MonGOOS-SEAMLESS workshop, 16 Nov 2023 preparation instructions

https://bit.ly/eat-mongoos

SEAMLESS Training workshop on "Biogeochemical Data assimilation with EAT"

National School of management of Tangier, Morocco, 16th November 2023

Welcome!

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101004032.

Stefano Ciavatta,
Mercator Ocean international
MEAP-TT co-chair
SEAMLESS Advisory Board

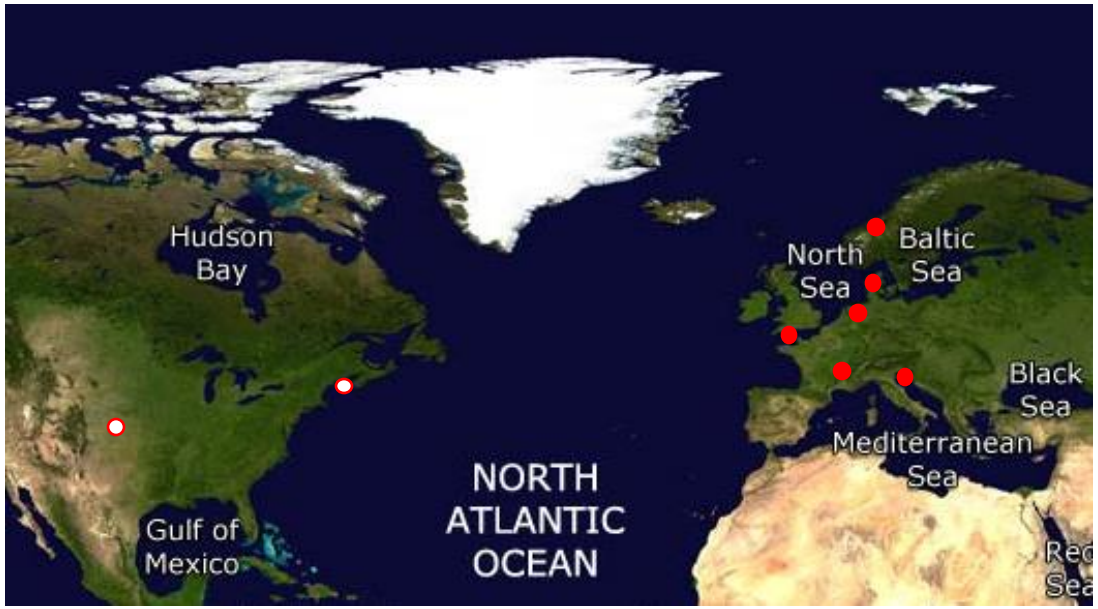
The SEAMLESS project: overview

Mission: improve the operational simulation of indicators related to climate impact, marine food-webs and stakeholders' needs

"Key Facts":

- Horizon H2020 project for Copernicus Service Evolution
- Duration: 2021-2023
- Budget: 1.5M Euro
- Partners: 6 from 6 European countries
- Project coordination: Jozef Skakala, PML

- 10+ investigators are also members of OceanPredict



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101004032.

Copernicus Marine Service

www.seamlessproject.org

The SEAMLESS project: research streams

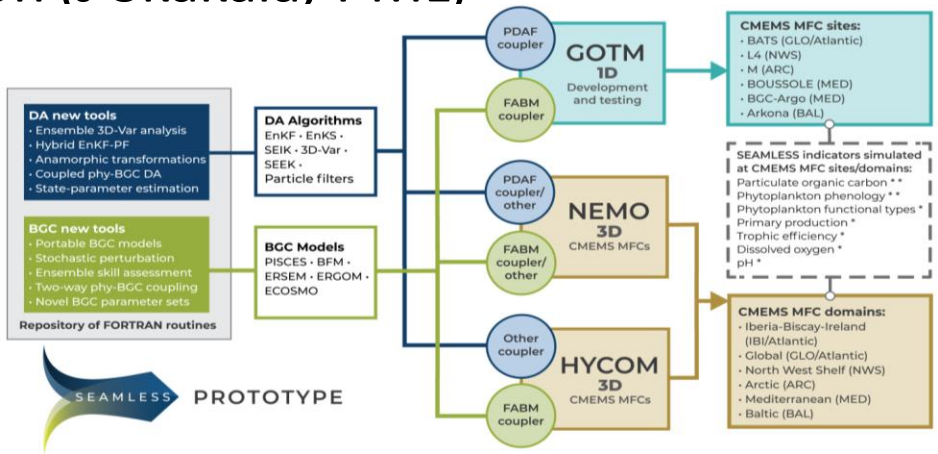
1. New ensemble generation and data assimilation methods (P Brasseur, UGA)

2. Coupled assimilation of physical and biogeochemical data (L Bertino, NERSC)

3. Coupled assimilation of remote sensing & in situ biogeochemical data (G Cossarini, OGS)

4. Coupled assimilation for joint state-parameter estimation (J Skakala, PML)

SEAMLESS Ensemble Assimilation Tool (EAT)
(J Bruggeman & K Bolding, BB; L Nerger, AWI & al.)



Ambition: to make it a free reference tool for teaching, training, research and applications in BGC modelling & DA

Being developed further by:

Copernicus Marine Service

NECCTON
NEW COPERNICUS CAPABILITY FOR TROPHIC OCEAN NETWORKS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101004032.

Ensemble and Assimilation Tool

Jorn Bruggeman, Karsten Bolding, Lars Nerger

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101004032.

Workshop outline

- **What is EAT?**
- **Components underneath**
  - GOTM: General Ocean Turbulence Model
  - FABM: Framework for Aquatic Biogeochemical Models
  - PDAF: Parallel Data Assimilation Framework
- **Features**
- **Hands-on exercises**

About water column models

- **Concept: keep vertical structure, assume horizontal homogeneity**
- **Realistic enough for many purposes**
  - vertical gradients in temperature, light and biogeochemistry,
  - (turbulent) mixing and its response to meteorological forcing
  - pelagic, surface, bottom: air-sea exchange, benthic biogeochemistry, nutrients, plankton, higher trophic levels
- **Fast**
  - simulate 1 year of physics + biogeochemistry in under 1 minute

What is EAT?

"A flexible and extensible software package for data assimilation of physical and biogeochemical variables in a one-dimensional water column"

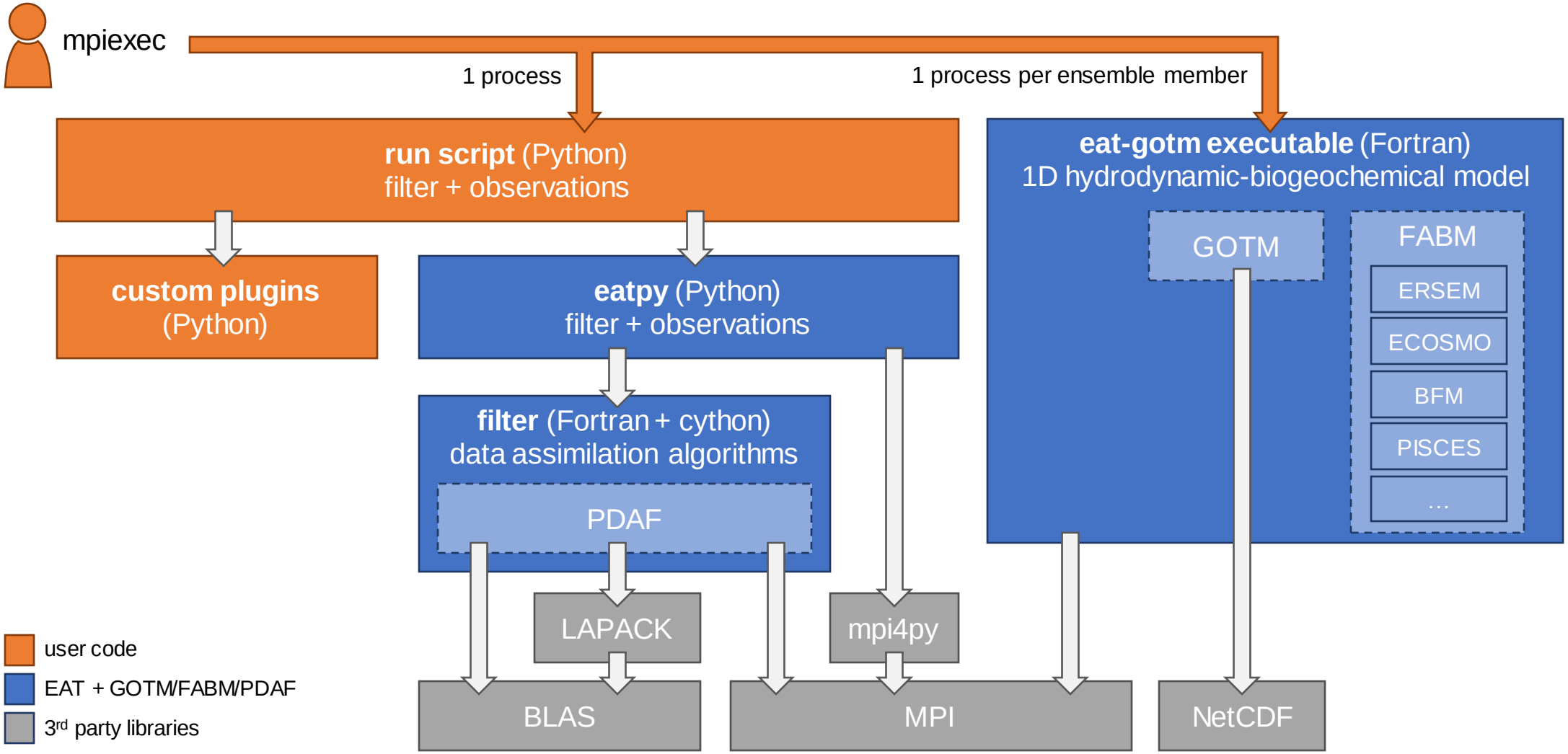
Key ingredients:

- 1D hydrodynamics (temperature, salinity, mixing) – General Ocean Turbulence Model
- A wide range of biogeochemical models – Framework for Aquatic Biogeochemical Models
- A wide range of data assimilation algorithms – Parallel Data Assimilation Framework

EAT is a compact codebase (< 5,000 lines) with a Python frontend and a Fortran backend

Emphasis on usability and flexibility when it comes to installation, file formats, execution

EAT structure



GOTM: General Ocean Turbulence Model

- **Visit https://gotm.net**
- **Approaching 25 years of continuous development**
- **Highly configurable 1D water column model**
  - Key focus was vertical mixing – large number of turbulence closure schemes
- **Linked to FABM to provide large selection of biogeochemical models (configurable at run-time)**
- **Configuration via YAML-formatted file**
- **Very few changes necessary to 'core' GOTM to integrate with EAT**
  - New *main()* calling *initialize_gotm(), integrate_gotm(), finalize_gotm()* and doing MPI exchange of state vectors with eatpy

GOTM: External Input

- **Typically read from files (but can also be constants or analytical)**
- **Initial conditions for all state variables**
  - E.g., temperature, salinity, biogeochemistry
- **Air/sea exchange – fluxes of heat and momentum**
  - u10, v10, t2, airp, humidity, precipitation (not mandatory)
  - ERA5 is a good place to look – but many other providers work as well

Live demo

https://bit.ly/igotm-mongoos

FABM: Framework for Aquatic Biogeochemical Models

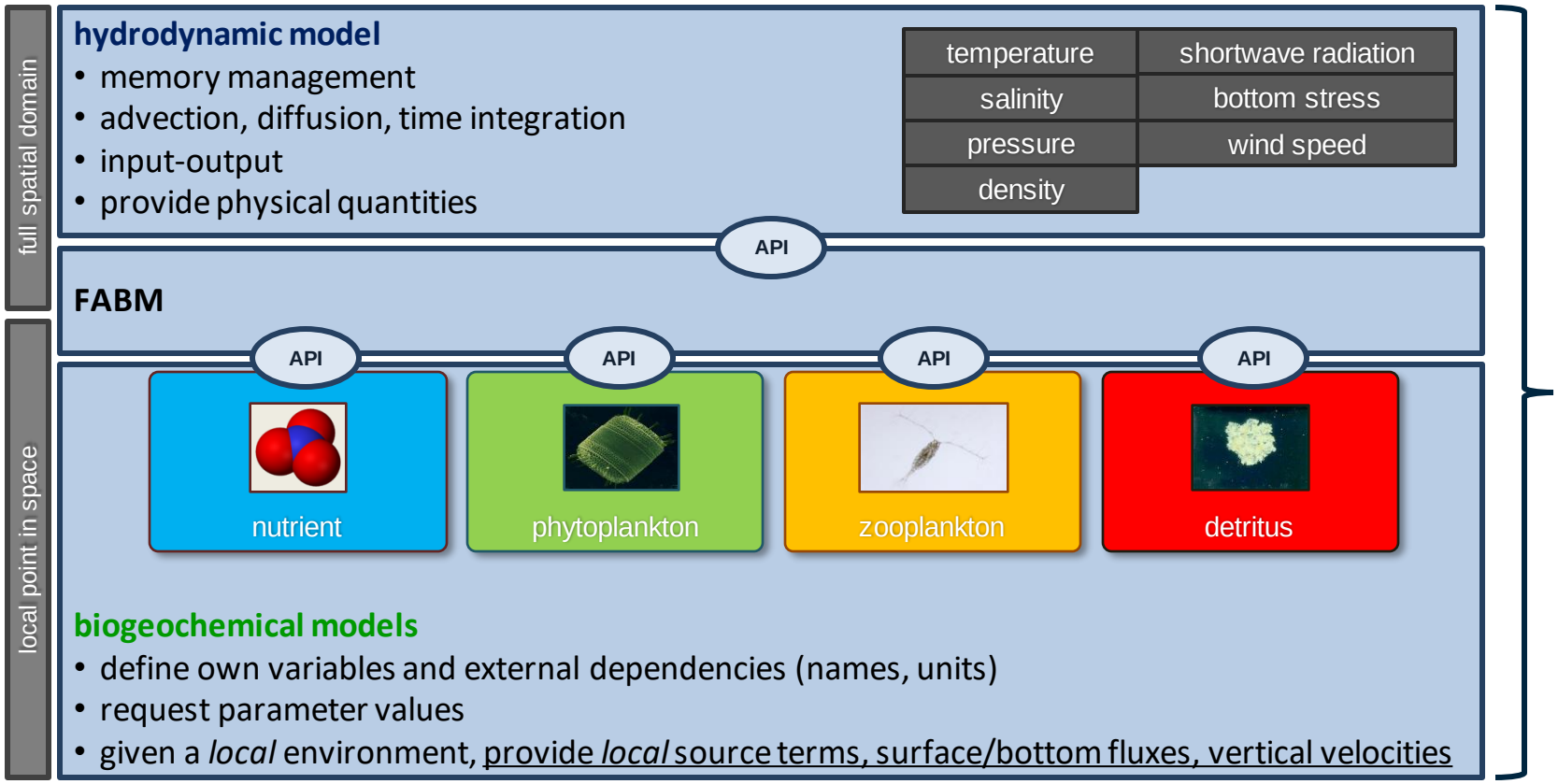
- **Developed since 2009**
  - supported by EU projects: MEECE, SEAMLESS, NECCTON, OceanICU
- **Aims:**
  - **portability**: share one biogeochemical code between different hydrodynamic models (0D, 1D, 2D, 3D)
  - **modularity**: stand-alone, process-specific biogeochemical modules, combined at run time to create the ecosystem
  - **stability:** API changes are infrequent and well documented (e.g., upcoming FABM 2.0)
- **One library, runs in host with same grid, same domain decomposition**
  - FABM does not couple across grids or time steps
  - FABM does not define its own domain decomposition or control parallelization

  Thus, it does not overlap or compete with MCT, OASIS, ESMF
- **Fortran 2003 (Intel, gfortran, Cray, PGI, AMD)**
- **Open source code + extensive documentation: https://fabm.net**
- **Published**

  Bruggeman & Bolding (2014) A general framework for aquatic biogeochemical models. *Environmental Modelling & Software* 61, 249–265. 10.1016/j.envsoft.2014.04.002

  Code releases archived on Zenodo: 10.5281/zenodo.3774497

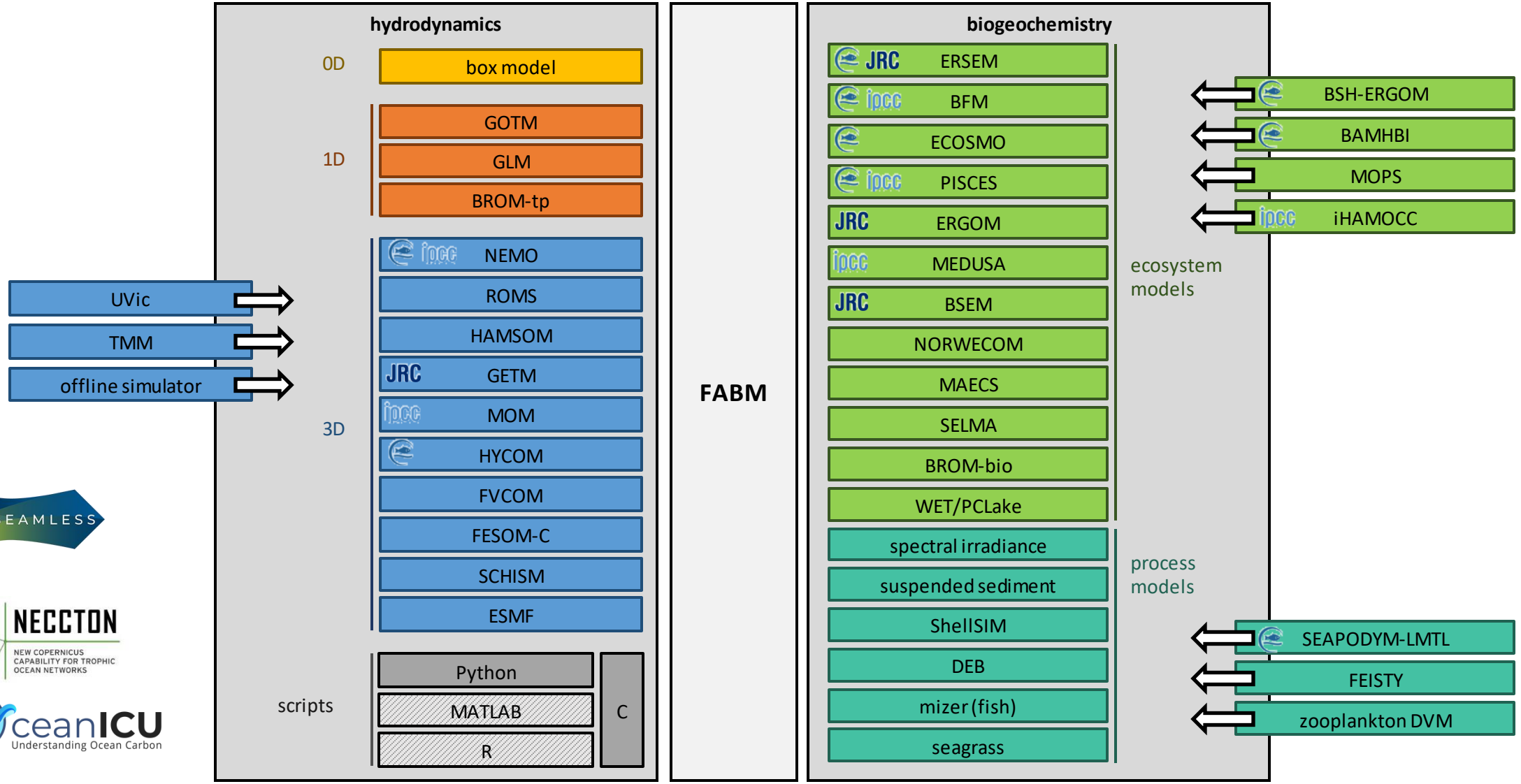
FABM: separation of concerns



$$\partial_t c + \partial_x(uc) + \partial_y(vc) + \partial_z(wc) - \partial_x(K_H \partial_x c) - \partial_y\left(K_H \partial_y c\right) - \partial_z(K_V \partial_z c) = -\partial_z(w_s c) + f$$

$$K_V \partial_z c\Big|_{z=0} = F_s,\ K_V \partial_z c\Big|_{z=z_b} = F_b$$

FABM: supported physical and biogeochemical models



NECCTON and Ocean-ICU have received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101081273 and 101083922

PDAF: Parallel Data Assimilation Framework

PDAF Parallel Data Assimilation Framework

- **A unified tool for interdisciplinary data assimilation …**
  - provide support for parallel ensemble forecasts
  - provide assimilation methods (solvers) - fully-implemented & parallelized
  - provide tools for observation handling and for diagnostics
  - easily useable with (probably) any numerical model
  - a program library (PDAF-core) plus additional functions
  - run from notebooks to supercomputers (Fortran, MPI & OpenMP)
  - ensure separation of concerns (model – DA method – observations – covariances)
  - First release in 2004 – continuous development

Open source:
Code, documentation, and tutorial available at

http://pdaf.awi.de

github.com/PDAF/PDAF

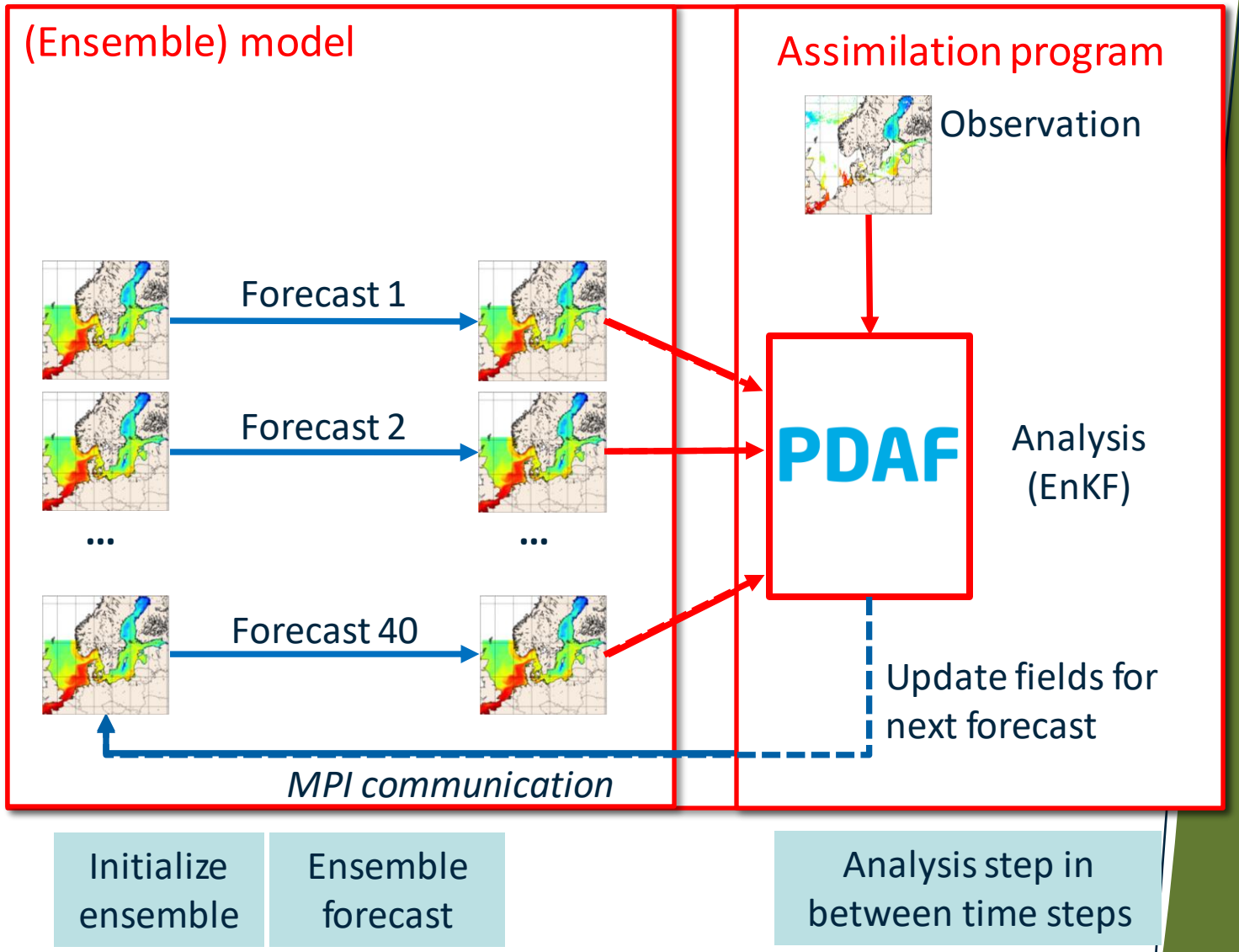
L. Nerger, W. Hiller, Computers & Geosciences 55 (2013) 110-118

Assimilation-enabled model using PDAF

PDAF Parallel Data Assimilation Framework

EAT- Coupling of model with PDAF

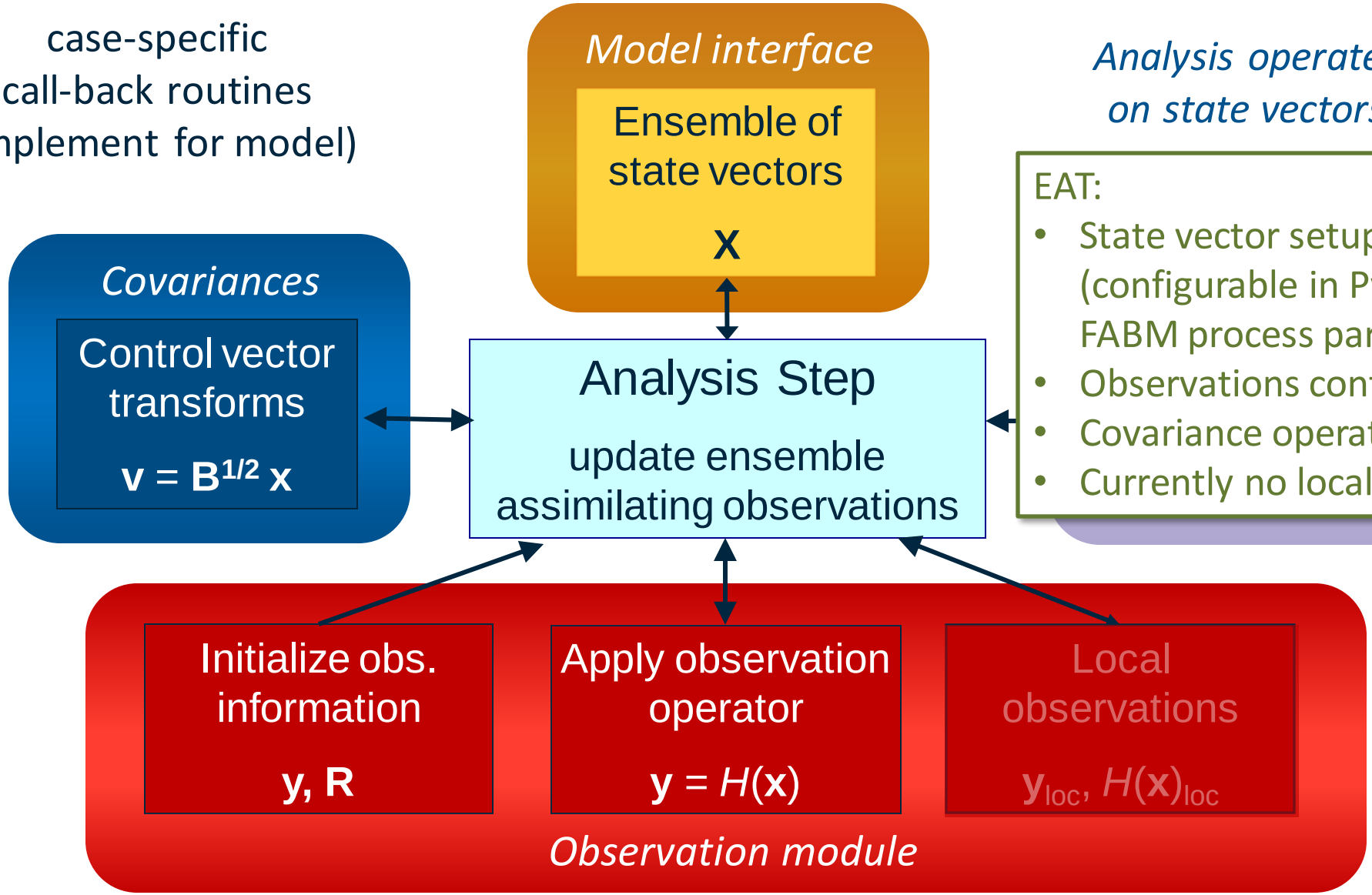
- Modify model to simulate ensemble of model states
- Insert analysis step/solver to be executed at prescribed interval *(EAT: insert MPI communication)*
- Run model as usual, but with more processors and additional options



Implementation of analysis step

PDAF Parallel Data Assimilation Framework

case-specific
call-back routines
(implement for model)



Analysis operates on state vectors

EAT:

- State vector setup fully coded (configurable in Python including FABM process parameters)
- Observations configured in Python
- Covariance operations in Python
- Currently no localization (1D)

PDAF Package: DA Methods, Models, etc.

PDAF Parallel Data Assimilation Framework

PDAF originated from comparison studies of different filters

Ensemble Filters and smoothers - *global and localized*

- EnKF (Evensen, 1994 + perturbed obs.)
- (L)ETKF (Bishop et al., 2001/Hunt et al. 2007)
- ESTKF (Nerger et al., 2012)
- NETF (Toedter & Ahrens, 2015)
- Particle filter
- Hybrid LKNETF (Nerger, 2022)
- *EnOI mode*

Available in EAT

3D-Var schemes
(incremental with control variable transformation)

- 3D-Var with parameterized covar.
- 3D Ensemble Var
- Hybrid 3D-Var

Model bindings

- MITgcm
- NEMO (separate repo)
- AWI-CM / FESOM

Toy models (full implementations with PDAF)

- Lorenz-96 / Lorenz-63
- Lorenz-2005 models II and III

Community:

- pyPDAF (Python-coded models)
- TerrSysMP-PDAF

In progress

- SCHISM/ESMF (VIMS)

EAT: installation

- **Pre-built conda package for Linux, Mac, Windows**

  ```
  conda create -n eat -c bolding-bruggeman -c conda-forge eatpy
  ```

- **Build yourself with conda compilers/MPI/NetCDF/BLAS/LAPACK**

  to include custom FABM-based biogeochemistry

- **Build yourself with system compilers/MPI/NetCDF/BLAS/LAPACK**

  on HPC systems

more info on wiki

Ensemble generation

You can perturb:

- Physical parameters and forcing (`gotm.yaml`)
- Biogeochemical parameters (`fabm.yaml`)
- Initial conditions (`restart.nc`)

```python
import eatpy
import numpy as np

N = 20    # ensemble size
with eatpy.models.gotm.YAMLEnsemble("gotm.yaml", N) as f:
    f["surface/u10/scale_factor"] = np.random.lognormal(sigma=0.1, size=N)
    f["surface/v10/scale_factor"] = np.random.lognormal(sigma=0.1, size=N)
    f["turbulence/turb_param/k_min"] = 3e-6 * np.random.lognormal(sigma=0.1, size=N)
```

Observations: simple text files

depth-dependent (e.g., Argo)

```
# time  depth  temperature  s.d.
2020-05-08 10:12:00 -10.0 15.2 0.1
2020-05-08 10:12:00 -20.0 14.6 0.1
                    …
```

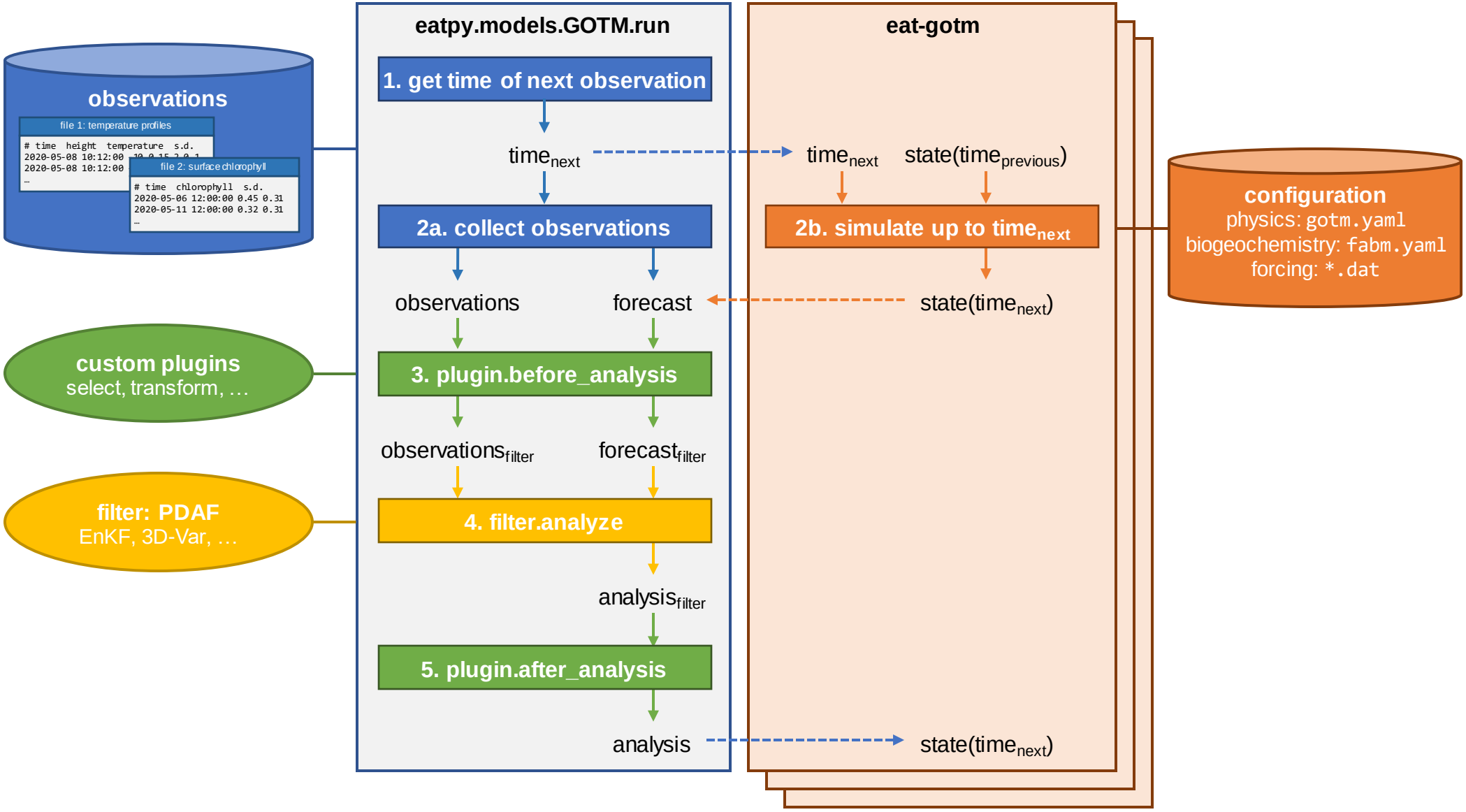
depth-independent (e.g., satellite)

```
# time  chlorophyll  s.d.
2020-05-06 12:00:00 0.45 0.31
2020-05-11 12:00:00 0.32 0.29
                    …
```

The (augmented) model state

- **Starting point: complete model state**
  - All physical state variables
  - All biogeochemical state variables (pelagic and benthic)
- **Optionally, select a subset to present to the DA filter**
  - For instance, “update temperature and salinity only”
- **Augment the model state to:**
  - Assimilate observations on physical and biogeochemical diagnostics (for instance, primary production)
  - Estimate biogeochemical parameters (scalar)

EAT data assimilation cycle



Plugins

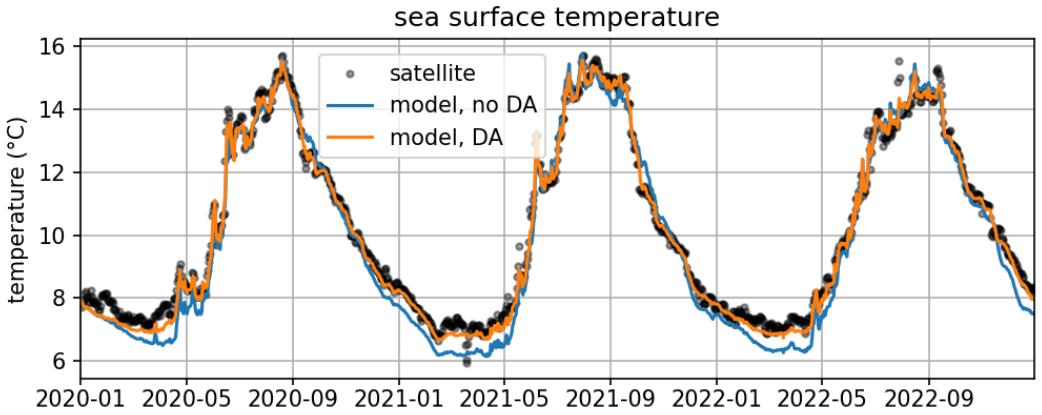
- **Typical uses**
  - Limit the data assimilation update to a subset of the model state
  - Transform variables into “Gaussian” space
  - Check state validity
  - Apply additional constraints to state variable values
    For instance, to ensure values remain physically meaningful, or to ensure mass conservation
  - Specify the background error covariance matrix in variational schemes
  - Save ensemble state or custom diagnostics
- **Suitable for advanced uses**
  - Reconstruct density increments from forecast/analysis T&S, and from these, calculate nutrient increments (Anna Teruzzi, CMEMS MED MFC)

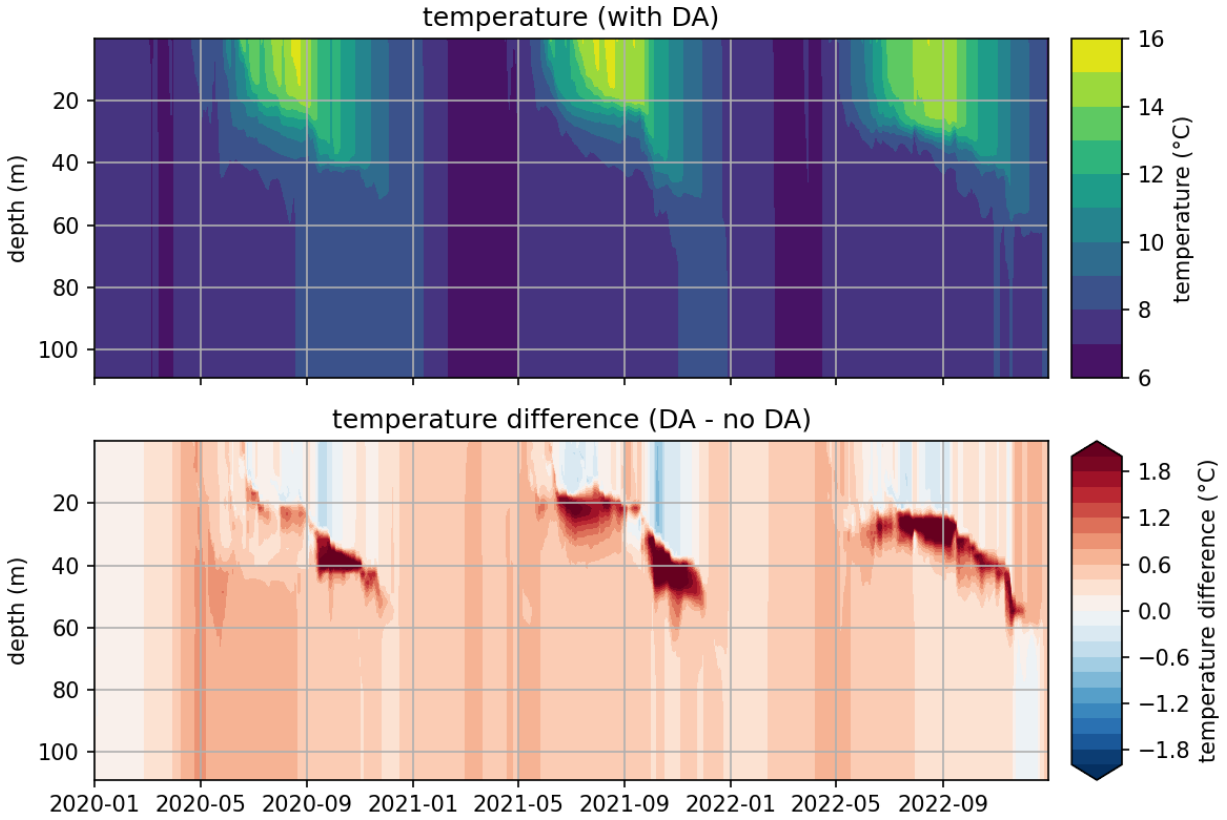
SST assimilation: example run script

```
import eatpy

experiment = eatpy.models.GOTM()
filter = eatpy.PDAF(eatpy.pdaf.FilterType.ESTKF)
experiment.add_plugin(eatpy.plugins.select.Select(include=("temp", "salt")))
experiment.add_observations("temp[-1]", "cci_sst.dat")
experiment.run(filter)
```

SST assimilation: example results

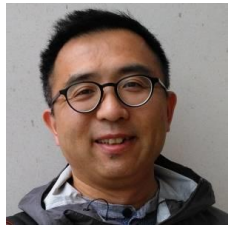




EAT resources

- **Code:** https://github.com/BoldingBruggeman/eat
- **User guide:** https://github.com/BoldingBruggeman/eat/wiki
- **Report** D2.4-v3.pdf **with example applications:**

Biogeochemical parameter estimation (ERSEM)
Jozef Skákala, PML



ARC MFC-like: integrate EAT in custom workflows, e.g., 8d assimilation cycle (ECOSMO)
Tsuyoshi Wakamatsu, NERSC

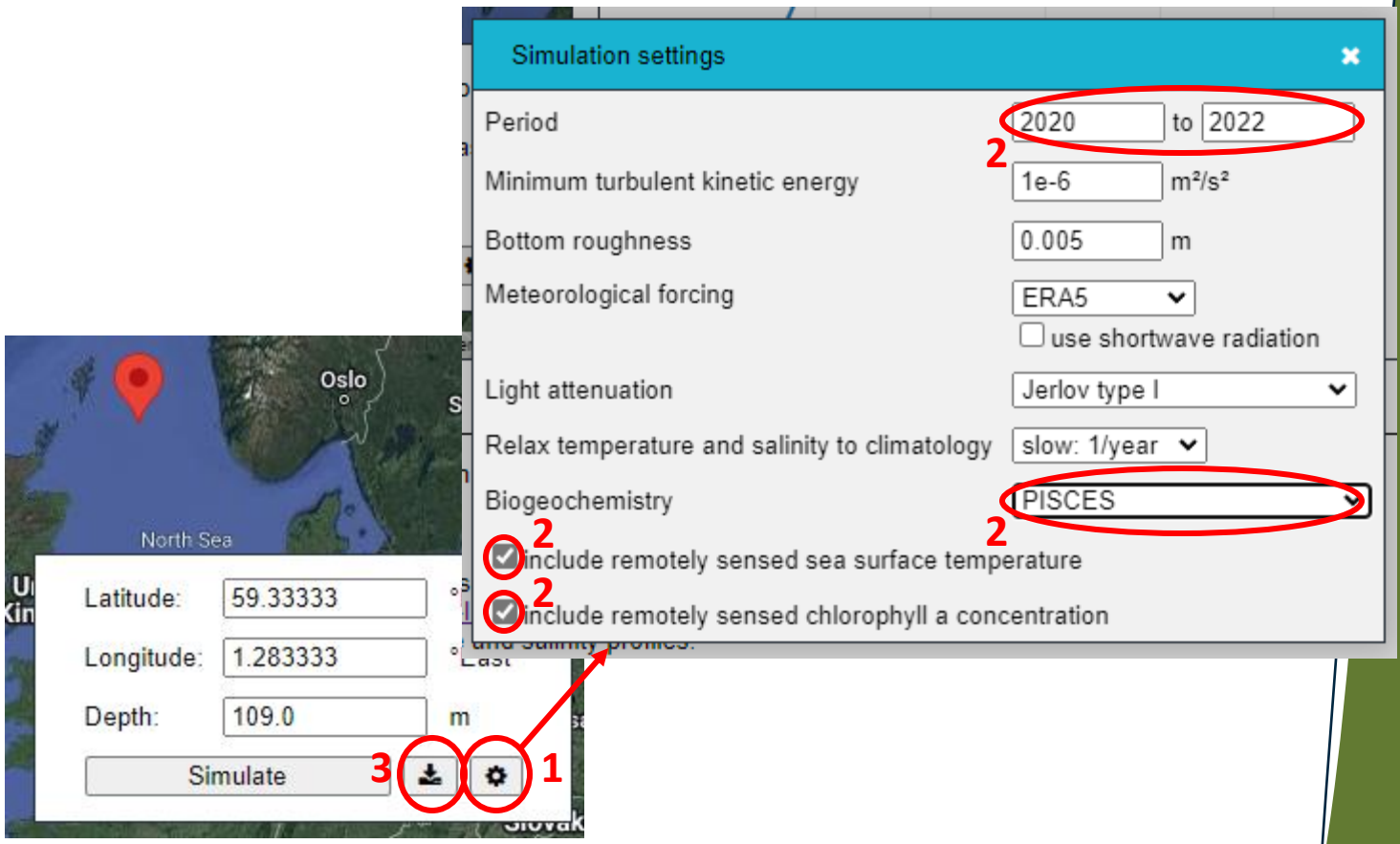
MED MFC-like: 3D-Var with background covariance based on EOFs (BFM)
Anna Teruzzi, OGS

Hands-on

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101004032.

Download a GOTM-FABM setup

- Visit https://bit.ly/igotm-mongoos
- Click a location to simulate, then click settings (1)
- Customize (2):
  - the time period (recommended: 3 years)
  - biogeochemistry (recommended: PISCES, ERSEM, BFM or ECOSMO)
  - include remotely sensed surface temperature and chlorophyll
- Click download (3)
- Extract the zip file



What's in an iGOTM setup?

`gotm.yaml`	physics configuration
`fabm.yaml`	biogeochemistry configuration
`grid.dat`	vertical grid (relative layer thicknesses)
`sprof.dat` `tprof.dat`	temperature and salinity profiles for initialization and relaxation (WOA2018)

`meteo.dat`
`precip.dat`
`ssr.dat`
	atmospheric forcing (ERA5)

`ext_press.dat` `zeta.dat`	tidal forcing (TPXOv9)

`nitrate.dat`
`phosphate.dat`
`silicate.dat`
`oxygen.dat`
`TAlk.dat`
`TCO2.dat`
	biogeochemical profiles for initialization (WOA2018 and GLODAPv2)

`cci_sst.dat` `cci_chl.dat`	remotely sensed surface temperature and chlorophyll (SST CCI and OC CCI)

YAML configuration: gotm.yaml

```yaml
# This file was created with only commonly used settings, plus those that differ from the default specified by GOTM.
# You can generate a configuration with every possible setting with: gotm --write_yaml <OUTFILE> --detail full
# To see only the settings that differ from the default, use: gotm --write_yaml <OUTFILE> --detail minimal
version: 6                                          # version of configuration file [default=6]
title: iGOTM simulation                             # simulation title used in output [default=GOTM simulation]
location:
   name: iGOTM station                              # station name used in output [default=GOTM site]
   latitude: 5.79381800E+01                         # latitude [degrees North; min=-90.0; max=90.0; default=0.0]
   longitude: 2.52136000E+00                        # longitude [degrees East; min=-360.0; max=360.0; default=0.0]
   depth: 70.0                                      # water depth [m; min=0.0; default=100.0]
time:
   start: 2020-01-01 00:00:00                       # start date and time [yyyy-mm-dd HH:MM:SS; default=2017-01-01 00:00:00]
   stop: 2022-12-31 18:00:00                        # stop date and time [yyyy-mm-dd HH:MM:SS; default=2018-01-01 00:00:00]
   dt: 1800.0                                       # time step for integration [s; min=0.0; default=3600.0]
grid:
   nlev: 82                                         # number of layers [min=1; default=100]
   method: file_sigma                               # layer thickness specification [analytical=equal by default with optional zooming
   ddu: 1.0                                         # surface zooming [dimensionless; min=0.0; default=0.0]
   ddl: 1.0                                         # bottom zooming [dimensionless; min=0.0; default=0.0]
   file: grid.dat                                   # path to file with layer thicknesses [default=]
temperature:                                        # temperature profile used for initialization and optionally relaxation [Celsius]
   method: file                                     # method [off, file=from file, constant, two_layer=two layers with linear gradient
   constant_value: 0.0                              # value to use throughout the simulation [Celsius; default=0.0]
   file: tprof.dat                                  # path to file with series of profiles [default=]
   column: 1                                        # index of column to read from [default=1]
   two_layer:
      z_s: 0.0                                      # depth where upper layer ends [m; min=0.0; default=0.0]
      t_s: 0.0                                      # upper layer temperature [Celsius; min=0.0; max=40.0; default=0.0]
      z_b: 0.0                                      # depth where lower layer begins [m; min=0.0; default=0.0]
      t_b: 0.0                                      # lower layer temperature [Celsius; min=0.0; max=40.0; default=0.0]
   NN: 0.0                                          # square of buoyancy frequency [s^-2; min=0.0; default=0.0]
   relax:                                           # relax model temperature to observed/prescribed value
      tau: 3.15000000E+07                           # time scale for interior layer [s; min=0.0; default=1.00000000E+15]
```

Using EAT (next slides show example scripts)

- **Load the EAT environment whenever you open a new terminal**
  ```
  conda activate eat
  ```
- **Commands will be executed from the directory where you extracted the iGOTM zip file**
  ```
  cd <SETUPDIR>
  ```
- **Run a stand-alone simulation (no ensemble, no data assimilation)**
  ```
  eat-gotm
  ```
- **Generate an ensemble (example generation scripts on next slides)**
  ```
  python <GENERATE_SCRIPT>
  ```
- **Run a data assimilation experiment (example run scripts on next slides)**
  ```
  mpiexec -n 1 python <RUN_SCRIPT> \
        : -n <NENS> eat-gotm [--separate_gotm_yaml] [--separate_restart_file]
  ```
- **Analyze results:**
  ```
  jupyter lab
  ```
  <allow the browser to open, then click one of the Jupyter notebooks: *.ipynb>

The following slides show example scripts. These are also available here.

Generate the ensemble

```
import numpy as np
import eatpy
N = 20    # ensemble size

gotm = eatpy.models.gotm.YAMLEnsemble("gotm.yaml", N)
fabm = eatpy.models.gotm.YAMLEnsemble("fabm.yaml", N)
with gotm, fabm:
    gotm["surface/u10/scale_factor"] = np.random.lognormal(sigma=0.2, size=N)
    gotm["surface/v10/scale_factor"] = np.random.lognormal(sigma=0.2, size=N)
    gotm["turbulence/turb_param/k_min"] *= np.random.lognormal(sigma=0.2, size=N)
    gotm["fabm/yaml_file"] = fabm.file_paths
    fabm["instances/phy/parameters/mumax0"] *= np.random.lognormal(sigma=0.2, size=N)
    fabm["instances/dia/parameters/mumax0"] *= np.random.lognormal(sigma=0.2, size=N)
```

In the repository: generate_ensemble_phys_bgc.py

Run script: assimilate SST and chlorophyll

```python
import eatpy

# Notes:
# * If you are running ERGOM, replace total_chlorophyll_calculator_result with msi_ergom1_tot_chla
# * A simpler example where only SST is assimilated is given in assimilate_sst.py

experiment = eatpy.models.GOTM(
    diagnostics_in_state=["total_chlorophyll_calculator_result"]
)

filter = eatpy.PDAF(eatpy.pdaf.FilterType.ESTKF)

bgc_variables = [v for v in experiment.variables if "_" in v]
experiment.add_plugin(
    eatpy.plugins.select.Select(include=["temp", "salt"] + bgc_variables)
)
experiment.add_plugin(eatpy.plugins.check.Finite())
experiment.add_plugin(
    eatpy.plugins.transform.Log(
        "total_chlorophyll_calculator_result",
        *bgc_variables,
        transform_obs=False,
        minimum=1e-12
    )
)

# If you comment out the two lines below, you run the ensemble only without assimilation
experiment.add_observations("temp[-1]", "cci_sst.dat")
experiment.add_observations("total_chlorophyll_calculator_result[-1]", "cci_chl.dat")

experiment.run(filter)
```

In the repository: assimilate_sst_chl.py

Run script: estimate BGC parameters

```python
import eatpy

# Notes:
# * You can estimate any biogeochemical parameter included in fabm.yaml.
#   Check this file to see possible options. The example below is for the PISCES model.
# * If you are running ERGOM, replace total_chlorophyll_calculator_result with msi_ergom1_tot_chla

experiment = eatpy.models.GOTM(
    diagnostics_in_state=["total_chlorophyll_calculator_result"],
    fabm_parameters_in_state=["instances/phy/parameters/mumax0", "instances/dia/parameters/mumax0"]
)

filter = eatpy.PDAF(eatpy.p

par_variables = [v for v in
experiment.add_plugin(
    eatpy.plugins.select.Se
)
experiment.add_plugin(eatpy
experiment.add_plugin(
    eatpy.plugins.transform.Log(
        "total_chlorophyll_calculator_result",
        *par_variables,
        transform_obs=False,
        minimum=1e-12
    )
)

# If you comment out the two lines below, you run the ensemble only without assimilation
experiment.add_observations("temp[-1]", "cci_sst.dat")
experiment.add_observations("total_chlorophyll_calculator_result[-1]", "cci_chl.dat")

experiment.run(filter)
```

But first: make sure parameters are included in output!
Customize output section in `gotm.yaml`
Then, regenerate the ensemble (perturbed `gotm.yaml` files)

In the repository: assimilate_sst_chl_pars.py

More information on the EAT wiki

- **Generating ensembles, perturbation approaches**
- **Specifying observations, file formats**
- **Filter settings**
- **Using and writing plugins**

If something goes wrong…

- **The ensemble collapses (part of the model state becomes identical in all ensemble members)**
  - Perturb additional forcing variables and/or parameters, or perturb them more strongly, to ensure spread is sustained throughout the simulation
  - Configure the data assimilation filter to inflate the ensemble (forget parameter)
- **The model state becomes corrupted (e.g., NaN)**
  - Reduce the model time step (`time/dt` in `gotm.yaml`)
  - Ensure the model state stays physically/biogeochemically meaningful by (log)transforming selected variables, or clipping the analysis state to lower/upper bounds

Acknowledgements

PML | Plymouth Marine Laboratory

UGA Université Grenoble Alpes

AWI

Bolding & Bruggeman

OGS

NERSC

www.seamlessproject.org

seamless@pml.ac.uk

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101004032