 75
52 p1 = ax.quiver(fvcom.grid.xc[0:-1:2],
53               fvcom.grid.yc[0:-1:2],
54               uwind[0:-1:2],
55               vwind[0:-1:2],
56               scale=scale,
57               headlength=8,
58               headaxislength=8,
59               width=0.0015)
60
61 arrow_legend = 2
62 plt.text(5e4, 6e4, '{} m/s'.format(str(arrow_legend)), fontsize=10)
63 q = ax.quiver(5e4, 5e4, arrow_legend, 0, scale=scale,
64               headlength=8, headaxislength=8, width=0.0015)
65 ax.set_aspect('equal', 'box')
66 name = os.path.join(fvcom_path, 'wind_forcing')
67 fig.savefig(name, bbox_inches='tight', pad_inches=0.2, dpi=300)
68 plt.close()
69
70 print("Plotting short wave")
71 # short wave

```

(continues on next page)



(continued from previous page)

```

72 fvcom = fvcom_frc
73 fig = plt.figure(figsize=(8, 6))
74 ax = fig.add_subplot(111)
75 idnode = 500
76 shortwave = getattr(fvcom_frc.data, 'short_wave')
77 ax.plot(fvcom.time.datetime[0:8*365*1], shortwave[0:8*365*1, idnode])
78 ax.set_xlabel('Time (DD-MM HH)', fontsize='x-large')
79 ax.set_ylabel('W/m2')
80 name = os.path.join(fvcom_path, 'short_wave')
81 fig.savefig(name, bbox_inches='tight', pad_inches=0.2, dpi=300)
82 plt.close()
83
84
85 print("Plotting net heat flux")
86 # net heat flux
87 fig = plt.figure(figsize=(8, 6))
88 ax = fig.add_subplot(111)
89 idnode = 500
90 netheat = getattr(fvcom_frc.data, 'net_heat_flux')
91 ax.plot(fvcom.time.datetime, netheat[:, idnode])
92 ax.set_xlabel('Time (year)', fontsize='x-large')
93 ax.set_ylabel('W/m2')
94 name = os.path.join(fvcom_path, 'net_heat_flux')
95 fig.savefig(name, bbox_inches='tight', pad_inches=0.2, dpi=300)
96 plt.close()
97
98 print("Plotting OBC ts")
99 # OBC T S
100 obc_node = 40
101 fvcom_files = os.path.join(estuary_path,
102                             'output',
103                             'estuary_avg_0001.nc')
104 fvcom = MFileReader(fvcom_files, dims={'time': range(1)})
105 fvcom_files_obc = os.path.join(estuary_path,
106                                 'input',
107                                 '2D_obc_ts_forcing.nc')
108 nc = Dataset(fvcom_files_obc).variables
109 fig = plt.figure(figsize=(10, 5))
110
111 ax = fig.add_subplot(121)
112 month = np.arange(1, 5)
113 for month in month:
114     obc_temp = nc['obc_temp'][month+12*0, :, obc_node]
115     depth = fvcom.grid.siglay_z[:, obc_node]
116     if month in np.arange(1, 7):
117         ax.plot(obc_temp, -depth, label=month)
118     else:
119         ax.plot(obc_temp, -depth, '--', label=month)
120 ax.legend(loc='upper right')
121 ax.set_xlim(4.5, 33)
122 ax.set_xlabel('Temp (degC)')
123 ax.set_ylabel('Depth (m)')

```

(continues on next page)

(continued from previous page)

```

124     ax.set_title('obc node {}'.format(str(obc_node)))
125
126 ax = fig.add_subplot(122)
127 month = np.arange(1, 5)
128 for month in month:
129     obc_salt = nc['obc_salinity'][month+12*0, :, obc_node]
130     depth = fvcom.grid.siglay_z[:, obc_node]
131     if month in np.arange(1, 7):
132         ax.plot(obc_salt, -depth, label=month)
133     else:
134         ax.plot(obc_salt, -depth, '--', label=month)
135     ax.legend(loc='upper right')
136     ax.set_xlim(28, 37)
137     ax.set_xlabel('Salinity (psu)')
138     ax.set_ylabel('Depth (m)')
139     ax.set_title('obc node {}'.format(str(obc_node)))
140 name = os.path.join(figure_path, 'obc_TS_node_{}'.format(str(obc_node)))
141 fig.savefig(name, bbox_inches='tight', pad_inches=0.2, dpi=300)
142 plt.close()
143
144 print("Plotting river forcing")
145 # River forcing
146 tracer_f = "2D_bio_River_50_years_20180821_monthly_river_2_tracers_T1T2.nc"
147 fvcom_file_river = \
148     os.path.join(estuary_path,
149                 'input',
150                 tracer_f)
151 nc = Dataset(fvcom_file_river)
152 nc.variables.keys()
153 varlist = ['river_flux', 'river_temp', 'river_salt', 'river_sed',
154            'N4_n', 'N3_n']
155 fig = plt.figure(figsize=(12, 8))
156 time_var = nc['time']
157 dtime = num2date(time_var[:], time_var.units)
158 for var, num in zip(varlist, np.arange(1, 3)):
159     ax = fig.add_subplot(3, 2, num)
160     if var in ('river_flux'):
161         river_var = np.sum(nc[var][:, :], 1)
162     else:
163         river_var = nc[var][:, 0]
164     ax.plot(dtime, river_var)
165     ax.set_title('{} ({}).format(nc[var].long_name, nc[var].units))
166     ax.set_xlabel('Time (month)')
167     plt.tight_layout()
168 name = os.path.join(figure_path, 'river_flux1')
169 fig.savefig(name, bbox_inches='tight', pad_inches=0.2, dpi=300)
170 plt.close()
171
172 varlist = ['N1_p', 'N5_s', 'O3_c', 'O3_TA', 'O3_bioalk', 'O2_o']
173 fig = plt.figure(figsize=(12, 8))
174 for var, num in zip(varlist, np.arange(1, 3)):
175     ax = fig.add_subplot(3, 2, num)

```

(continues on next page)

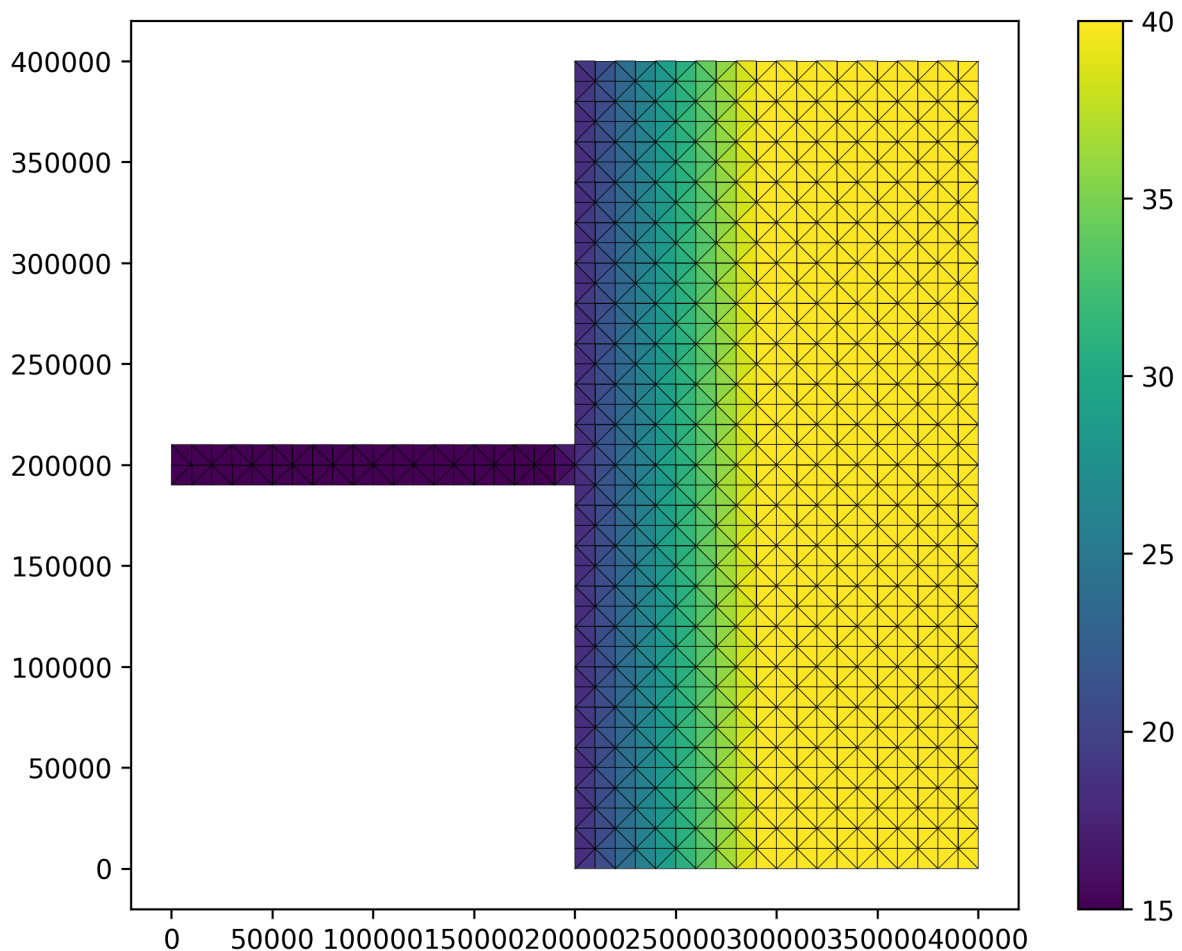
(continued from previous page)

```

176     river_var = nc[var][:, 0]
177     ax.plot(dtime, river_var)
178     ax.set_title('{} {}'.format(nc[var].long_name, nc[var].units))
179     ax.set_xlabel('Time (month)')
180     plt.tight_layout()
181     name = os.path.join(figure_path, 'river_flux2')
182     fig.savefig(name, bbox_inches='tight', pad_inches=0.2, dpi=300)
183     plt.close()

```

Using the input python plot script, we generate the domain as follows:



python output plot script

```

1  import os
2  import glob
3  import numpy as np
4
5  from PyFVCOM.read import MFileReader
6  from pythontools.visual.utils import colourmap
7  import PyFVCOM as pf

```

(continues on next page)

(continued from previous page)

```

8 from PyFVCOM.plot import Plotter, Depth
9 import matplotlib.pyplot as plt
10 import matplotlib
11
12 project0 = 'estuary'
13 casename = 'ideal'
14 base = os.path.join('model', 'estuary')
15 experiment = ''
16 suffix = ''
17 level_label = 'vertical_transect'
18 level_label2 = 'surface'
19 fname = 'estuary_avg_0001.nc'
20 fpath1 = os.path.join(base, 'output', fname)
21 fvcom_files = sorted(glob.glob(fpath1))
22
23 noisy = True
24 if noisy:
25     print(fpath1, flush=True)
26     print(fvcom_files, flush=True)
27
28 if not fvcom_files:
29     raise Exception('Cannot find FVCOM output: {}'.format(fpath1))
30
31 MyFile = fvcom_files
32
33 plot_map = True # to plot horizontal map
34 save_plot = True # to save the plot
35
36 fvcom = MFileReader(MyFile)
37
38 varlist = ['GPP', 'temp', 'salinity',
39            'P1_Ch1', 'O3_c', 'R6_c', 'N1_p', 'N3_n',
40            'tracer1_c', 'total_chl']
41
42 clims = {
43     'temp': [5, 25],
44     'O3_c': [1900, 2300],
45     'R6_c': [0, 200],
46     'salinity': [0, 31]
47 }
48
49 # plot all time series, maybe too much
50 time_indices = range(0, fvcom.dims.time)
51 level = 0 # select vertical layer (0: surface, -1: bottom) if plot_map = True
52
53 plt.rcParams['axes.facecolor'] = '0.6'
54 for var in varlist:
55     baseDir = os.path.join(base, 'figures', var)
56     baseName = os.path.join(baseDir, var)
57
58     if var in ('uv'):
59         plotvars = ['u', 'v']

```

(continues on next page)

(continued from previous page)

```

60 elif var == 'total_chl':
61     plotvars = ['P1_Ch1', 'P2_Ch1', 'P3_Ch1', 'P4_Ch1']
62 elif var == 'netPP':
63     plotvars = ['P1_Ch1', 'P2_Ch1', 'P3_Ch1', 'P4_Ch1']
64 elif var == 'GPP':
65     plotvars = ["P1_f03PIc", "P2_f03PIc", "P3_f03PIc", "P4_f03PIc"]
66 else:
67     plotvars = [var]
68
69 if noisy:
70     print('Loading {} data from netCDF... '.format(var),
71           end='',
72           flush=True)
73     if save_plot:
74         print('Saving files to {}'.format(baseDir), flush=True)
75         if not os.path.isdir(baseDir):
76             os.makedirs(baseDir)
77         print('Working on {}...'.format(var), flush=True),
78
79 fvcom = MFileReader(MyFile, variables=plotvars)
80 fvcom.grid.lon = fvcom.grid.x
81 fvcom.grid.lat = fvcom.grid.y
82 positions = np.array(((0, 2e5), (4e5, 2e5)))
83 indices, distance = fvcom.horizontal_transect_nodes(positions)
84 cmap = colourmap(var)
85
86 if var in ('total_chl'):
87     setattr(
88         fvcom.data, var, fvcom.data.P1_Ch1 + fvcom.data.P2_Ch1 +
89         fvcom.data.P3_Ch1 + fvcom.data.P4_Ch1)
90     attributes = type('attributes', (object, ), {})(
91         setattr(attributes, 'long_name', 'Total chlorophyll')
92         setattr(attributes, 'units', fvcom.attrs.P1_Ch1.units)
93         setattr(fvcom.attrs, var, attributes)
94 elif var in ('total_food'):
95     setattr(fvcom.data, var, fvcom.data.P1_c + fvcom.data.P2_c
96         + fvcom.data.P3_c + fvcom.data.P4_c + fvcom.data.Z5_c
97         + fvcom.data.Z5_c + fvcom.data.R8_c + fvcom.data.R6_c
98         + fvcom.data.R4_c)
99     attributes = pf.utilities.general.PassiveStore()
100     setattr(attributes, 'long_name', 'Total mussel food ')
101     setattr(attributes, 'units', fvcom.attrs.P1_c.units)
102     setattr(fvcom.attrs, var, attributes)
103     attributes = pf.utilities.general.PassiveStore()
104     setattr(fvcom.data, var, getattr(fvcom.data, var) / 1000000)
105     setattr(attributes, 'long_name', 'Integrated Total mussel food')
106     setattr(attributes, 'units', 'Kg\C')
107     setattr(fvcom.attrs, var, attributes)
108 elif var in ('GPP'):
109     setattr(fvcom.data, var, fvcom.data.P1_f03PIc +
110         fvcom.data.P2_f03PIc + fvcom.data.P3_f03PIc +
111         fvcom.data.P4_f03PIc)

```

(continues on next page)

(continued from previous page)

```

112     attributes = pf.utilities.general.PassiveStore()
113     setattr(attributes, 'long_name', 'Gross Primary Productivity')
114     setattr(attributes, 'units', fvcom.attrs.P1_f03PIc.units)
115     setattr(fvcom.attrs, var, attributes)
116
117     if var in clims:
118         clim = clims[var]
119     else:
120         clim = [
121             np.nanpercentile(getattr(fvcom.data, var), 3),
122             np.nanpercentile(getattr(fvcom.data, var), 97)
123         ]
124     plot = Plotter(fvcom,
125                   figsize=(20, 20),
126                   res='i',
127                   cb_label='{} ({}).format(
128                       getattr(fvcom.attrs, var).long_name,
129                       getattr(fvcom.attrs, var).units),
130                   cmap=cmap,
131                   vmin=clim[0],
132                   vmax=clim[1],
133                   cartesian=True)
134
135
136
137     # Plot a temperature transect between two locations.
138     depth_plot = Depth(fvcom,
139                       figsize=(20, 9),
140                       cb_label='{} ({}).format(
141                           getattr(fvcom.attrs, var).long_name,
142                           getattr(fvcom.attrs, var).units),
143                       cmap=cmap)
144     depth_plot.axes.set_xlabel('Distance (km)')
145     depth_plot.axes.set_ylabel('Depth (m)')
146     for ntime in time_indices:
147         print("Time step index: {}".format(ntime))
148
149         # Make a plot of the surface temperature.
150         plot.plot_field(np.squeeze(getattr(fvcom.data, var))[ntime, level, :])
151         plot.axes.set_title(
152             fvcom.time.datetime[ntime].strftime('%Y-%m-%d %H:%M:%S'))
153
154         if save_plot:
155             plot.figure.savefig('{}{}_{}_{}_{:04d}.png'.format(
156                 baseName, suffix, experiment, level_label2, ntime),
157                 bbox_inches='tight',
158                 pad_inches=0.2,
159                 dpi=120)
160
161         # fill_seabed makes the part of the plot below the seabed grey.
162         # plot.plot_slice(distance / 1000, # to kilometres from metres
163         depth_plot.plot_slice(

```

(continues on next page)

(continued from previous page)

```

164         fvcom.grid.lon[indices] / 1000, # to kilometres from metres
165         fvcom.grid.siglay_z[:, indices],
166         getattr(fvcom.data, var)[ntime, :, indices],
167         fill_seabed=True,
168         vmin=clim[0],
169         vmax=clim[1])
170     depth_plot.axes.set_title(
171         fvcom.time.datetime[ntime].strftime('%Y-%m-%d %H:%M:%S'))
172     depth_plot.axes.set_xlim(
173         right=(fvcom.grid.lon[indices] /
174                1000).max()) # set the x-axis to the data range
175
176     if save_plot:
177         depth_plot.figure.savefig('{}{}_{}_{}_{:04d}.png'.format(
178             baseName, suffix, experiment, level_label, ntime),
179             bbox_inches='tight',
180             pad_inches=0.2,
181             dpi=120)

```

The following plots videos are produced from the plots produced by the python output plot script.

Gros primary production

Phosphate phosphorus

Nitrate nitrogen

Carbonate total dissolved inorganic carbon

Diatoms chlorophyll

Medium-sized POM carbon

Salinity

Temperature

Total chorophyll

Tracer1 concentration

## 2.2.5 NEMO: Atlantic Margin Model

This tutorial gives an end-to-end example on how to install and then run NEMO-ERSEM on the 7 km Atlantic Margin Model 7 (AMM7) domain using a high performance computing (HPC) machine. Here we have used UK National Supercomputing Service ARCHER2 with a [singularity container](#) with NEMO-ERSEM installed on it.

This tutorial is based on utilising a [singularity container](#) with a NEMO-ERSEM install within. The configuration scripts used in the installation can be found [here](#). You will also need access to the shared folder within the ARCHER2 project id *n01* to obtain the forcing files.

### Obtaining NEMO-ERSEM container

The key packages that are installed into NEMO-ERSEM container are:

- NEMO (physical ocean model)
- XIOS (input/output server)
- FABM (biogeochemical framework)
- ERSEM (biogeochemical model)

NEMO, ERSEM and FABM are freely available on GitHub and XIOS is available through the NEMO consortium svn server. Full instructions how to build the container can be found [here](#). We note these instructions are based on the “Containerisation of NEMO Employing Singularity” (CoNES) repository.

### Running the container on ARCHER2

The instructions to generate the [slurm](#) HPC scheduling scripts can be found [here](#). These are used to run the model on ARCHER2.

The scripts generated via [mkslurm](#) require two executables, namely *xios\_server.exe* and *nemo*. In a native installation, one would simply link the *XIOS* and *NEMO* executables to *xios\_server.exe* and *nemo*. However, when using the singularity container, you are required to create *xios\_server.exe* and *nemo* executables based on the following bash script:

Listing 1: Example script to run container. User is required to change *[RUNING-DIR]* and *[INPUT-DIR]* to the corresponding directories, and *[NEMO-XIOS]* to either *nemo* or *xios*.

```
#!/bin/bash
BIND_OPTS="-B [RUNING-DIR], "
BIND_OPTS="${BIND_OPTS}[INPUT-DIR], "
BIND_OPTS="${BIND_OPTS}/usr/lib64/liblustreapi.so:/opt/lib64/liblustreapi.so,"
BIND_OPTS="${BIND_OPTS}/usr/lib64/liblustreapi.so.1:/opt/lib64/liblustreapi.so.1,"
BIND_OPTS="${BIND_OPTS}/usr/lib64/liblustreapi.so.1.0.0:/opt/lib64/liblustreapi.so.1.0.0,
↪"
BIND_OPTS="${BIND_OPTS}/opt/cray/libfabric/1.11.0.4.71/lib64:/opt/fabric,"
BIND_OPTS="${BIND_OPTS}/usr/lib64/libpals.a:/opt/lib64/libpals.a,"
BIND_OPTS="${BIND_OPTS}/usr/lib64/libpals.so:/opt/lib64/libpals.so,"
BIND_OPTS="${BIND_OPTS}/usr/lib64/libpals.so.0:/opt/lib64/libpals.so.0,"
BIND_OPTS="${BIND_OPTS}/usr/lib64/libpals.so.0.0.0:/opt/lib64/libpals.so.0.0.0,"
BIND_OPTS="${BIND_OPTS}/opt/cray/pe/mpich/8.1.9/ofc/gnu/9.1:/opt/mpi/install/",
BIND_OPTS="${BIND_OPTS}/opt/cray/pe/pmi/6.0.13/lib,"
BIND_OPTS="${BIND_OPTS}/var/spool/slurmd/mpi_cray_shasta"
```

(continues on next page)



(continued from previous page)

```

export SINGULARITYENV_LD_LIBRARY_PATH=/opt/hdf5/install/lib:$SINGULARITYENV_LD_LIBRARY_
↪PATH
export SINGULARITYENV_LD_LIBRARY_PATH=/opt/mpi/install/lib-abi-mpich:$SINGULARITYENV_LD_
↪LIBRARY_PATH
export SINGULARITYENV_LD_LIBRARY_PATH=/.singularity.d/libs:$SINGULARITYENV_LD_LIBRARY_
↪PATH
export SINGULARITYENV_LD_LIBRARY_PATH=/opt/cray/pe/pmi/6.0.13/lib:$SINGULARITYENV_LD_
↪LIBRARY_PATH
export SINGULARITYENV_LD_LIBRARY_PATH=/opt/lib64:$SINGULARITYENV_LD_LIBRARY_PATH
export SINGULARITYENV_LD_LIBRARY_PATH=/opt/fabric:$SINGULARITYENV_LD_LIBRARY_PATH
export LD_LIBRARY_PATH=/opt/hdf5/install/lib
export OMP_NUM_THREADS=1

singularity run ${BIND_OPTS} --home [RUNING-DIR] nemo.sif [NEMO-XIOS]

```

The CoNES [documentation](#) gives additional information on how to run NEMO singularity containers.

### Example output from NEMO-ERSEM

To visualise NEMO output, we recommend using [nctoolkit](#). A basic example of how to use *nctoolkit* is given below. Within the [documentation](#) of *nctoolkit* one will find additional example uses.

Listing 2: Example plotting script using *nctoolkit*, user is required to change *[PATH\_TO\_NETCDF\_FILE]* and *[VARIABLE]* to the exact location of the NEMO output and the variable to be plotted, respectively.

```

import nctoolkit as nc

# path to netCDF file
ff = [PATH_TO_NETCDF_FILE]
ds = nc.open_data(ff)

# removes colours from land values
ds.set_missing(0)

# draws the outline of the land
ds.fix_nemo_erssem_grid()

# selects the final timestep for plotting
ds.select(time=[len(ds.times)-1])

# plots the variable. Note, the colour bar will change for each plot since autoscale
# is set to False
plot = ds.plot([VARIABLE], autoscale=False)

```

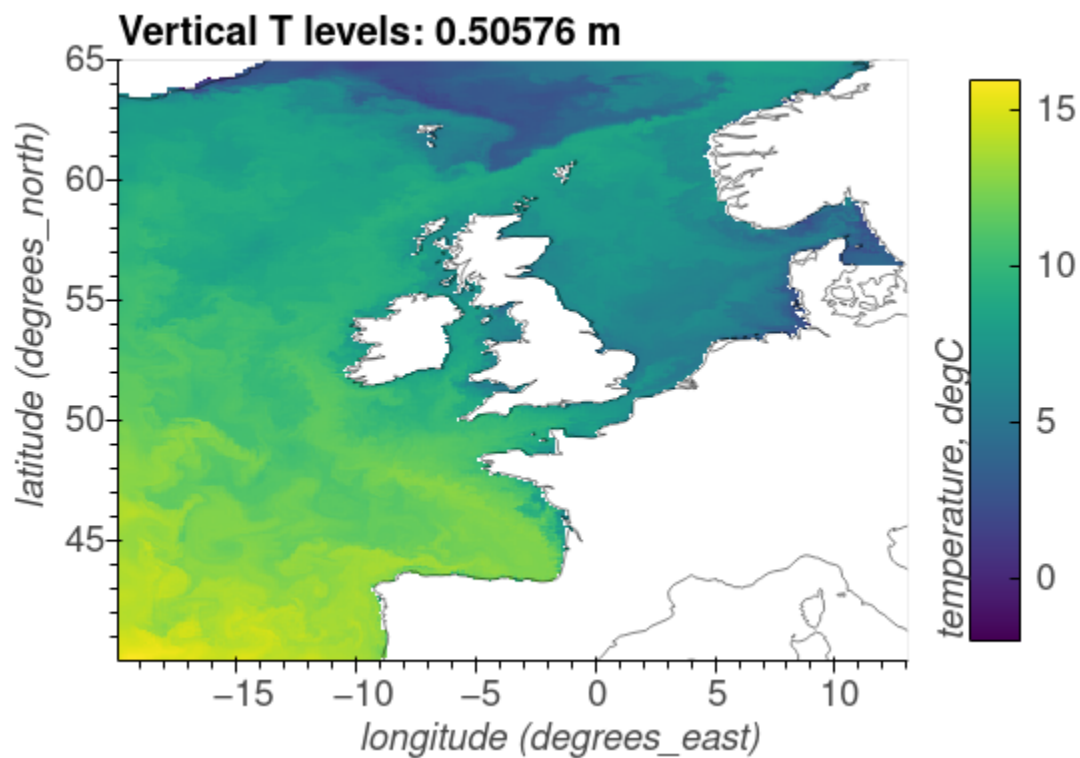
The following plots show the surface distributions of a subset of variables from the NEMO-ERSEM simulation on the AMM7 domain.

---

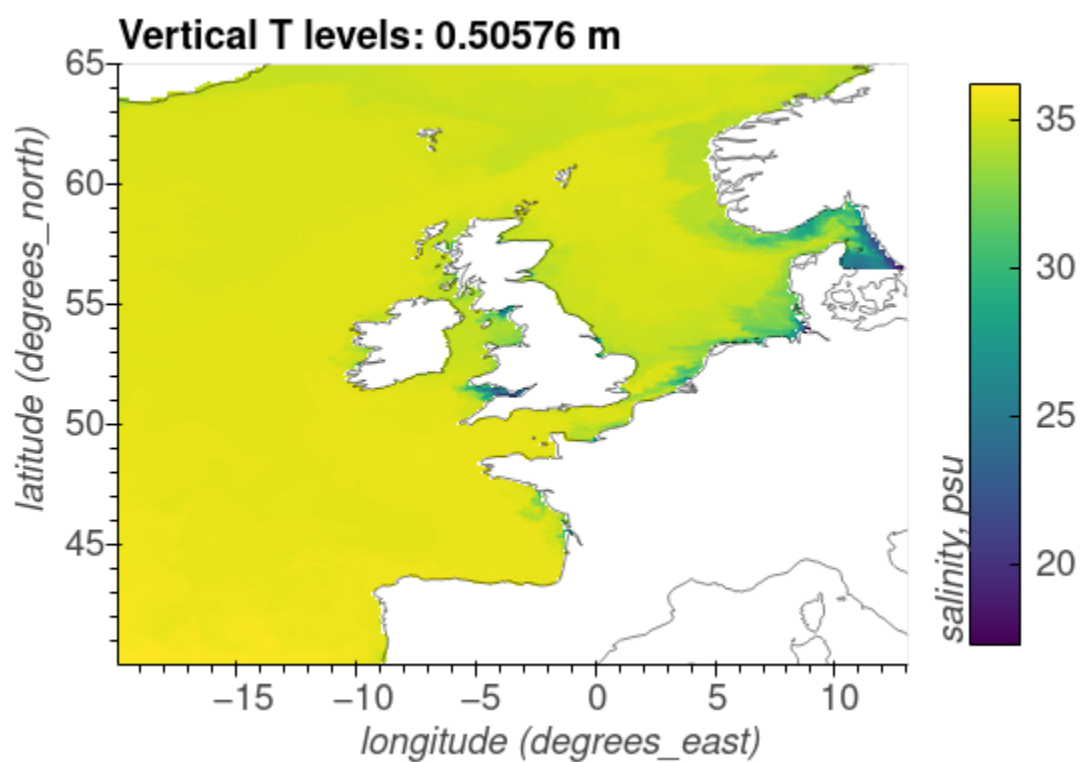
**Note:** The plots below are snapshots take from the model output at 30/01/2005.

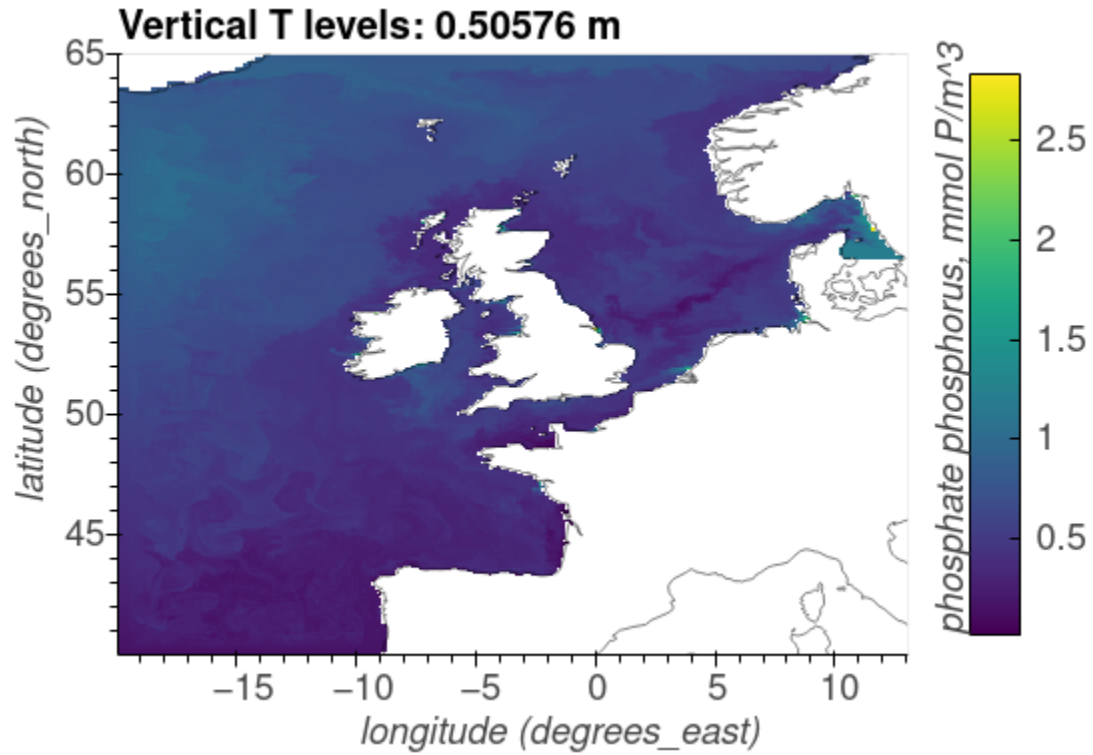
---

Potential temperature, degC

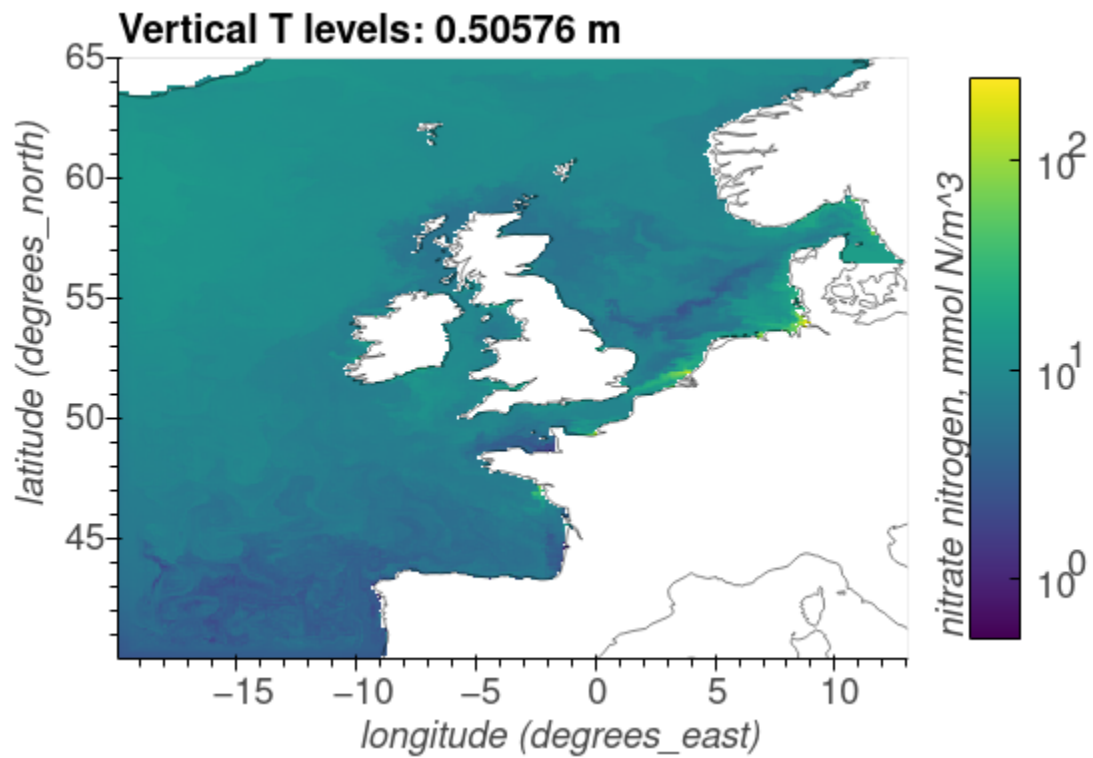


Salinity, psu

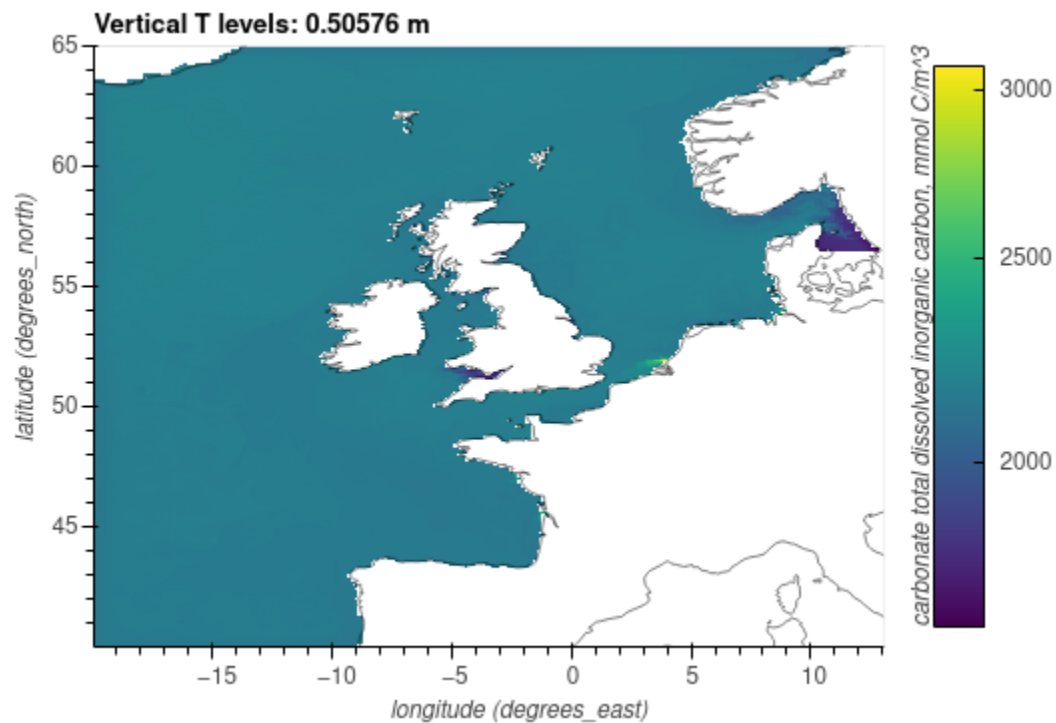
Phosphate phosphorus, mmol P/m<sup>3</sup>



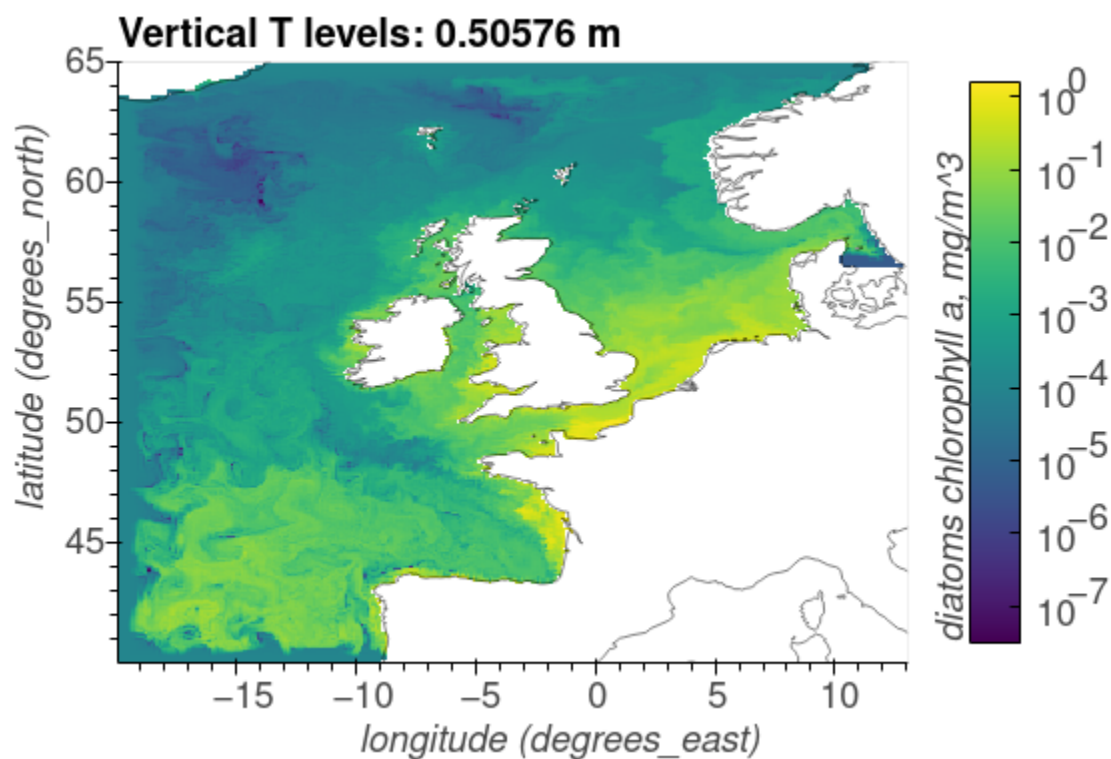
Nitrate nitrogen,  $\text{mmol N/m}^3$



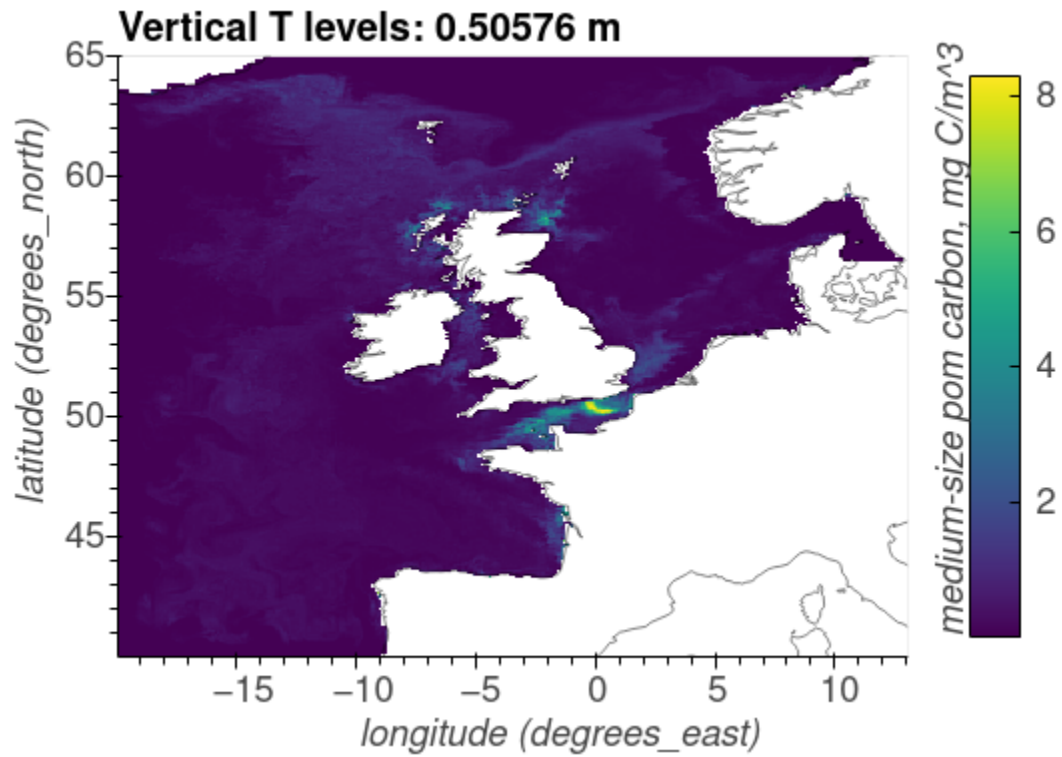
Carbonate total dissolved inorganic carbon,  $\text{mmol C/m}^3$



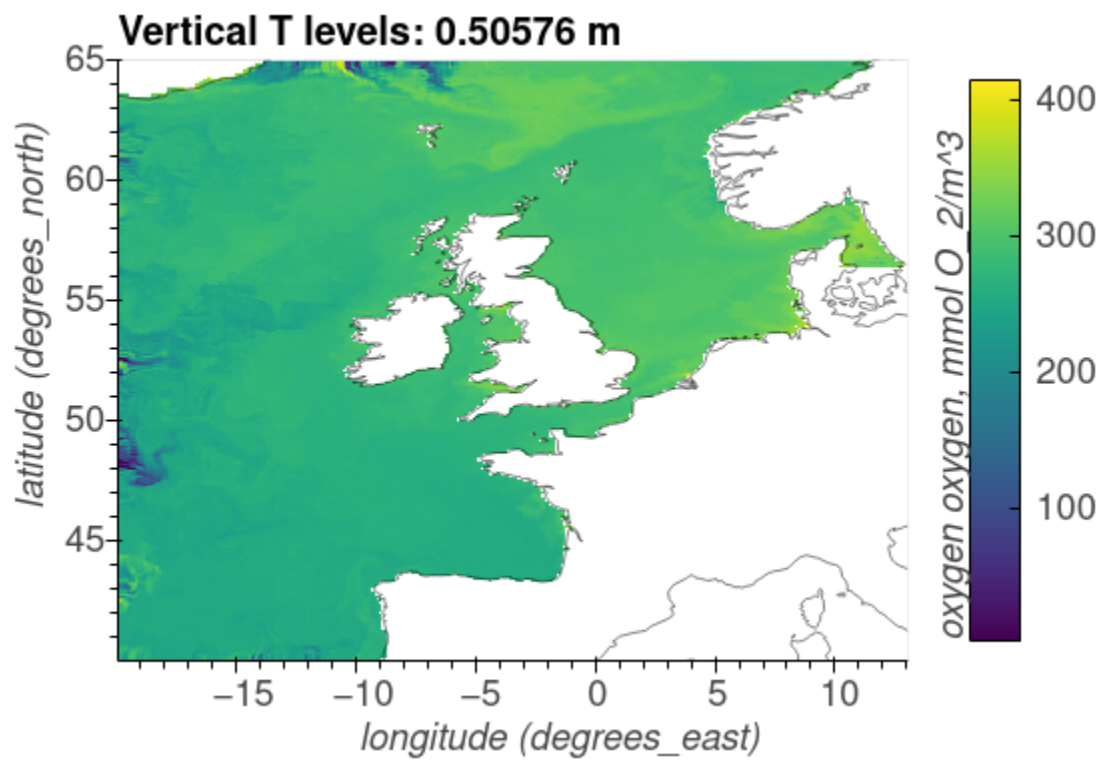
Diatoms chlorophyll, mg/m<sup>3</sup>



Medium-sized POM carbon, mg C/m<sup>3</sup>



Oxygen,  $\text{O}_2/\text{m}^3$



## 2.2.6 NEMO-ERSEM in a water column

This tutorial demonstrates how to configure and run NEMO 1D (water-column) reference configuration C1D\_PAPA with ERSEM biogeochemistry. This example is a demonstration of the concept, as it combines hydrodynamics of ocean station PAPA in the North Pacific Ocean with biogeochemistry of L4 station in the Western English Channel. Users are encouraged to modify this example to fit their own purposes.

We recommend to read the brief description of the NEMO C1D\_PAPA configuration before continuing with this tutorial.

### Step 1: Obtaining the code

The input data contained in INPUTS\_C1D\_PAPA\_v4.0.tar file must be obtained from NEMO Reference configurations inputs repository on Zenodo and unpacked into the working directory.

NEMO4 code base with FABM support can be obtained in the corresponding repository. The next step is to download FABM and ERSEM. Finally, I/O server XIOS-2.5 must be downloaded and installed. The compiled xios\_server.exe executable should be copied into the working directory.

### Step 2: Compiling the code

First, FABM must be compiled with ERSEM support, specifying nemo as a physical host model. The following commands can be run in a command line or wrapped into an executable file for easy recompilation at any future point:

```
old=$(pwd)                #remember current (working) directory
mkdir -p ~/build/nemo-fabm-ersem # create directory for the build
cd ~/build/nemo-fabm-ersem    # go to the build directory
cmake <FABM_DIR> -DFABM_HOST=nemo -DFABM_ERSEM_BASE=<ERSEM_DIR> -DCMAKE_
↳INSTALL_PREFIX=~/local/fabm/nemo-fabm-ersem
#replace <FABM_DIR> and <ERSEM_DIR> with the corresponding directories the
↳FABM and ERSEM code bases were downloaded to.
make -j4
make install
cd $old                    # return to the working directory
```

Thereafter, it is time to compile the NEMO executable. Users may refer to the C1D\_PAPA\_FABM\_ERSEM configuration provided with the NEMO4.0-FABM in cfmts directory. The critical point is to specify all the necessary compilation keys in cpp\_X.fcm file, i.e. key\_c1d for compilation in 1D, and key\_fabm for FABM support:

```
bld::tool::fppkeys    key_c1d key_mpp_mpi key_iomput key_nosignedzero key_top_
↳key_fabm
```

Next, compile the model by executing the following lines:

```
#!/bin/bash

NEMO_BUILD_DIR=$<NEMO_DIR>
RUN_DIR=$<RUN_DIR>
export XIOS_HOME=$<XIOS_DIR>
export FABM_HOME=$HOME/local/fabm/nemo-fabm-ersem
# replace <NEMO_DIR> and <XIOS_DIR> with the path to the corresponding code_
↳bases and <RUNDIR> with the working directory. FABM_HOME in this example_
↳corresponds to the directory where FABM-ERSEM was installed.
```

(continues on next page)



(continued from previous page)

```

ARCH=GCC_P MPC # specify build architecture

cd $NEMO_BUILD_DIR
./makenemo -m $ARCH -r C1D_PAPA_FABM_ERSEM -n C1D_PAPA_FABM_ERSEM_BLD_SCRATCH_
→ | tee compile.log
mv $NEMO_BUILD_DIR/cfgs/C1D_PAPA_FABM_ERSEM_BLD_SCRATCH/BLD/bin/nemo.exe $RUN_
→ DIR/
echo "Done."

```

The script above will compile the model and move the nemo executable into the working directory.

Note that the architecture file might require some editing depending on the machine the model is compiled and run on. This will include compiler version, compiler flags and links to netCDF libraries. Here, we are pointing our compilation to arch-GCC\_P MPC.fcm file (available within NEMO4.0-FABM repository) to compile on a typical PML workstation running Fedora Linux distribution and using a GNU Fortran compiler.

### Step 3: Getting ready to run the model

Within the working directory, create a link to the desired model configuration file. For this tutorial, we will link fabm.yaml provided with the cfgs/C1D\_PAPA\_FABM\_ERSEM configuration:

```

ln -sf <NEMO_DIR>/cfgs/C1D_PAPA_FABM_ERSEM/fabm.yaml.erscm_c1d fabm.yaml
# replace <NEMO_DIR> with the directory containing the ERSEM code.

```

ERSEM requires some external inputs, which must be provided. The following lines have been appended to the fabm.yaml file we are using (for simplicity we are using constant parameter values here). Depending on the configuration, the list of required external inputs will vary.

```

pco2a:
  model: horizontal_constant
  parameters:
    value: 400.
    standard_name: mole_fraction_of_carbon_dioxide_in_air
ADY_0:
  model: horizontal_constant
  parameters:
    value: 1.0e-10
    standard_name: gelbstoff_absorption_satellite

```

Create links to, or copy namelist files from NEMO cfgs/C1D\_PAPA\_FABM\_ERSEM folder into the working directory. Repeat the same procedure for \*.xml files. file\_def\_nemo.xml defines which outputs will be saved, and at what frequency. For the purpose of this example, we will save a range of daily averaged pelagic and benthic state and diagnostic variables. This file can be used as a template to specify the desired range of model outputs.

## Step 4: Running the model and visualising the outputs

The model is deployed by running the executable file in the working directory:

```
./nemo.exe
```

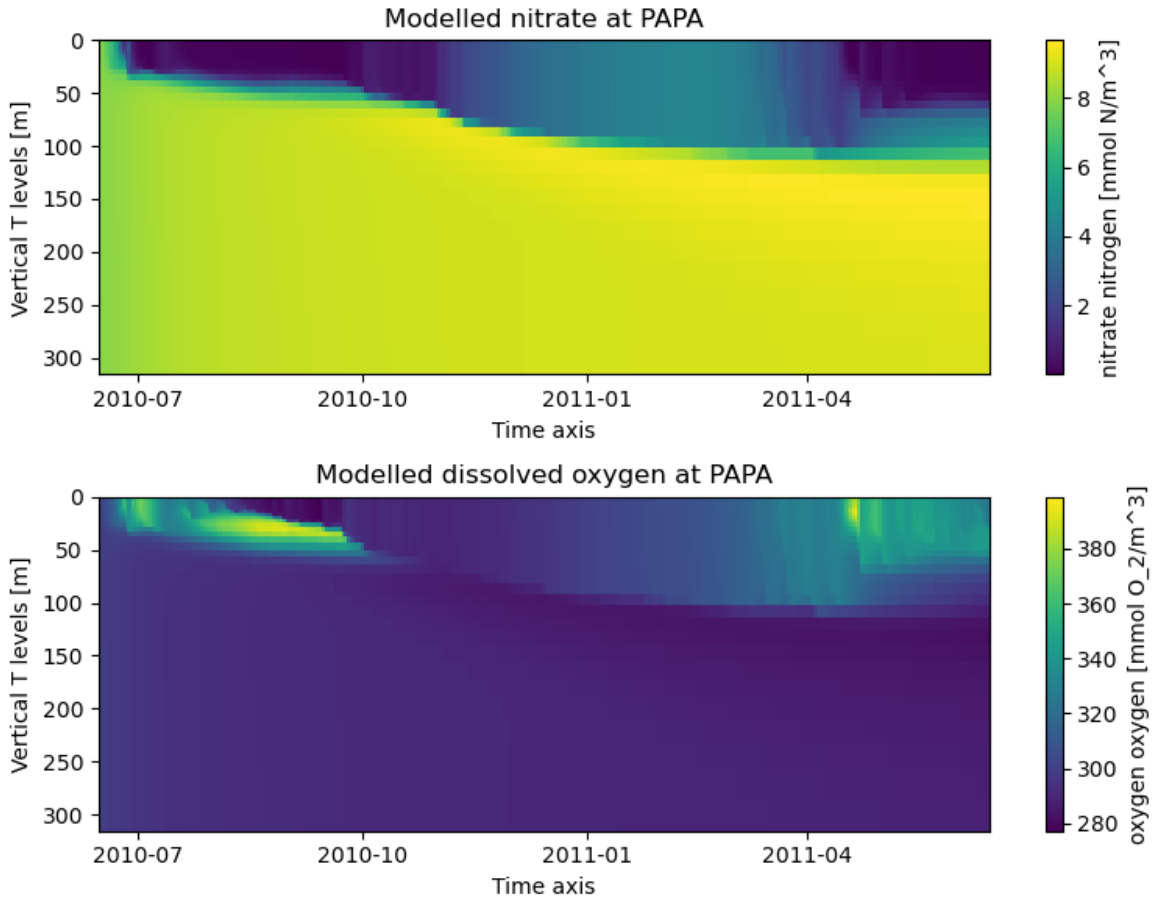
As a result, several output files will be generated according to specifications in file\_def\_nemo.xml file. Additionally, NEMO will generate restart files. The outputs can be visualised using netCDF viewer (e.g. [ncview](#) or [PyNcView](#)), or in Python using [xarray](#):

```
import xarray as xr                                # import_
↳xarray
import matplotlib.pyplot as plt                    # import_
↳matplotlib
dat = xr.open_dataset('C1D_PAPA_1d_20100615_20110614_ptrc_T.nc') # open_
↳dataset with ERSEM variables
fig, axes = plt.subplots(nrows=2, figsize=(8, 6))    # create_
↳figure with 2 subplots
dat.N3_n[:, 0:35, 1, 1].plot(x='time_counter', yincrease=False, ax=axes[0]) # plot_
↳Hovmöller diagram for nitrate
dat.O2_o[:, 0:35, 1, 1].plot(x='time_counter', yincrease=False, ax=axes[1]) # plot_
↳Hovmöller diagram for oxygen
axes[0].set_title('Modelled nitrate at PAPA')        # add title
axes[1].set_title('Modelled dissolved oxygen at PAPA') # add_
↳another title
fig.tight_layout()                                # make it_
↳look nice
```

In the example above we plot the variables over the entire modelled period, but only in the upper 35 vertical layers (from surface down to ~300 m depth). We also have to specify spatial dimensions of the output, as 1D configuration in NEMO actually has horizontal resolution of 3x3 grid points.

The resulting figure should look like this:





## 2.2.7 conda installation

We recommend using conda to obtain the latest version of `ersem`. The conda package includes interfaces for *Python front-end (pyfabm)*, *fabm0d: ERSEM in an aquarium* and *GOTM: ERSEM in a water column* tutorials, and can be installed as follows:

```
conda create -n ersem-tut -y
conda activate ersem-tut
conda install -c pmlmodelling ersem -y
```

To install `ersem` from source please look at the [developers documentation](#).

## 2.3 Tips and tricks/troubleshooting

### 2.3.1 General:

- `cmake` autodetects a suitable Fortran compiler. If a Fortran compiler is not found or you are unhappy with the compiler `cmake` selected, you can override this by providing `cmake` with `-DCMAKE_Fortran_COMPILER=<COMPILER_EXECUTABLE>`. Note: if you change `CMAKE_Fortran_COMPILER` (or provide it for the first time, while you previously ran `cmake` without), you need to empty your build directory first!

- When building GOTM or FABM's 0d driver, cmake will try to auto-detect NetCDF using `nf-config`. If `nf-config` is not present on your system, you'll need to provide cmake with the path(s) to the NetCDF include directories (`-DNetCDF_INCLUDE_DIRS=<PATH>`) and the path(s) to the NetCDF libraries (`-DNetCDF_LIBRARIES=<PATH>`). If you need to provide multiple paths to these variables, the individual paths should be separated by semi-colons. A common reason to use this: On some systems, `nf-config` does not detect where `netcdf.mod` is installed, which means you have to tell cmake by adding `-DNetCDF_INCLUDE_DIRS=<netcdf.mod_DIR>`. For instance, when using gfortran on Fedora, `<netcdf.mod_DIR>` can be `/usr/lib64/gfortran/modules`. In that case you usually do not need to provide `-DNetCDF_LIBRARIES=<PATH>`. Also, on some systems (like in the current LTS release of Ubuntu), `nf-config` is not included in the NetCDF packages. In this case, you can use `nc-config`: specify `-DNetCDF_CONFIG_EXECUTABLE=<PATH_TO_nc-config>` when calling cmake, or create a link to `nc-config` somewhere in your default path, to get auto-detection working.
- By default, cmake will select the "release" build type, which creates an executable without debugging information. If you want to compile in debug mode, specify `-DCMAKE_BUILD_TYPE=debug` in the call to cmake. When using gfortran, you may also want to add `-DCMAKE_Fortran_FLAGS_DEBUG=-fcheck=bounds` to catch out-of-bounds array access.
- To see the settings you specified when you ran cmake, and to selectively make changes to these settings, you can run `ccmake .` in your build directory.
- To speed up compilation, you can perform a parallel build by providing the `-j N` switch to `make` (not `cmake`!), with `N` being the number of cores you want to use.

Specific platforms:

- **ARCHER**: make sure to first load the `cmake` and `cray-netcdf` modules (`module load cmake cray-netcdf`). These enable cmake and autodetection of NetCDF paths, respectively. Provide `-DCMAKE_Fortran_COMPILER=ftn` to cmake to make sure cmake uses Cray's compiler wrapper script for Fortran compilation. Compilation should succeed with all three programming environments (intel, gnu, cray).
- **Ubuntu LTS**: `nf-config` is not included in the `netcdf` packages, but `nc-config` is. Specify `-DNetCDF_CONFIG_EXECUTABLE=<PATH_TO_nc-config>` when calling cmake, or create a link from `nf-config` to `nc-config` somewhere in your default path to get auto-detection of NetCDF paths working.
- **Fedora**: `nf-config` does not detect where `netcdf.mod` is installed (typically in `/usr/lib64/gfortran/modules`). This means you have to tell cmake by adding `-DNetCDF_INCLUDE_DIRS=/usr/lib64/gfortran/modules`.

## 2.3.2 Migrating from an earlier ERSEM version

If you were using an earlier release of ERSEM, you can update your old ERSEM configuration (`fabm.yaml`) to the latest by running the Python script `<SOURCEDIR>/ersem/testcases/update.py` with one argument: the path to your old `fabm.yaml` file.

This script requires a recent version of Python 2 or 3 and the [PyYAML package](#). Ideally, you also have the latest version of *the Python front end to FABM-ERSEM* installed; this enables the update script to clean up the `yaml` file and add documentation to it.

## 2.4 Developers: installation from source

### 2.4.1 pyfabm-ersem

To build the python driver and use it with ERSEM, you must first obtain copies of the FABM and ERSEM codes. The latest version of each can be obtained by cloning their respective code repositories using [git](#). The instructions of how to do this are found [here](#).

To run the install script you will need to have `wheel` and `numpy` installed. This can be done via:

```
1  #!/bin/bash
2
3  echo "Installing numpy and wheel"
4  python -m pip install wheel numpy
```

To install PyFABM-ERSEM, we suggest you use the following script below

```
1  #!/bin/bash
2  BRANCH=${1:-master}
3
4  echo "Cloning ERSEM"
5  git clone https://github.com/pmlmodelling/ersem.git
6
7  echo "Cloning FABM"
8  git clone https://github.com/fabm-model/fabm.git
9
10 echo "Checking out branch: $BRANCH"
11 cd ersem && git checkout $BRANCH && cd ..
12
13 echo "Building PyFABM-ERSEM"
14 mkdir build && cd build
15 cmake ../fabm/src/drivers/python -DFABM_ERSEM_BASE=../ersem
16 make install
```

### 2.4.2 fabm0d-ersem

To run the install script, you will need to have `netCDF` installed. An example of how to do this is here:

```
1  #!/bin/bash
2
3  echo "Installing netCDF"
4  sudo apt update
5  sudo apt install libnetcdf-dev
```

To install FABM0d-GOTM-ERSEM we suggest you use the following script below

```
1  #!/bin/bash
2
3  BRANCH="master"
4  CMAKE_FLAG=""
5  while getopts ":b:f:" flag; do
6      case "${flag}" in
```

(continues on next page)

(continued from previous page)

```

7         b) BRANCH=${OPTARG};;
8         f) CMAKE_FLAG=${OPTARG};;
9     esac
10 done
11
12 CPU="$(nproc)"
13
14 echo "Cloning ERSEM"
15 git clone https://github.com/pmlmodelling/ersem.git
16
17 echo "Cloning FABM"
18 git clone https://github.com/fabm-model/fabm.git
19
20 echo "Cloning GOTM"
21 git clone https://github.com/gotm-model/code.git gotm
22 cd gotm && git checkout v6.0 && git submodule update --init --recursive && cd ..
23
24 echo "Checking out branch: $BRANCH"
25 cd ersem && git checkout $BRANCH && cd ..
26
27 echo "Building FABM-GOTM-ERSEM"
28 mkdir build && cd build
29 cmake ../fabm/src/drivers/0d -DGOTM_BASE=../gotm -DFABM_ERSEM_BASE=../ersem $CMAKE_FLAG
30 make install -j $CPU

```

### 2.4.3 gotm-ersem

To run the install script, you will need to have netCDF installed. An example of how to do this is here:

```

1 #!/bin/bash
2
3 echo "Installing netCDF"
4 sudo apt update
5 sudo apt install libnetcdf-dev

```

To install GOTM-FABM-ERSEM, we suggest you use the following script below

```

1 #!/bin/bash
2 BRANCH="master"
3 CMAKE_FLAG=""
4 while getopts ":b:f:" flag; do
5     case "${flag}" in
6         b) BRANCH=${OPTARG};;
7         f) CMAKE_FLAG=${OPTARG};;
8     esac
9 done
10
11 CPU="$(nproc)"
12
13 echo "Cloning ERSEM"
14 git clone https://github.com/pmlmodelling/ersem.git

```

(continues on next page)

(continued from previous page)

```

15
16 echo "Cloning FABM"
17 git clone https://github.com/fabm-model/fabm.git
18
19 echo "Cloning GOTM"
20 git clone https://github.com/gotm-model/code.git gotm
21 cd gotm && git checkout v6.0 && git submodule update --init --recursive && cd ..
22
23 echo "Checking out branch: $BRANCH"
24 cd ersem && git checkout $BRANCH && cd ..
25
26 echo "Building GOTM-FABM-ERSEM"
27 mkdir build && cd build
28 cmake ../gotm -DFABM_BASE=../fabm -DFABM_ERSEM_BASE=../ersem $CMAKE_FLAG
29 make install -j $CPU

```

If you experience NetCDF issues when running `make install`, see *tips and tricks/troubleshooting*.

Now you should have a GOTM executable with FABM and ERSEM support at `~/local/gotm/bin/gotm`.

## 2.4.4 Obtaining source code

To get the ERSEM and FABM source codes:

```

git clone https://github.com/pmlmodelling/ersem.git
git clone https://github.com/fabm-model/fabm.git

```

This locally creates `ersem` and `fabm` directories with source code.

For water column models, you will need the GOTM source code:

```

git clone --recurse-submodules https://github.com/gotm-model/code.git gotm

```

To build the code, you will need: \* a recent Fortran compiler \* `cmake` 3.0 or higher. First check whether you have that installed: run `cmake --version` on the command line.

---

**Note:** It is not necessary to use `pip` or `apt` to install dependences. We suggest that `conda` or `brew` would also work well.

---

## 2.5 ERSEM Workflow

As a rule of thumb, ERSEM developments should be made in the [public ERSEM](#) with the exception of specific project developments.

### 2.5.1 Issue tracking

All developments, either private or public, are required to create an [issue/issues](#) describing the problem or development required.

For larger project developments that require multiple issues, it is often preferable to create a [milestone](#) which you can then assign multiple issues too.

### 2.5.2 Private developments

Private developments are those generated within new and ongoing projects that requires significant scientific changes to the code that will be initially isolated from the public version of the code

#### Workflow for private developments

The workflow for private developments is as follows:

1. Create an issue or issues in the [public ERSEM](#) repo based on the development required.
2. Create a [branch](#) from the [private ERSEM](#) repo. The naming convention of the branch is as follows `XX-[description]` where `XX` is the issue ID (this can be found after creation of the issue) and `[description]` is a one- or two- word description to identify the branch with the corresponding issue.
3. Commit all changes outlined in the issue to this branch and push branch to the private ERSEM repo.
4. Create [pull request](#) to merge the branch you have made the changes on into the *master* branch. In the creation of the pull request you will need to assign specific people to review the changes you have made before it can be merged into the master branch.
5. Once the updates have been approved by the reviewer/reviewers they will merge the development into the master branch.
6. When the private master branch is updated you will need to talk to the ERSEM repo manager to merge those changes back to the public repo.

---

**Note:** When creating the pull request it is important to add *Fixes #[issue-id-number]*, this will insure that when the pull request is merged into master the corresponding issue will be closed at the same time.

---

### 2.5.3 Public developments

The following changes are considered public developments and would need to follow a slightly different workflow (see below):

1. Documentation - new descriptions and updates to current documentation, these include new ERSEM tutorials or fixes to current docs
2. Bug fixes - these require immediate attention when identified within the code base
3. Public project developments - if possible, small developments and additions to the code shall be added publicly, these could include different options or parameterisations of the model.

## Workflow for public developments

For public developments, the workflow is pretty simple:

1. Create an issue or issues in the [public ERSEM](#) repo based on the development required.
2. Create a [branch](#) from the [public ERSEM](#) repo. The naming convention of the branch is as follows `XX-[description]` where `XX` is the issue ID (this can be found after creation of the issue) and `[description]` is a one- or two- word description to identify the branch with the corresponding issue.
3. Commit all changes outlined in the issue to this branch and push branch to the ERSEM repo.
4. Create [pull request](#) to merge the branch you have made the changes on into the *master* branch. In the creation of the pull request you will need to assign specific people to review the changes you have made before it can be merged into the master branch.
5. Once the updates have been approved by the reviewer/reviewers they will merge the development into the master branch.
6. When the public master branch is updated you will need to talk to the ERSEM repo manager to merge those changes back to the private repo.

---

**Note:** When creating the pull request it is important to add *Fixes #[issue-id-number]*, this will insure that when the pull request is merged into master the corresponding issue will be closed at the same time.

---

### 2.5.4 Notes for reviewers

When [reviewing pull requests](#) there are a few things to check:

1. Ensure that the new changes do not adversely affect ERSEM. **Note** some modifications will change the results produced by the model, however, it is the author of the changes and reviewer's responsibility to decide whether these changes go into ERSEM. If unsure, please contact the ERSEM repository manager.
2. Ensure that the changes made to the code correctly address the issue created and that *Fixes #[issue-id-number]* is added to the pull request description to ensure the relevant issue is closed upon merging with master.
3. For the public ERSEM, make sure that the tests continue to pass.

---

**Note:** Any questions about the workflow, please contact the ERSEM repository manager.

---

## 2.6 ERSEM module index

### 2.6.1 Module name: `ersem_primary_producer`

**Include:**

```
"fabm_driver.h"
```

**Class inheritance:**

```
type_ersem_pelagic_base
```

**Description:** Silicate use (P1) is optional - it is activated by setting the “use\_Si” flag in the run-time configuration.

Calcification (P2) is optional - it is activated by setting the “calcify” flag in the run-time configuration. Preprocessor symbol CALC is no longer used.

The model can be switched from a constant ratio between labile and semi- labile dissolved organic matter production to a dynamic ratio by setting the “docdyn” flag in the run-time configuration. Preprocessor symbol DOCDYN is no longer used.

Iron use is controlled by use\_iron defined in ersem/shared.F90.

**Functions:**

```
function get_sinking_rate(self,e__arguments_local_e) result (sd)
  real(kind=rk) :: sd
```

### 2.6.2 Module name: ersem\_benthic\_bacteria

**Include:**

```
"fabm_driver.h"
```

**Class inheritance:**

```
type_erssem_benthic_base
```

**Description:**

### 2.6.3 Module name: ersem\_carbonate

**Include:**

```
"fabm_driver.h"
```

**Class inheritance:**

```
type_base_model
```

**Description:**

**Functions:**

```
function approximate_alkalinity(iswtalk,t,s) result (ta)
  integer, intent(in) :: iswtalk
  real(kind=rk), intent(in) :: t
  real(kind=rk), intent(in) :: s
  real(kind=rk) :: ta
```



### 2.6.4 Module name: ersem\_light

**Include:**

```
"fabm_driver.h"
```

**Class inheritance:**

```
type_base_model
```

**Description:**

### 2.6.5 Module name: ersem\_mesozooplankton

**Include:**

```
"fabm_driver.h"
```

**Class inheritance:**

```
type_ersm_pelagic_base
```

**Description:**

### 2.6.6 Module name: ersem\_benthic\_carbonate

**Include:**

```
"fabm_driver.h"
```

**Class inheritance:**

```
type_base_model
```

**Description:**

### 2.6.7 Module name: ersem\_benthic\_erosion

**Include:**

```
"fabm_driver.h"
```

**Class inheritance:**

```
type_base_model
```

**Description:** type\_ersm\_benthic\_erosion

This model estimates sediment erosion in m/d from the bottom shear stress.

### 2.6.8 Module name: ersem\_bacteria\_docdyn

**Include:**

`"fabm_driver.h"`

**Class inheritance:**

`type_ersm_pelagic_base`

**Description:**

### 2.6.9 Module name: ersem\_benthic\_calcite

**Include:**

`"fabm_driver.h"`

**Class inheritance:**

`type_ersm_benthic_base`

**Description:** Benthic variable that supports resuspension and remineralization. Both processes return material to the pelagic.

default: all is private.

### 2.6.10 Module name: ersem\_benthic\_column

**Include:**

`"fabm_driver.h"`

**Class inheritance:**

`type_base_model`

**Description:** type\_ersm\_benthic\_column

This model specifies the structure of the three-layer sediment column.

This model also computes the diffusivity of solutes in the different layers, and a “particulate diffusivity” that represents bioturbation. These variables account for variable bioturbation and bioirrigation activity, respectively.

### 2.6.11 Module name: ersem\_light\_iop

**Include:**

```
"fabm_driver.h"
```

**Class inheritance:**

```
type_base_model
```

**Description:**

### 2.6.12 Module name: ersem\_calcification

**Include:**

```
"fabm_driver.h"
```

**Class inheritance:**

```
type_ersm_pelagic_base
```

**Description:**

### 2.6.13 Module name: ersem\_microzooplankton

**Include:**

```
"fabm_driver.h"
```

**Class inheritance:**

```
type_ersm_pelagic_base
```

**Description:**

### 2.6.14 Module name: ersem\_benthic\_nitrogen\_cycle

**Include:**

```
"fabm_driver.h"
```

**Class inheritance:**

```
type_base_model
```

**Description:** Model for nitrogen cycle

### 2.6.15 Module name: ersem\_benthic\_base

**Include:**

```
"fabm_driver.h"
```

**Class inheritance:**

```
type_particle_model
```

**Description:** Benthic variable that supports resuspension and remineralization. Both processes return material to the pelagic.

default: all is private.

### 2.6.16 Module name: ersem\_pelagic\_base

**Include:**

```
"fabm_driver.h"
```

**Class inheritance:**

```
type_particle_model
```

**Description:** default: all is private.

**Functions:**

```
function get_sinking_rate(self,e__arguments_local_e) result (rm)  
  real(kind=rk) :: rm
```

### 2.6.17 Module name: ersem\_light\_iop\_ady

**Include:**

```
"fabm_driver.h"
```

**Class inheritance:**

```
type_base_model
```

**Description:**

## 2.6.18 Module name: ersem\_benthic\_column\_particulate\_matter

### Include:

```
"fabm_driver.h"
surface flux $-D C0 z/z_mean d/dz exp(-z/z_mean) = 0$, as well as non-
↪ zero bottom flux
```

### Class inheritance:

```
type_erssem_benthic_base
```

**Description:** Particulate organic matter with idealized [exponential] profile.

The concentration of matter, in mass per unit sediment volume, is assumed to be an exponential function of depth  $z$ :

$$C(z) = C_0 \exp(-b \cdot z)$$

What value do constants  $C_0$  and  $b$  take?

Let's define the mean depth of matter within entire sediment column (depth range  $0$  to  $\infty$ ):

$$z_{\text{mean}} = \int_0^{\infty} z C(z) dz / \int_0^{\infty} C(z) dz$$

When we insert the exponential distribution of  $C(z)$ ,  $C_0$  drops out and we obtain

$$z_{\text{mean}} = \int_0^{\infty} z \exp(-b \cdot z) dz / \int_0^{\infty} \exp(-b \cdot z) dz$$

For the denominator of  $z_{\text{mean}}$  (concentration integrated from surface to infinite depth) we find through standard integration:

$$\int_0^{\infty} \exp(-b \cdot z) dz = [-1/b \exp(-b \cdot z)]_0^{\infty} = 1/b$$

Numerator  $\int_0^{\infty} z \exp(-b \cdot z) dz$  can be found through integration by parts: General rule:  $\int u v dz = u \int v dz - \int (u' \int v dz) dz$  We can apply this to find the antiderivative of  $z \exp(-b \cdot z)$ :

$$\int z \exp(-b \cdot z) dz = -z/b \exp(-b \cdot z) - 1/b^2 \exp(-b \cdot z) = -(z+1/b)/b \exp(-b \cdot z)$$

Verify by differentiation (apply chain rule):

$$-1/b \exp(-b \cdot z) + z \exp(-b \cdot z) + 1/b \exp(-b \cdot z) = z \exp(-b \cdot z) \text{ [OK]}$$

Integrating from  $0$  to  $\infty$  while assuming  $b > 0$  we obtain

$$\int_0^{\infty} z \exp(-b \cdot z) dz = [-(z+1/b)/b \exp(-b \cdot z)]_0^{\infty} = 1/b^2$$

Combining the expressions for the numerator and denominator of  $z_{\text{mean}}$ :

$$z_{\text{mean}} = (1/b^2)/(1/b) = 1/b$$

Thus, the exponential decay constant  $b$  is equal to  $1/z_{\text{mean}}$ , and

$$C(z) = C_0 \exp(-z/z_{\text{mean}})$$

NB this is a standard result (mean of the exponential distribution); the above derivation is given simply for completeness.

The integral of  $C(z)$  from  $0$  to the bottom of the modelled column,  $z_{\text{bot}}$ , should equal the modelled density of mass:

$$\int_0^{z_{\text{bot}}} C(z) dz = [-z_{\text{mean}} C_0 \exp(-z/z_{\text{mean}})]_0^{z_{\text{bot}}} = z_{\text{mean}} C_0 (1 - \exp(-z_{\text{bot}}/z_{\text{mean}})) = C_{\text{int}}$$

Thus, surface concentration  $C_0$  can be rewritten in terms of the depth-integrated concentration  $C_{\text{int}}$ , (integrated up to  $z_{\text{bot}}$ , not  $\infty$ )

$$C_0 = C_{\text{int}}/z_{\text{mean}}/(1-\exp(-z_{\text{bot}}/z_{\text{mean}}))$$

Note: Ebenhoeh et al. 1995 define the penetration depth as mean depth within the model domain, i.e., with lower boundary  $z_{\text{bot}}$  rather than  $\infty$ :

$$\begin{aligned} z_{\text{mean}}' &= \int_0^{z_{\text{bot}}} z C(z) dz / \int_0^{z_{\text{bot}}} C(z) dz = \frac{[-(z+1/b)/b \exp(-b*z)]_0^{z_{\text{bot}}} / [-1/b \exp(-b*z)]_0^{z_{\text{bot}}}}{[1/b^2-(z_{\text{bot}}+1/b)/b \exp(-b*z_{\text{bot}})] / [1/b-1/b \exp(-b*z_{\text{bot}})]} \\ &= 1/b [1-(z_{\text{bot}}/b+1) \exp(-b*z_{\text{bot}})] / [1-\exp(-b*z_{\text{bot}})] = \\ &= 1/b [1-z_{\text{bot}}/b \exp(-b*z_{\text{bot}})]/[1-\exp(-b*z_{\text{bot}})] \end{aligned}$$

This will NOT produce an explicit expression for  $b$  as a function of  $z_{\text{mean}}$ . Instead, Ebenhoeh et al. implicitly assume  $z_{\text{bot}} \gg z_{\text{mean}}$ , which leads to  $b \approx 1/z_{\text{mean}}$ . However, this approximation becomes invalid when  $z_{\text{mean}}$  becomes large, and that can happen in the model. To avoid this problem, we define penetration depth as the mean depth of mass between 0 and  $\infty$ . Since our exponential function remains the same, expressions based on that (e.g., impact of bioturbation) are the same as in the Oldenburg model. Expressions based on the interpretation of penetration depth (the mean depth of mass between 0 and  $\infty$  in our case, and between 0 and  $z_{\text{bot}}$  for the Oldenburg model) will differ slightly, as e.g. in the next section.

Sources and sinks change  $C(z)$ , and therefore also penetration depth

$$z_{\text{mean}} = \int_0^{\infty} z C(z) dz / \int_0^{\infty} C(z) dz$$

The time derivative of this expression is found by applying the chain rule

$$\frac{d}{dt} z_{\text{mean}} = [\int_0^{\infty} z \frac{d}{dt} C(z) dz - \int_0^{\infty} \frac{d}{dt} C(z) dz \int_0^{\infty} z C(z) dz] / \int_0^{\infty} C(z) dz$$

Introducing depth-integrated concentration

$$C_{\text{int}} = \int_0^{\infty} C(z) dz,$$

depth-integrated sources minus sinks

$$s_{\text{ms}} = \int_0^{\infty} \frac{d}{dt} C(z) dz,$$

and the mean depth of the sources minus sinks

$$z_{\text{sms}} = \int_0^{\infty} z \frac{d}{dt} C(z) dz / \int_0^{\infty} \frac{d}{dt} C(z) dz,$$

we can simplify this to

$$\frac{d}{dt} z_{\text{mean}} = (z_{\text{sms}} - z_{\text{mean}}) s_{\text{ms}} / C_{\text{int}}$$

It is worth noting that the final division is by the concentration integrated from surface to *infinite depth*,  $C_{\text{int}}$ :

$$C_{\text{int}} = z_{\text{mean}} C_0$$

That is not the same as  $C_{\text{int}}$ , i.e., the concentration integrated over the modelled depth interval ( $0$  to  $z_{\text{bot}}$ ), which equals

$$C_{\text{int}} = z_{\text{mean}} C_0 (1-\exp(-z_{\text{bot}}/z_{\text{mean}}))$$

In the original Oldenburg implementation, penetration depth is defined as the mean depth of mass between 0 and  $z_{\text{bot}}$  (not 0 to  $\infty$ ). As a result,  $C_{\text{int}}$  is substituted for  $C_{\text{int}}$ . One consequence of this difference is that penetration depth in the Oldenburg model can “run away” (tend to infinity), while that is prevented in our expression due to the additional multiplication of the rate of change with  $1-\exp(-z_{\text{bot}}/z_{\text{mean}})$ .

It is worth noting that the resulting expression for  $d/dt z_{\text{mean}}$  does NOT depend on distribution  $C(z)$ . That is, it is valid for ANY vertical distribution, exponential or otherwise.

Modelled as a diffusion process, bioturbation changes  $C(z)$ , and therefore also penetration depth

$$z\_mean = \int_0^\infty z C(z) dz / \int_0^\infty C(z) dz$$

Let us try this first for the denominator:

$$d/dt \int_0^\infty C(z) dz = \int_0^\infty d/dt C(z) dz$$

For  $d/dt C(z)$  we have the normal diffusion equation:

$$d/dt C(z) = d/dz(D d/dz C(z)) = d/dz(-D/z\_mean C_0 \exp(-z/z\_mean)) = D C_0/z\_mean^2 \exp(-z/z\_mean)$$

Inserting this expression we obtain for the change in depth-integrated mass:

$$d/dt \int_0^\infty C(z) dz = \int_0^\infty D C_0/z\_mean^2 \exp(-z/z\_mean) dz = [-D C_0/z\_mean \exp(-z/z\_mean)]_0^\infty = D C_0/z\_mean$$

However, we KNOW that diffusion within the column should not affect the mass integral. Why is this then non-zero? The reason for this is that we have not accounted for the no-flux boundary conditions. As a result, we are implicitly using a non-zero inward flux at the surface that is determined by the gradient:

$$-D d/dz C(0) = D C_0/z\_mean \exp(-z/z\_mean) = D C_0/z\_mean$$

In other words, to close the column for mass, we need to subtract this surface flux. (NB the gradient is zero at infinite depth, i.e., the bottom flux is zero)

With this knowledge, we can revisit the numerator in the change in penetration depth  $z\_mean$ . Its time derivative equals

$$d/dt \int_0^\infty z C(z) dz = \int_0^\infty z d/dt C(z) dz$$

From the diffusion equation for  $C(z)$  we derived  $d/dt C(z) = D C_0/z\_mean^2 \exp(-z/z\_mean)$ . Inserting this in  $d/dt z\_mean$ , while limiting bioturbation to maximum depth  $z\_tur$ , we obtain

$$d/dt \int_0^{z\_tur} z C(z) dz = D C_0/z\_mean^2 \int_0^{z\_tur} z \exp(-z/z\_mean) dz$$

For the expression within the integral we previously found antiderivative  $-(z+z\_mean)z\_mean \exp(-z/z\_mean)$ . Thus,

$$\int_0^{z\_tur} z \exp(-z/z\_mean) dz = [-(z+z\_mean)z\_mean \exp(-z/z\_mean)]_0^{z\_tur} = -(z\_tur+z\_mean)z\_mean \exp(-z\_tur/z\_mean) + z\_mean^2$$

Using this antiderivative to solve the integral from 0 to  $z\_tur$ :

$$d/dt \int_0^\infty z C(z) dz = D C_0 [1 - (z\_tur/z\_mean + 1)] \exp(-z\_tur/z\_mean)$$

$-D C_0 z\_tur/z\_mean \exp(-z\_tur/z\_mean)$ . Subtracting the latter we obtain:

$$d/dt \int_0^\infty z C(z) dz = D C_0 [1 - \exp(-z\_tur/z\_mean)]$$

Finally,  $d/dt z\_mean$  is given by the ratio of this expression to  $\int_0^\infty C(z) dz = C_0 z\_mean$ :

$$d/dt z\_mean = D/z\_mean [1 - \exp(-z\_tur/z\_mean)]$$

This describes how the penetration depth  $z\_mean$  changes due to bioturbation up to depth  $z\_tur$ , with the intensity of bioturbation described by diffusivity  $D$ . It is worth noting that  $d/dt z\_mean$  to infinity when  $z\_mean$  to 0. Thus, in the analytical solution of the model penetration depth cannot become negative as long as  $D > 0$ .

A change in penetration depth without an associated change in total mass  $C\_int\_infy$  will cause a change in the mass in the depth interval described by the model, i.e.,  $\int_0^{z\_bot} C_0 \exp(-z/z\_mean)$ . Inserting  $C_0 = C\_int\_infy/z\_mean$ :

$$\int_0^{z\_bot} C_0 \exp(-z/z\_mean) dz = C\_int\_infy/z\_mean \int_0^{z\_bot} \exp(-z/z\_mean) dz$$

Taking the time derivative, with only  $z\_mean$  depending on time ( $C\_int\_infy$  is constant

$$d/dt \int_0^{z_{\text{bot}}} C_0 \exp(-z/z_{\text{mean}}) dz = d/dt z_{\text{mean}} - C_{\text{int\_infy}}/z_{\text{mean}}^2 \int_0^{z_{\text{bot}}} \exp(-z/z_{\text{mean}}) dz$$

$$\bullet C_{\text{int\_infy}}/z_{\text{mean}} \int_0^{z_{\text{bot}}} z/z_{\text{mean}}^2 d/dt z_{\text{mean}} \exp(-z/z_{\text{mean}}) dz$$

For the first term (first line), we have:

$$d/dt z_{\text{mean}} - C_{\text{int\_infy}}/z_{\text{mean}}^2 \int_0^{z_{\text{bot}}} \exp(-z/z_{\text{mean}}) dz = -1/z_{\text{mean}} d/dt z_{\text{mean}} C_{\text{int}}$$

with  $C_{\text{int}}$  being the concentration integrated from surface to model bottom [not infinite depth]

We can rewrite the second term by moving constants outside the integral:

$$C_{\text{int\_infy}}/z_{\text{mean}}^3 d/dt z_{\text{mean}} \int_0^{z_{\text{bot}}} z \exp(-z/z_{\text{mean}}) dz$$

Now we can solve the integral with our previously found antiderivative:

$$\int_0^{z_{\text{bot}}} z \exp(-z/z_{\text{mean}}) dz = [-(z+z_{\text{mean}})z_{\text{mean}} \exp(-z/z_{\text{mean}})]_0^{z_{\text{bot}}} = -(z_{\text{bot}}+z_{\text{mean}})z_{\text{mean}} \exp(-z_{\text{bot}}/z_{\text{mean}}) + z_{\text{mean}}^2$$

Inserting this

$$C_{\text{int\_infy}}/z_{\text{mean}} d/dt z_{\text{mean}} [-(z_{\text{bot}}/z_{\text{mean}}+1) \exp(-z_{\text{bot}}/z_{\text{mean}}) + 1] = C_{\text{int\_infy}}/z_{\text{mean}} d/dt z_{\text{mean}} [1 - \exp(-z_{\text{bot}}/z_{\text{mean}}) - z_{\text{bot}}/z_{\text{mean}} \exp(-z_{\text{bot}}/z_{\text{mean}})]$$

Since  $C_{\text{int}} = C_{\text{int\_infy}} [1 - \exp(-z_{\text{bot}}/z_{\text{mean}})]$ , we can rewrite this as

$$1/z_{\text{mean}} d/dt z_{\text{mean}} C_{\text{int}} - C_{\text{int\_infy}}/z_{\text{mean}} d/dt z_{\text{mean}} z_{\text{bot}}/z_{\text{mean}} \exp(-z_{\text{bot}}/z_{\text{mean}})$$

Combining the two contributions to  $d/dt \int_0^{z_{\text{bot}}} C_0 \exp(-z/z_{\text{mean}}) dz$ , we are left with

$$d/dt \int_0^{z_{\text{bot}}} C_0 \exp(-z/z_{\text{mean}}) dz = -C_{\text{int\_infy}} z_{\text{bot}}/z_{\text{mean}}^2 \exp(-z_{\text{bot}}/z_{\text{mean}}) d/dt z_{\text{mean}}$$

JB 28/11/2014: this solution was validated in Python with sympy, using the following code

```
C_int_infy,z_bot,z,t = sympy.symbols('C_int_infy,z_bot,z,t',positive=True,real=True)
z_mean = sympy.Function('z_mean',positive=True,real=True)(t) C =
C_int_infy/z_mean*sympy.exp(-z/z_mean) dC_dt = sympy.diff(C,t) print
sympy.integrate(dC_dt,(z,0,z_bot),conds='none')
```

Inserting  $C_{\text{int\_infy}} = C_{\text{int}}/[1 - \exp(-z_{\text{bot}}/z_{\text{mean}})]$

$$d/dt \int_0^{z_{\text{bot}}} C_0 \exp(-z/z_{\text{mean}}) dz = -C_{\text{int}}/[1 - \exp(-z_{\text{bot}}/z_{\text{mean}})] z_{\text{bot}}/z_{\text{mean}}^2 \exp(-z_{\text{bot}}/z_{\text{mean}}) d/dt z_{\text{mean}}$$

Since  $C_0 = C_{\text{int}}/[1 - \exp(-z_{\text{bot}}/z_{\text{mean}})]/z_{\text{mean}}$ , this is equivalent to

$$d/dt \int_0^{z_{\text{bot}}} C_0 \exp(-z/z_{\text{mean}}) dz = C(z_{\text{bot}}) z_{\text{bot}}/z_{\text{mean}} d/dt z_{\text{mean}}$$

In the Oldenburg code, the final expression for the change in mass within model domain due to burial is

$$\bullet C_{\text{int}}/[1 - \exp(-z_{\text{bot}}/z_{\text{mean}})] [\exp(-(z_{\text{bot}} - d/dt z_{\text{mean}})/z_{\text{mean}} - \exp(-z_{\text{bot}}/z_{\text{mean}}))]$$

Both our expression and the Oldenburg one contain  $-C_{\text{int}}/[1 - \exp(-z_{\text{bot}}/z_{\text{mean}})] \exp(-z_{\text{bot}}/z_{\text{mean}})$ . A discrepancy remains in the scale factor applied to it:

$$\text{our derivation: } z_{\text{bot}}/z_{\text{mean}}^2 d/dt z_{\text{mean}} \quad \text{Oldenburg: } \exp(d/dt z_{\text{mean}}/z_{\text{mean}}) - 1$$

It's worth noting that the Oldenburg model here has a unit problem: the factor in the exponential is not dimensionless, but has unit  $\text{time}^{-1}$ .



This file contains two modules that can be instantiated by the user (from fabm.yaml):  
 type\_erssem\_benthic\_column\_particulate\_matter: describes particulate organic matter in terms of column-integrated mass and penetration depth  
 type\_erssem\_benthic\_pom\_layer: describes particulate organic matter within a user-specified depth interval

The idealized profile is defined only in terms of its density (quantity/m<sup>2</sup>) and its penetration depth. By assuming an exponential distribution of matter (constant positive concentration at sediment surface, tending to zero at infinite depth), these two variables suffice to specify the concentration profile.

From the idealized concentration profile, we first compute densities (quantity/m<sup>2</sup>) per layer. These layer-specific densities act as state variable: other modules can retrieve their value, but also provide their rate of change. We finally retrieve these rates of change, and convert them into a change in total depth-integrated matter *and* a change in penetration depth.

The “within a single depth interval” functionality is partitioned over two modules:

type\_erssem\_benthic\_pom\_layer computes the density of all constituents within the specified depth interval. This is the module that can be instantiated from fabm.yaml.

type\_constituent\_for\_single\_layer\_change collects interval-specific sink-source terms for a single constituent and translates those into the change in column-integrated matter and penetration depth. It is created automatically as a submodel of type\_erssem\_benthic\_pom\_layer and does not interact with the user.

The split of functionality across two modules is needed because the mass within the interval must be known early, so all other modules can use it, while the change in column-integrated mass and penetration depth must be computed late, after all other individual modules have computed interval-specific rates of change. Thus we have the dependency chain:

```
1 mass within the desired depth interval (type_erssem_benthic_pom_layer)
-> 2 interval-specific sink-source terms (all other modules) -> 3 change in column-
integrated mass and penetration depth (type_constituent_for_single_layer_change)
```

**Putting 1 and 3 in the same module creates a circular dependency [that’s BAD].** default: all is private.

Module for particulate organic matter class with idealized profile (e.g., Q6, Q7) Also handles the change in average penetration depth due to bioturbation, as well as burial in the form of organic matter moving beyond the bottom of the modelled column due to an increase in penetration depth. This module will always create a “surface” submodel for organic matter at the very surface of the sediment; this is typically used as target for sedimentation/resuspension.

#### Functions:

```
function partq(d_pen,d_top,d_bot,d_max)
  real(kind=rk), intent(in) :: d_pen
  real(kind=rk), intent(in) :: d_top
  real(kind=rk), intent(in) :: d_bot
  real(kind=rk), intent(in) :: d_max
```

### 2.6.19 Module name: `ersem_benthic_column_dissolved_matter`

**Include:**

```
"fabm_driver.h"
```

**Class inheritance:**

```
type_base_model
```

**Description:** depth-integrated mass in the benthic column depth-integrated [negative] mass below the zero-concentration isocline concentration in bottom-most pelagic cell sources-sinks specified for layer-integrated variables (pore water and adsorbed) sources-sinks specified for layer-integrated pore water concentration pelagic-benthic flux Flag specifying whether constituent is non-negative Model for dissolved matter in sediment, using idealized equilibrium profiles to determine pelagic-benthic diffusive flux, and to determine equilibrium depths of all but the last layer (NB layer depth will be relaxed to equilibrium value).

### 2.6.20 Module name: `ersem_nitrification`

**Include:**

```
"fabm_driver.h"
```

**Class inheritance:**

```
type_ersem_pelagic_base
```

**Description:**

### 2.6.21 Module name: `ersem_oxygen`

**Include:**

```
"fabm_driver.h"  
as part of utilization.
```

**Class inheritance:**

```
type_base_model
```

**Description:** Computes oxygen saturation, apparent oxygen utilization and handles exchange of oxygen across the water surface. Note: negative oxygen concentrations are permitted. These reflect an oxygen debt (e.g., presence of H<sub>2</sub>S) In this case, oxygen saturation will be zero (not negative while apparent oxygen utilization will still be the difference between saturation concentration and [negative] oxygen concentration.

**Functions:**

```
function oxygen_saturation_concentration(self,etw,x1x) result (osat)  
  real(kind=rk), intent(in) :: etw  
  real(kind=rk), intent(in) :: x1x  
  real(kind=rk) :: osat
```

## 2.6.22 Module name: ersem\_zenith\_angle

### Include:

```
"fabm_driver.h"
```

### Class inheritance:

```
type_base_model
```

**Description:** This module calculates the solar zenith angle as a function of longitude, latitude, day of year, and time of day. The code that performs this calculation has been adapted from the General Ocean Turbulence Model (GOTM, <http://www.gotm.net>). Original GOTM source file: src/airsea/solar\_zenith\_angle.F90.

## 2.6.23 Module name: ersem\_model\_library

### Include:

### Class inheritance:

```
type_base_model_factory
```

### Description:

## 2.6.24 Module name: ersem\_nitrous\_oxide

### Include:

```
"fabm_driver.h"
```

### Class inheritance:

```
type_ersm_pelagic_base
```

**Description:** This module calculates saturation concentrations and air-sea exchange of nitrous oxide. Production of nitrous oxide is implemented in nitrification module (nitrification.F90). For implementation and validation of nitrous oxide on the North-West European Shelf see Lessin et al. (2020): doi.org/10.1029/2019JG005613

### Functions:

```
function n2o_transfer_coefficient(self,etw,x1x) result (kon2o)
  real(kind=rk), intent(in) :: etw
  real(kind=rk), intent(in) :: x1x
  real(kind=rk) :: kon2o
```

### 2.6.25 Module name: ersem\_bacteria

**Include:**

`"fabm_driver.h"`

**Class inheritance:**

`type_erssem_pelagic_base`

**Description:**

### 2.6.26 Module name: ersem\_benthic\_fauna

**Include:**

`"fabm_driver.h"`

**Class inheritance:**

`type_erssem_benthic_base`

**Description:** To achieve compatibility with legacy ERSEM, we need to be able to decouple the variable from which food availability is derived from the variable that absorbs the loss due to gross food uptake. The following variables absorb the loss due to food uptake - by default they are coupled to the same variable from which available food is derived.

### 2.6.27 Module name: ersem\_fluff

**Include:**

`"fabm_driver.h"`

**Class inheritance:**

`type_erssem_benthic_base`

**Description:**

### 2.6.28 Module name: ersem\_fluff

**Include:**

`"fabm_driver.h"`

**Class inheritance:**

`type_erssem_benthic_base`

**Description:**

## 2.7 License

ERSEM is free software: you can redistribute it and/or modify it under the terms of the [GNU General Public License](#) as published by the [Free Software Foundation](#), either version 3 of the License, or (at your option) any later version. It is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. A copy of the license is provided in COPYING.

Copyright 2016-2021 Plymouth Marine Laboratory.

## 2.8 Acknowledgements

Over the years, ERSEM has been developed using funding from a wide variety of sources and projects. The current codebase was developed in the NERC/Defra Shelf Seas Biogeochemistry [Grant: \[NE/K001876/1\]](#) and Marine Ecosystems Research Programmes [Grant: \[NE/L003066/1\]](#). Today, ERSEM maintenance and development continues to be funded through the NERC Climate Linked Atlantic Sector Science programme [Grant \[NE/R015953/1\]](#) and a combination of UK Research and Innovation (UKRI) and European Research Council (ERC) funded research projects.

## 2.9 Bibliography



## BIBLIOGRAPHY

- [1] J.C. Blackford and P.J. Radford. A structure and methodology for marine ecosystem modelling. *Netherlands Journal of Sea Research*, 33(3-4):247–260, 1995. doi:[10.1016/0077-7579\(95\)90048-9](https://doi.org/10.1016/0077-7579(95)90048-9).
- [2] M. Butenschön, J. Clark, J. N. Aldridge, J. I. Allen, Y. Artioli, J. Blackford, J. Bruggeman, P. Cazenave, S. Ciavatta, S. Kay, G. Lessin, S. van Leeuwen, J. van der Molen, L. de Mora, L. Polimene, S. Sailley, N. Stephens, and R. Torres. Ersem 15.06: a generic model for marine biogeochemistry and the ecosystem dynamics of the lower trophic levels. *Geoscientific Model Development*, 9(4):1293–1339, 2016. doi:[10.5194/gmd-9-1293-2016](https://doi.org/10.5194/gmd-9-1293-2016).
- [3] R.F. Weiss. The solubility of nitrogen, oxygen and argon in water and seawater. *Deep Sea Research and Oceanographic Abstracts*, 17(4):721–735, 1970. doi:[10.1016/0011-7471\(70\)90037-9](https://doi.org/10.1016/0011-7471(70)90037-9).