Contents

1 ESBMTK - An Earth-sciences box modeling toolkit

ESBMTK is python library which aims to simplify typical box modeling projects the in Earth-Sciences. The general focus is to make box modeling more approachable for classroom teaching. So performance and scalability currently no priority. Specifically, the solver is just a simple forward Euler scheme, so stiff problems are not handled gracefully.

At present, it will calculate masses/concentrations in reservoirs and fluxes including isotope ratios. It provides a variety of classes which allow the creation and manipulation of input signals, and the generation of graphical results.

2 News

- August 10^{th}, Added KF and KS values for hydrofen fluoride and bisulfate to the seawater chemistry module.
- August 2^{nd}, 0.7.0.0 Python namespaces are now the default. Esbmtk now supports carbonate chemistry. Tracers like bicarbonate and carbonate-ion concentration can be calculated for reservoir groups which track total alkalinity and dissolved inorganic carbon. The function `add_carbonate_system_1` will add these tracers to a given reservoir group. The function `add_carbonate_system_2` will additionally compute carbonate burial and dissolution fluxes, following the approach of [**?**]. Big thanks to Tina and Mahrukh who developed and

tested the carbonate chemistry module. Note that the current release has not yet updated the documentation or the examples in the github repository.

- July 28th, esbmtk now supports python name spaces. The default is still to register all esbmtk objects in the global namespace. However, in cases where models need to be integrated into python code, you can now set the `register = 'local'` keyword in the model declaration. In this case, all model object follow are hierarchical naming scheme e.g., `M.A_sb.DIC` denotes the DIC concentration in the `A_sb` reservoir group which belongs to model `M`.

- July 20th, the model object now provides a `sub_sample_data()` which will resample all model data to a default grid of 1000 data points, before plotting.

- July 17th, `ExternalCode` is a new class to allow integration of arbitrary code. This replaces the `VirtualReservoir-no_set` class

- June 17th, data and state files are now stored in sub directories. Model runs can now be broken down into individual segments which allows for long running models without exhausting memory. See the `step_limit` parameter in `Model`. ESBMTK now automatically reduces the number of datapoints to 1000 before saving (or plotting) data. See the `number_of_datapoints` parameter in `Model`.

- May 26th, 0.6.0.0 changed data-structure for the `Generic_Function` class. This will break any previous use of `VirtualReservoir_no_set` instances. See the API documentation on how to update. Changed the data-structure of all process classes. There should be no user facing changes.

- May 13th, 0.5.1.3 Multiple regression fixes, the `ref` keyword is now called `ref_reservoirs`. Added two new classes `Reservoir_no_set`, and `VirtualReservoir_no_set`. Both classes are agnostic about changes to their data. `=Reservoir_no_set` will only change in response to fluxes, but will not update concentration data etc. Likewise for `VirtualReservoir_no_set` whose values will only change in response to the associated function

- May 5th, 0.5.0.1 The Datafield Class now accepts lists of datasets. This facilitates the grouping of data which belong together into a single graph.

- April 26th Further changes to the naming scheme in group connections. A connection group object now consists of the group name followed by the connection name, e.g., `sb2@db.PO4_2_PO4` which denotes a connection from `sb` to `db` transferring the species `PO4`. This conflicts with the previous scheme where the above would reference a flux. The corresponding flux can referenced by adding `_F` to the above connection name. I.e., `sb2@db.PO4_2_PO4_F`. Since this breaks previous code, the version is updated to 0.5.0.0

- April 25th v 0.4.3.0 ESBMTK has now 3 different solvers. The hybrid solver mentioned below, and a full numba solver which is about 10 faster. The latter does not yet support all connection properties though. The solver is chosen via the optional solver keyword in the run method: `M.run(solver = "hybrid")`, or `M.run(solver = "numba")`. Both incur a startup overhead of about 3 to 5 seconds. In order to make the numba solver work, the interface definition for the `GenericFunction` and `VirtualReservoir` classes changed from 6 to 3 arguments, an all 3 arguments must be present and follow a strict structure (see the class definitions). This also required changes in the carbonate chemistry module, specifically the functions which calculate pH and carbonate alkalinity. The documentation is now available at `https://uliw.github.io/esbmtk/`

- April 13th: rewrote the solver which is now 3 times faster. Added numba to the solver code, however the performance gain is currently only a few percent.. Added plot method to the model class. This method will plot any object inn a given list. This is useful for larger models where one is only interested in a subset of results.

- April 10th: The hopefully last tweak to the naming scheme. All fluxes belong to a connection (see `model.connection_summmary()`), and registered in the respective connection namespace (i.e., `sb2ib.flux_name`). All processes are now registered in the respective flux name space, i.e., `sb2ib.flux_name.process_name`. All of these can be queried with the info method, e.g., `sb2ib.flux_name.process_name.info()`

- April 6th, added several function which aid in the bulk creation of reservoirs and connections (i.e., `create_reservoirs`, `create_bulk_connections`). The hypsometry class is now part of the Model object and now has method to calculate the volume contained in a given depth interval. To calculate the ocean volume you can call e.g.,

`Model.hyp(0,-6000)` see the api docs for the sealevel module for details. Reservoirs can now be specified by their geometry rather than by volume or mass. See the documentation of the reservoir class.

The DataField class will now print a warning when used before model results are computed

- April 1st. Added `carbonate_system()` function to the carbonate chemistry module. This function simplifies the setup of the H^+ and carbonate alkalinity reservoirs. See the api docs for details.

  - March 28th added a `flux_summmary()` and `connection_summary()` methods to the model class.

- March 27th, 0.4.0.5 added the hypsometry class which provides a spline representation of the hypsometry between -6000 mbsl and 1000 asl.This class provides the `area()` method which calculates the seafloor surface area between two depth dates. See the online api documentation for details.

- March 26th, 0.4.0.4 the `write_state` and `read_state` methods are now compatible with ReservoirGroups

- March 18th esbmtk 0.4.0.0 now has a carbonate chemistry module which currently includes methods to calculate PCO_2, CA, and H^+ concentrations from TA and DIC. The seawater class has been renamed `SeawaterConstants` and provides access to a limited set of seawater species concentrations and their K and Pk constants at given set of temperature, salinity and pressure conditions. This version also includes some refactoring in the `Connnection` and `ConnectionGroup` classes. It is likely that this broke some connection types.

- March 13th, cleaned up the use of the `k_value` keyword which is now restricted to the `flux_balance` connection type. In all other instances use the `scale` keyword instead. The old keyword is still working, but will print a warning message. The `describe()` method is now called `info()`.

- March 11th, added a seawater class which provides access to K-values, and concentrations.

- March 10th, the code documentation is now available at `https://uliw.github.io/esbmtk/`

- March 6th, the plot reservoir function now takes and additional filename argument e.g., (fn="foo.pdf"). Signals now accept an optional reservoir argument. This simplifies signal creation as the source and reservoir connection can be created implicitly.

- Feb. 28th, added a VirtualReservoir class. This class allows the definition of reservoirs which depend on the execution of a user-defined function. See the class documentation for details.

  Display precision can now be set independently for each Reservoir, Flux, Signal, Datafield and VirtualReservoir

- Jan. 30th, added oxygen and nitrogen species definitions

- Jan. 18th, Reading a previous model state is now more robust. It no longer requires the models model have the same numbers of fluxes. It will attempt to match by name, and print a warning for those fluxes it could not match.

- Jan. 12th, The model object now accepts a `plot_style` keyword

- Jan. 5th, Connector objects and fluxes use now a more consistent naming scheme: `Source_2_Sink_Connector`, and the associated flux is named `Source_2_Sink_Flux`. Processes acting on flux are named `Source_2_Sink_Pname`

  The model type (`m_type`) now defaults to `mass_only`, and will ignore isotope calculations. Use `m_type = "both"` to get the old behavior.

- Dec. 30th, the connection object has now a generalized update method which allows to update all or a subset of all parameters

- Dec. 23rd, the connection object has now the basic machinery to allow updates to the connection properties after the connection has been established. If need be, updates will trigger a change to the connection type and re-initialize the associated processes. At present this works for changes to the rate, the fractionation factor, possibly delta.

- Dec. 20th, added a new connection type (`flux_balance`) which allows equilibration fluxes between two reservoirs without the need to specify forward and backwards fluxes explicitly. See the equilibration example in the example directory.

- Dec. 9th, added a basic logging infrastructure. Added `describe()` method to `Model`, `Reservoir` and `Connnection` classes. This will list details about the fluxes and processes etc. Lot's of code cleanup and refactoring.
- Dec. 7th, When calling an instance without arguments, it now returns the values it was initialized with. In other words, it will print the code which was used to initialize the instance.
- Dec. 5th, added a DataField Class. This allows for the integration of data which is computed after the model finishes into the model summary plots.
- Nov. 26th Species definitions now accept an optional display string. This allows pretty printed output for chemical formulas.
- Nov. 24th New functions to list all connections of a reservoir, and to list all processes associated with a connection. This allows the use of the help system on process names. New interface to specify connections with more complex characteristics (e.g., scale a flux in response to reservoir concentration). This will breaks existing scripts which use these kind of connections. See the Quickstart guide how to change the connection definition.
- Nov. 23rd A model can now save it's state, which can then be used to initialize a subsequent model run. This is particularly useful for models which require a spin up phase to reach equilibrium
- Nov. 18th, started to add unit tests for selected modules. Added unit conversions to external data sets. External data can now be directly associated with a reservoir.
- Nov. 5th, released version 0.2. This version is now unit aware. So rather than having a separate keyword for `unit`, quantities are now specified together wit their unit, e.g., `rate = "15 mol/s"`. This breaks the API, and requires that existing scripts are modified. I thus also removed much of the existing documentation until I have time to update it.
- Oct. 27th, added documentation on how to integrate user written process classes, added a class which allows for concentration dependent flux. Updated the documentation, added examples
- Oct. 25th, Initial release on github.

3 Contributing

Don't be shy. Contributing is as easy as finding bugs by using the code, or maybe you want to add a new process code? If you have plenty of time to spare, ESMBTK could use a solver for stiff problems, or a graphical interface ;-) See the todo section for ideas.

4 Installation

ESBMTK relies on the following python versions and libraries

- python > 3.6
- matplotlib
- numpy
- pandas
- typing
- nptyping
- pint

If you work with conda, it is recommended to install the above via conda. If you work with pip, the installer should install these libraries automatically. ESBMTK itself can be installed with pip

- pip install esbmtk

5 Documentation

The documentation is available in org format or in pdf format. See the documentation folder, specifically the quickstart guide.

The API documentation is available at `https://uliw.github.io/esbmtk/esbmtk/index.html`

At present, I also provide the following example cases (as py-files and in jupyter notebook format)

- A trivial carbon cycle model which shows how to set up the model, and read an external csv file to force the model.
-

6 Todo

- expand the documentation
- provide more examples
- do more testing

7 License