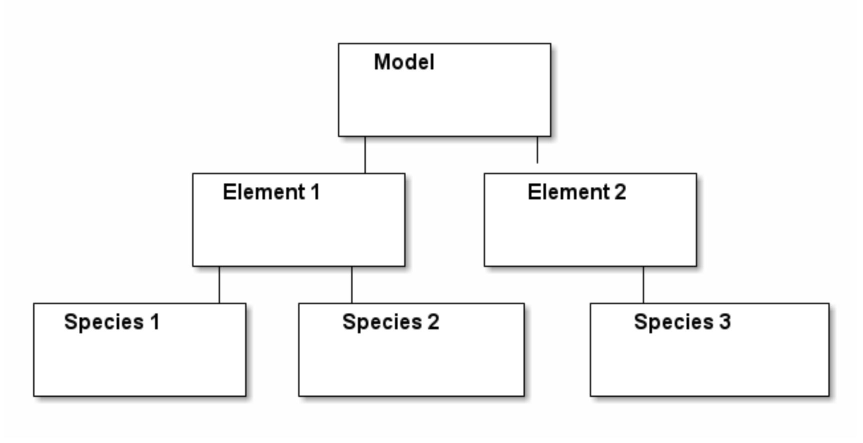
ESBMTK Quick Start Guide

Ulrich G Wortmann

October 30, 2020

1 Overview

ESBMTK provides a couple of classes which are used to define a model. The classes are arranged in a hierarchical manner, starting with the model class. The model class (or object), set's global parameters like the model name, the time step etc. Next comes one or more element objects which define basic element properties. Each element can have one or more species. Note that these classes specify units, however at present, there is incomplete support for unit conversions. So it is up to the author to ensure proper unit conversions. It is therefore best, to use the same units throughout.

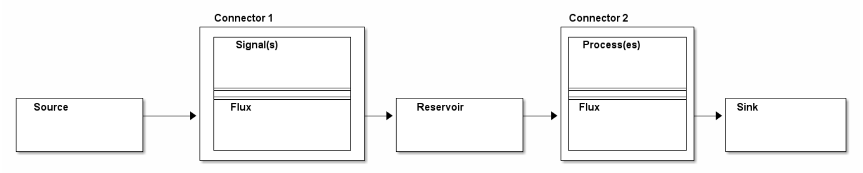


At present, these classes only specify the isotopic reference ratios, and plot labels.

Next comes the source and sink class, which simply specify the name and species, followed by reservoir object which specifies things like reservoir name, reservoir size, reservoir species etc. Each reservoir can only have one element. This is counter intuitive since we would think of the ocean as single reservoir with many elements. However, the model only attempts to track the transfer of mass between reservoirs, for a given element. For multi element models, you need to setup reservoirs and connections for each element.

Sources, sinks, and reservoirs are connected through fluxes, and fluxes are either forced by signals, or affected by processes. Most processes can be derived implicitly from the way we connect reservoirs (see below), however signal must be specified explicitly. It is thus best to define these first (see the worked example below).

The connection between a source and a sink (or two reservoirs) is handled by the connection class. This class will create the necessary fluxes, and where possible add processes. Fluxes and processes are also implemented as objects.

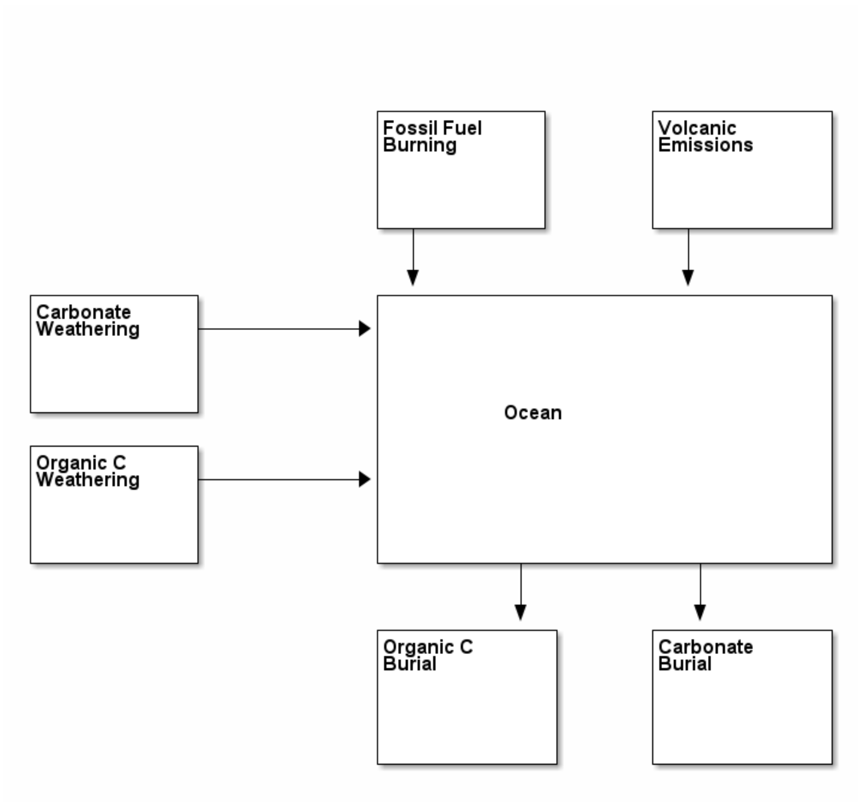


Each connector object can have more than one signal or process (or a mixture of signals and processes). Each reservoir can have more than one connector object.

1.1 A worked example

In the following example we will set up a simple carbon cycle model. The data forcing the anthropogenic carbon flux will be read from a csv file. Interaction with external data is handled through the external data object which allows to integrate external data into the model framework. It can then be used to generate a signal, or it can be associated with a reservoir so that the data is plotted with the reservoir data.

The model consists of four sources, two sinks, and one reservoir



1.2 Setting up the model

We need to load all required libraries and all classes we want to use. Interaction with the model classes is done through keyword/value pairs. Use `help()` to inquire about the supported keyword value pairs.

```python
from esbmtk import Model, Element, Species, Reservoir
from esbmtk import Signal, Connect, Source, Sink, Flux
from esbmtk import ExternalData

# create model
Model(
    name="C_Cycle",  # model name
    stop=100,         # end time of model
    time_unit="yr",  # time units
    dt=1,            # time step
)
```

1.3 Declare elements and species

We register the elements(s) with the model by providing the model name. Note that this is not a string, but the model handle which is derived from the model name in the model definition above. We use the element handle in a similar way to register the species with an element.

```
# Element properties
Element(
    name="C",                   # Element Name
    model=C_Cycle,              # Model handle
    mass_unit="mmol",           # base mass unit
    li_label="C^{12$S",         # Name of light isotope
    hi_label="C^{13}$S",        # Name of heavy isotope
    d_label="$\delta^{13}$C",        # Name of isotope delta
    d_scale="VPDB",             # Isotope scale. End of plot labels
    r=0.0112372,  # VPDB C13/C12 ratio https://www-pub.iaea.org/MTCD/publications/PDF/te_825_
)

# add species
Species(name="CO2", element=C)  # Name & element handle
Species(name="DIC", element=C)
Species(name="OM", element=C)
Species(name="CaCO3", element=C)
```

1.4 Using external data to initialize a signal

We can use an external csv file to create a signal. The first column contains the time coordinates, the second the flux rate, and the third the delta value of the flux. The first row must contain a header. All values will be interpolated to fit the model time resolution.

Signals can also by created by specifying a signal type. At present the class understands, square, and pyramidal signal forms, as well as repetition. Signal can be added to each other (i.e., you can specify a signal which effects the flux, and then add another signal which effects the isotope ratio).

```
Signal(name = "ACR",    # Signal name
       species = CO2,  # Species
```

```
        duration = 100, # must match what is in the file
        filename = "test-data.csv" # filename
)
```

Once a signal instance has been created, it can be passed to a connector object in order to associate it with a flux (see the first connection below as an example).

1.5 Sources, Sinks and Reservoirs

```
Source(name="Fossil_Fuel_Burning", species=CO2)
Source(name="Carbonate_Weathering", species=CO2)
Source(name="Organic_Weathering", species=CO2)
Source(name="Volcanic", species=CO2)
Sink(name="Carbonate_burial", species=CaCO3)
Sink(name="OM_burial", species=OM)

Reservoir(
    name="Ocean",        # Name of reservoir
    species=DIC,         # Species handle
    delta=0,             # initial delta
    concentration=2.62, # cocentration
    unit="mmol",         # mass unit
    volume=1.332E18,     # reservoir size (m^3)
)
```

1.6 Connecting sources, reservoirs and sinks

Now that all model elements are specified, we can connect everything. Note how the previously specified `ACR` signal is added to the fossil fuel burning source. If the flux rate and delta are provided, the flux is treated a static. If the delta is omitted, the flux delta is driven by the upstream reservoir. If the flux is omitted, the flux is set in such a way that it maintains the mass in the reservoir. If the connection specifies a fractionation factor, the flux delta is function of the upstream reservoir delta plus the fractionation factor (`OM_burial`). Other processes like concentration dependent fluxes will be available soon.

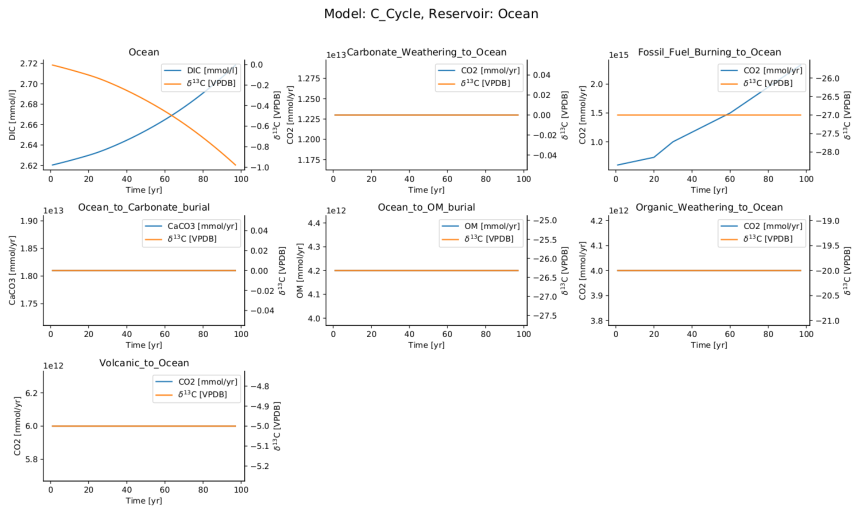
Fluxes can be circular, care must however be taken in which sequence they are defined.

```
# connect source to reservoir
Connect(
    source=Fossil_Fuel_Burning,  # source of flux
    sink=Ocean,          # target of flux
    rate=0,              # weathering flux in
    delta=0,             # set a default flux
    pl=[ACR],            # process list, here the anthropogenic carbon release
)

Connect(
    source=Carbonate_Weathering,  # source of flux
    sink=Ocean,         # target of flux
    rate=12.3E12,       # weathering flux in
    delta=0,            # isotope ratio
)

Connect(
    source=Organic_Weathering,  # source of flux
    sink=Ocean,         # target of flux
    rate=4.0E12,        # flux rate
    delta=-20,          # isotope ratio

)

Connect(
    source=Volcanic,    # source of flux
    sink=Ocean,         # target of flux
    rate=6.0E12,        # flux rate
    delta=-5,           # isotope ratio
)

Connect(
    source=Ocean,       # source of flux
    sink=OM_burial,     # target of flux
    rate=4.2E12,        # burial rate
    alpha=-26.32,       # fractionation factor
)

Connect(
    source=Ocean,       # source of flux
    sink=Carbonate_burial,     # target of flux
    rate=18.1E12,              # burial rate
    alpha=0,            # set the istope fractionation
)
```

1.7 Running the model

The model is executed via the `run()` method. The results can be displayed withe the `plot_data()` method which will generate an overview graph for each reservoir. Export of the results to a csv file is done via the `save_data()` method which will create csv file for each reservoir.

```
# Run the model
C_Cycle.run()

# plot the results
C_Cycle.plot_data()
# save the results
C_Cycle.save_data()
```



2 Controlling the flux type

The connect method has a variety of way to specify the flux:

- If both **rate** and **delta** are given, the flux is treated as a fixed flux with a given isotope ratio. This is usually the case for most source objects (they can still be affected by a signal, see above), but makes little sense for reservoirs and sinks.

- If both the `rate` and `alpha` are given, the flux rate is fixed (subject to any signals), but the isotopic ratio of the output flux depends on the isotopic ratio of the upstream reservoir plus and isotopic offset specified by `alpha`. This is typically the case for fluxes which include an isotopic fractionation (i.e., pyrite burial). This combination is not particularly useful for source objects.

- If the connection specifies only `delta` the flux is treated as a variable flux which is computed in such a way that the reservoir maintains steady state with respect to it's mass.

- If the connection specifies only `rate` the flux is treated as a fixed flux which is computed in such a way that the reservoir maintains steady state with respect to it's isotope ratio.

- Other connection types are possible, but currently untested.

- A flux can be scaled relative to another flux. To do so, provide the `scale` keyword, and a flux reference via the `ref` keyword

- A flux can be scaled relative to the mass or concentration of the upstream reservoir. This is achieved by providing a `k_mass` or a `k_concentration` keyword with an appropriate k-value. If one additionally provides the `ref_value` keyword, the mass/concentration is first normalized, then mapped to zero, and the difference is being scale by the k-value

  F = (M/M0 -1) * k where F is forced to be >=0

  This allows to scale the rate dependence relative to an equilibrium value M0, i.e. when the reservoir approaches M0, the flux -> zero. The k-value then expresses how fast the system returns to the equilibrium value

- Fluxes can be scaled with a Michalis-Menten type scaling function

  F = F * a * F0 x C/(b+C)

  this type of scaling is invoked by providing the following keywords: `a_value`, `b_value`, `ref_value` the latter being F0 in the above equation

- Last but not least, Reservoirs can be specified by either providing `volume` and `concentration` or `volume` and `mass`