The biogeochemichal model data base `bgc_md2` and python packages `LAPM`, `CompartmentalSystems`, `ComputabilityGraphs`for the analysis of compartmental dynamical systems

Markus Müller[1], Holger Metzler[1], Verónika Ceballos-Núñez[1], and Carlos A. Sierra[1]

[1]Max Planck Institute for Biogeochemistry, Hans-Knöll-Str. 10, 07745 Jena, Germany

July 27, 2023

Abstract

Compartmental systems are a particular class of dynamical systems that describe the flow of conserved quantities such as mass and energy through a network of interconnected compartments. The main purpose and greatest challenge of this work is to make such models **transparent** by facilitating comparisons between them. To this end we enhance the common diagnostics of pool contents and fluxes by the implementing the computations of age and transit time distributions for all models. We create a language to describe different models in different ways with respect to common eqivalent building blocks, and start and extendable collection of models. We can compare not only what is formulated in the model description but what is **computable**. To this end we developed three python packages for the mathematical analysis and computation of system-level metrics for distinct groups of compartmental systems. `LAPM`(Linear Autonomous Pool Models) provides functions for the analysis of compartmental systems at equilibrium. `CompartmentalSystems`tools for the analysis of non-autonomous linear compartmental systems. To study properties of groups of models and make comparisons among them, we developed the package `bgc_md2` (Biogeochemical Model Database), motivated by our own work on the terrestrial carbon cycle. `ComputabilityGraphs`can determine the set of all possible computations that can be performed on a model depending on information available from its description, allowing to query `bgc_md2`'s collection of models. The implemented analysis tools include symbolic computations of model decomposition into subsystems, flowdiagrams, transformation and reformulation with respect to different building blocks e.g. matrix versus flux equations and also difficult to obtain numerical metrics to characterize timescales of mass flow in compartmental systems such as the age of the mass and transit time trough the entire system, subsystems like vegetation or soil, or individual compartments. Together, these packages offer advanced integrated tools for the symbolic and numerical characterization of compartmental dynamical systems.

1 Motivation and significance

The principle of mass conservation plays a central role in mathematical models of natural systems in a variety of scientific fields such as systems biology, toxicology, pharmacokinetics (**?**), ecology (**????**), hydrology (**???**), biogeochemistry (**??**), and epidemiology (**?**). In most cases such models are nonnegative dynamical systems that can be described by first-order systems of ordinary differential equations (ODEs) with strong structural constraints. Such systems are called compartmental systems (**????**), and have important mathematical properties that aid in their analysis and study.

As mass moves across a compartmental system, it is often of interest to study properties of the compartments, the entire system or subsystems related to the speed at which the mass moves, and the time it takes mass to pass through specific paths. These properties are generally characterized by metrics such as the age of mass with respect to the entry to the entire system, a subsystem, or a single compartment and the transit time of mass across the entire network of compartments or parts of it until its final exit. However, there are different methods to compute these metrics from compartmental systems depending on specific assumptions imposed on the system of equations (**?**). and available computational procedures may differ largely depending on specific assumptions.

To aid in the analysis of compartmental systems, and to determine the type of computational methods that can be performed for particular systems under given assumptions, we developed four python packages for their representation, classification, and analysis. These open source packages are called: `LAPM`(Linear Autonomous Pool Models), `CompartmentalSystems`, `bgc_md2` (Biogeochemical Model Database), and `ComputabilityGraphs`.

This manuscript provides a general introduction to these four python packages, with emphasis of their combined use via `bgc_md2` , and aims at providing a general guidance for their use, installation, and modification. We provide an example based on the global carbon cycle, but the packages can be used for a large variety of systems in which mass or energy conservation is required.

2 Conceptual framework

2.1 Definition and classification of compartmental systems

Compartmental systems describe the dynamics of a vector fo mass contents x of n kinetically homogeneous compartments. Because mass is a non-negative quantity, this vector of mass contents can only occupy the non-negative orthant of the state-space; i.e. $x \in \mathbb{R}^n_+$. A compartmental system generally receives mass from outside the system according to a vector of mass inputs $u \in \mathbb{R}^n_+$, and this mass is transferred among compartments and released back to the external environment according to rates encoded in a compartmental matrix $B \in \mathbb{R}^{n\times n}$. Therefore, we can write the dynamics of a compartmental system as a set of ordinary differential equations of the form

$$\frac{dx}{dt} = \dot{x} = u(x,t) + B(x,t)\,x(t) \tag{1}$$

The key property of compartmental systems is, in order for the system to balance mass, that the square matrix $B = (B_{ij})$ exhibits three main properties

1. $B_{ii} \leq 0$ for all i,
2. $B_{ij} \geq 0$ for all $i \neq j$, and

3. $\sum_i B_{ij} \leq 0$ for all j.

Then, B is called *compartmental* and governs all internal cycling of material as well as the exit of material from the system.

We distinguish between different types of compartmental systems, according to linearity and autonomy. If the vector of inputs and the compartmental matrix depend on the vector of states in system (1), we call it non-linear, and linear otherwise. Similarly, if the vector of inputs and the compartmental matrix depend on time, we call the system non-autonomous, and autonomous otherwise.

2.2 System level metrics: age, transit time, and entropy

Compartmental systems can be described by a set of metrics that characterize system level properties. Ages, transit times, and entropies are key quantities of compartmental systems that can be considered to better understand underlying system dynamics and to compare models with different sizes or structures. While age describes how old material in the system is, transit time describes how long material needs to travel through the entire system from entry to exit (**??**). These quantities provide us with information about the timescales at which systems operate and respond to perturbations.

The Shannon information entropy can be used as a complexity measure to characterize the complexity of dynamical systems (**?**). It can be used to describe the uncertainty of a particle's path through a compartmental system, quantifying how difficult it is to predict this path. It can be used for comparing path properties of models with different number of compartments and connections among them (**?**).

2.3 Compartmental systems in equilibrium

The concept of equilibrium is restricted to autonomous systems. It does not even make sense to ask the question otherwies. If the autonomous system is nonlinear it is possible but not certain that an equilibrium exists. The only case where we can expect an equilibrium are linear, autonomous, pool models,

$$\dot{x} = u + B\,x, \qquad \text{with} \quad x(t_0) = x_0. \tag{2}$$

for which interesting properties can be obtained by the `LAPM`package. The equilibrium x^* is defined by the condition $\dot{x^*} = 0$ which translates to $-Bx* = u$ which means that for pool contents $x*$ the influxes u match the outfluxes $Bx*$ exactly. It is straightforward to see that for this to happen all pools with influxes must be connected (possibly via other pools) to an outflux of the system, and that the (constant) rates for all the flux rates out of all pools along this paths are greater than zero, since an input receiving pool p without these conditions would necessarily grow over time, violation the equilibrium condition $\dot{x*}_p = 0$. Interestingly these conditions also gurantee that B is invertable and the equilibrium therefore uniquely determined by:

$$x^* = -{B^*}^{-1}\,u^*. \tag{3}$$

Although at different times different material moves through the system, the size of the pools does not depend on time if the system is in equilibrium: $x(t) = x^*$ This is also true for other properties such as the age distribution of mass in particular compartments and in the entire system and the transit time distribution, which is defined as the time it take masses in the input flux to appear in the output flux. Although the material moving through the system does change the amount, age and transit time distributions do not. They are in fact characterized by the Phase Type distribution, which depends on compartmental matrix B and the equilibrium solution x^* for the system age distribution and the B and the input u for the transit time distribution (**?**). Compartmental systems at equilibrium have similar properties as continuous-time absorbing Markov Chains (**?**). Therefore, we can obtain other quantities of interest such as the path entropy of particles that travel across the system and the occupation time of particles inside compartments (**?**). These properties of linear autonomous compartmental systems at equilibrium can be obtained with the `LAPM`package.

Interestingly the properties of linear autonomous systems in equilibrium can also be computed for nonlinear systems in equilibrium if such an equilibrium exists.

$$\dot{x} = u(x) + B(x)\,x, \qquad \text{with} \quad x(t_0) = x_0, \tag{4}$$

In equilibrium the system is indistinguishable from a linear autonomous one

$$0 = \dot{x^*} = u^* + B^*\,x^* \tag{5}$$

with $B^* = B(x^*)$ and $u^* = u(x^*)$ Therefore also nonlinear autonomous systems at equilbrium can be analyzed with the by the `LAPM`package. Note however,that we no such x^* might not exist for some systems while others may have multiple equilibrium solutions and the age and transittime distribution may be very different for these different equilibria.

2.4 Time evolution along a trajectory

We consider now linear non-autonomous systems of the form

$$\dot{x}(t) = u(t) + B(t)\,x, \qquad \text{with} \quad x(t_0) = x_0. \tag{6}$$

In this case, the inputs and the compartmental matrix are time-dependent and the system never converges to a fixed-point solution. In most cases, an analytical solution cannot be obtained, but the solution can be obtained numerically. In particular, the solution for systems of the form of equation (6) can be written as

$$x(t, t_0, x_0) = \Phi(t, t_0)x_0 + \int_{t_0}^{t} \Phi(t, \tau)u(\tau)\mathrm{d}\tau. \tag{7}$$

The state transition operator $\Phi(t, t_0)$ is a matrix-valued function that multiplied with the state x_0 at t_0 transitions it to the state $x(t)$ subsequent time $t > t_0$. It is numerically computable by solving an matrix ode derived from (2.4) From $\Phi(t, t_0)$ we can obtain not only the temporal evolution of the solution $x(t)$ but also of the distributions of ages of the mass in the compartments and in the entire system (**?**). The `CompartmentalSystems`package provides all the functionality necessary to do these computations, which rely on a description of the time-dependent input vector $u(t)$ and the compartmental matrix $B(t)$, as well as initial age distributions for the compartments.

Furthermore, these computations can also be obtained for nonlinear systems of the form

$$\dot{x}(t) = u(t, x) + B(t, x)\,x(t), \qquad \text{with} \quad x(t_0) = x_0. \tag{8}$$

by numerically solving (8) and plugging the solution $x(t, t_0, x_0)$ back into it, which results in $\tilde{B}(t) = B(t, x(t, t_0, x_0))$ and $\tilde{u}(t) = u(t, x(t, t_0, x_0)$ i.e a linear system in the form (6). Therefore, age and transit time distributions can be obtained for nonlinear non-autonomous systems along the specific trajectory. Detailed methods for the computation are provided in **?**

3 Software description

3.1 `LAPM`

Linear Autonomous Pool Models (`LAPM`) is a python 3 package for the study of autonomous compartmental systems at equilibrium such as those described in section 2.3. It provides access to the LinearAutonomousPoolModel class, with methods for the symbolic and numerical solutions of the steady state content in the compartments, and the steady state release out of the compartments. For transit time, system age, and pool age, it provides symbolic and numerical computations of distribution densities, cumulative distribution functions, mean, standard deviation, variance, higher order moments, and Laplace transforms.

For the analysis of compartmental systems in analogy to absorbing Markov chains, `LAPM`provides methods for the computation of the entropy rate per jump, the entropy rate per unit time, and path entropy. It provides the class DTMC (discrete-time Markov chains), with methods to compute the fundamental matrix, stationary distribution, and expected number of jumps of the Markov chain.

3.2 `CompartmentalSystems`

This package deals with non-equilibrium trajectories of compartmental systems. In particular, it provides the class smooth_reservoir_model to describe symbolically the general class of non-autonomous nonlinear compartmental dynamical systems of equation (1). It does not require code for numerical computations or model simulations, but rather defines the underlying structure of the respective model. All fluxes or matrix entries are expected to be SymPy expressions.

To obtain numerical results, the package provides the class smooth_model_run, which is initialized with the initial conditions of the system of equations, a set of parameter values, and a time sequence. It computes the solution trajectory for the given initial conditions and parameter values , finds the corresponding linear system with the same solution following the strategy described in section 2.4 computes the state transition operator $\Phi(t, t_0)$ for these solution trajectoriess and provides methods to obtain time dependent densities with corresponding moments and quantiles for system age, compartment age, and transit time.

An additional module provides functions to obtain initial age distributions required for the computation of time-dependent age distributions. This module relies uses `LAPM`.

3.3 The biogeochemical model database `bgc_md2`

This package can be seen as a frontend to `CompartmentalSystems`and `LAPM`and provides:

1. Datatypes defining **building blocks** of models e.g. `CompartmentalMatrix`, `InternalFluxesBySymbol`, ...
2. Functions operating on those properties (forming the edges of the graph where the Datatypes are nodes)
3. A user interface based on graph algorithms to
   (a) compute the set of computable properties (e.g. the comparable criteria for a set of models, database queries)
   (b) actually compute the desired properties by recursively connecting several function applications.
   (c) show what is missing to compute a desired property.

4. 30+ vegetation, soil or ecosystem models for carbon and nitrogen cycling as reusable python modules using the building blocks in a flexible way.

5. An interface to *many algorithms* in `CompartmentalSystems` to compute diagnostic variables for *many models* in `bgc_md2`.

The software simplifies the abstract description of biogeochemical cycling models using SymPy's symbolic mathematical representation, and serves as an wrapper for `LAPM`or `CompartmentalSystems`. Its main use is to make complex models immediately available for further investigation e.g. in Jupyter notebooks, but it also provides The package assists in the creation of reports in the form of Jupyter notebooks, either for a single model, or for comparing sets of models. In the first case the role of the `bgc_md2` package is to guide the author of a particular notebook concerned with a particular model by using a computability graph to show what results can be computed, given the information already present in the model initialization; or to show which additional information has to be provided in the model description to be able to obtain a desired result.

Model descriptions are provided in a 'source.py' file for each model.

3.4 Installation, documentation and demonstrations

The three packages are managed under version control in GitHub, and are publicly available for their evaluation, use, and further modification. They can be obtained using standard procedures for cloning and fetching Git repositories.

The three packages can be installed following the instructions provided for the installation of `bgc_md2`, which depends on `LAPM`and `CompartmentalSystems`, and therefore its installation script installs all packages at once. To install the packages, users must create a Python virtual environment using Conda and use the install script provided, which will take care that all dependencies required are properly installed. After installation, we strongly recommend users to run all test stored in the folder 'tests'. If they run successfully, then all the functionality of the three packages is readily accesible.

Detailed instructions of the installation procedure can be found in the 'README.md' file of the `bgc_md2` repository.

We use docstrings to document modules, functions, classes and methods. These docstrings are snippets of human readable text included with the source code of the package, and are processed by Sphynx, a documentation generator for python code. The formatted documentation is served by GitHub and it's automatically updated when changes to the documentation are pushed to the master branch of the repositories.

In addition to this documentation, we provide Jupyter notebooks for each package, with demonstrations of specific aspects of the available functionality. Each package has a 'notebooks' folder where the Jupyter notebooks are stored. However, they are stored as source .py files and not in the original .ipynb format. The reason for this is that to maintain the notebooks under version control and avoid conflicts among different versions on different machines, we store the notebooks as regular python files. To convert these files to Jupyter notebooks, we recommend to use Jupytext, which does the translation with a simple 'convert' command.

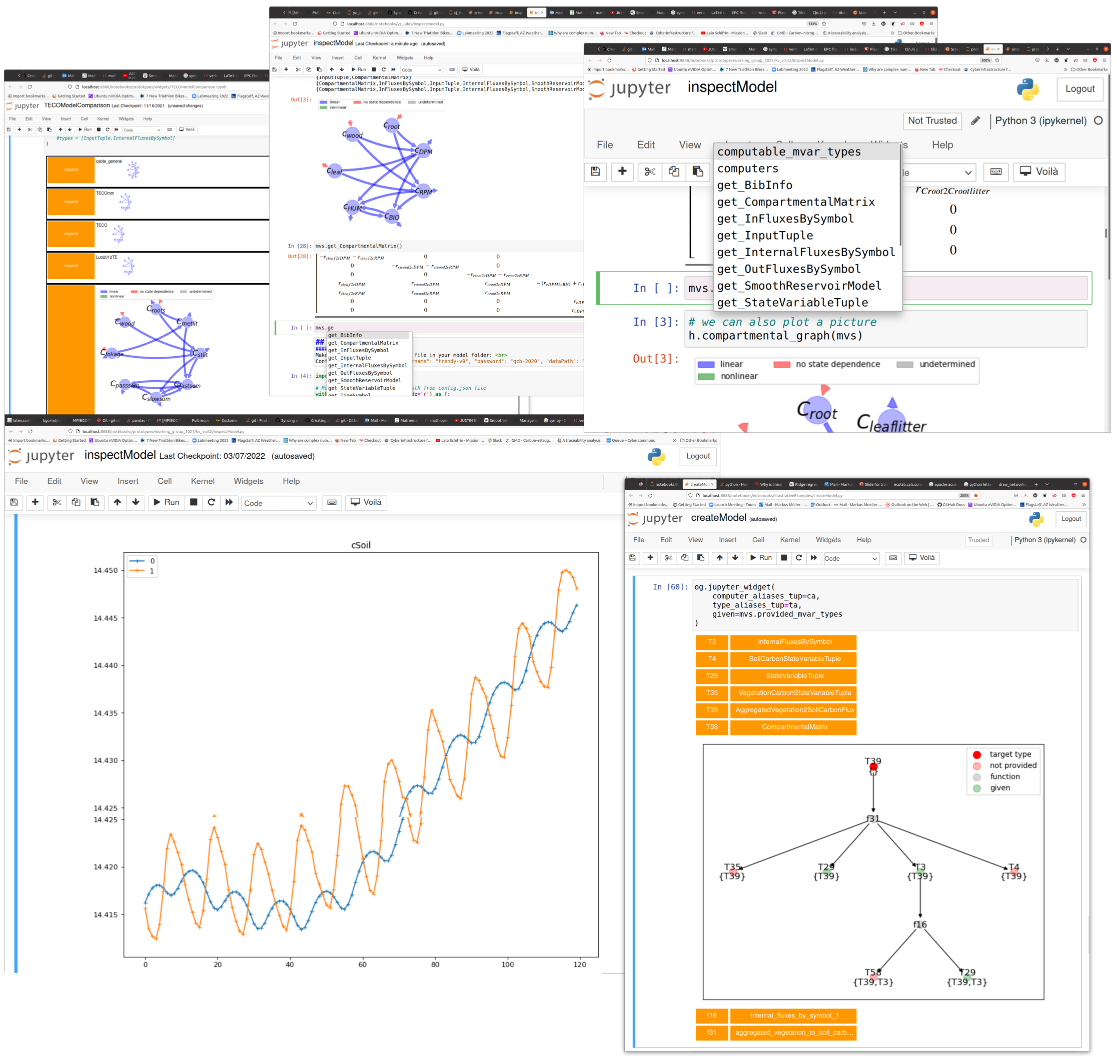
4 `ComputabilityGraphs`

5 Illustrative example

We provide an example notebook in https://github.com/MPIBGC-TEE/bgc˙md2/tree/master/notebooks/illustrative
It shows the interplay of all three packages on the collection of predefined models in `bgc_md2` and ways to extend this collection by defining a new model that can be compared with respect to diagnostic variables that are computed using `LAPM`and `CompartmentalSystems`. The best way to use the example is to install the packages and explore the example interactively.

6 Impact

7 Conclusions



```
In [ ]: mvs.
        computable_mvar_types
        computers
        get_BibInfo
        get_CompartmentalMatrix
        get_InFluxesBySymbol
        get_InputTuple
        get_InternalFluxesBySymbol
        get_OutFluxesBySymbol
        get_SmoothReservoirModel
        get_StateVariableTuple

In [3]: # we can also plot a picture
        h.compartmental_graph(mvs)
Out[3]:
```

```
In [60]: og.jupyter_widget(
             computer_aliases_tup=ca,
             type_aliases_tup=ta,
             given=mvs.provided_mvar_types
         )
```

jupyter inspectModel
Not Trusted
Python 3 (ipykernel)
Logout
Voilà
linear
nonlinear
no state dependence
undetermined
C_{root}
$C_{leaflitter}$
$r_{C_{root2}C_{rootlitter}}$
jupyter inspectModel Last Checkpoint: 03/07/2022 (autosaved)
cSoil
0
1
jupyter createModel (autosaved)
T3 InternalFluxesBySymbol
T4 SoilCarbonStateVariableTuple
T29 StateVariableTuple
T35 VegetationCarbonStateVariableTuple
T39 AggregatedVegetation2SoilCarbonFlux
T58 CompartmentalMatrix
target type
not provided
function
given
T39
f31
T35 {T39}
T29 {T39}
T3 {T39}
T4 {T39}
f16
T58 {T39,T3}
T29 {T39,T3}
f16 internal_fluxes_by_symbol_1
f31 aggregated_vegetation_to_soil_carb...
jupyter TECOModelComparison

Figure 1: Figure description, top row, left to right: Interactive jupiter widget with a table of models (orange buttons can be clicked to expand or collape a more detailed view of the particular model), Model inspection with pool connection graph, which can be derived from the symbolic description along other symbolic properties as flux equations and the compartmental matrix, Zoom into IPython/Jupyter UI, showing methods automatically added by the computability graph library.
Bottom row: Data assimilation with an automatically created numeric model (from symbolic description), Computability graph for a desired diagnostic (aggregated Flux from the vegetation to soil part, showing that the additionally needed information to compute the desired result)

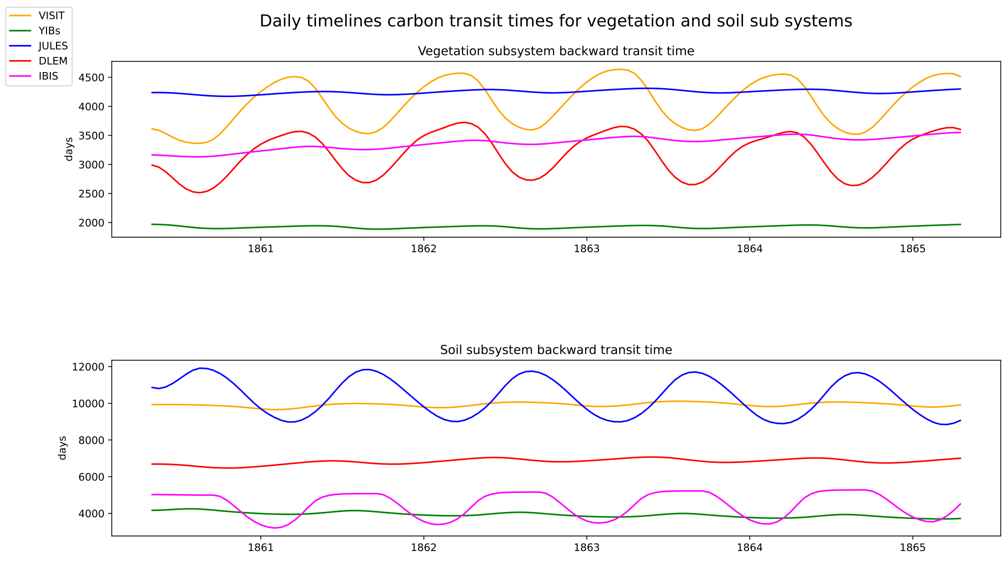


Figure 2: Figure above: Comparing the (real = transient) backward transit time through subsystems, accross different models.

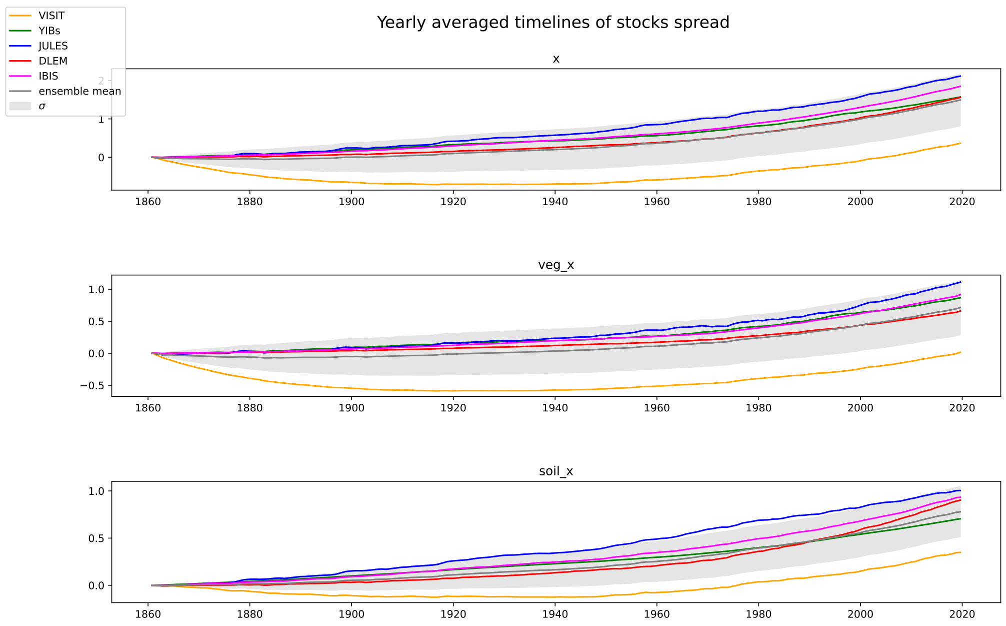
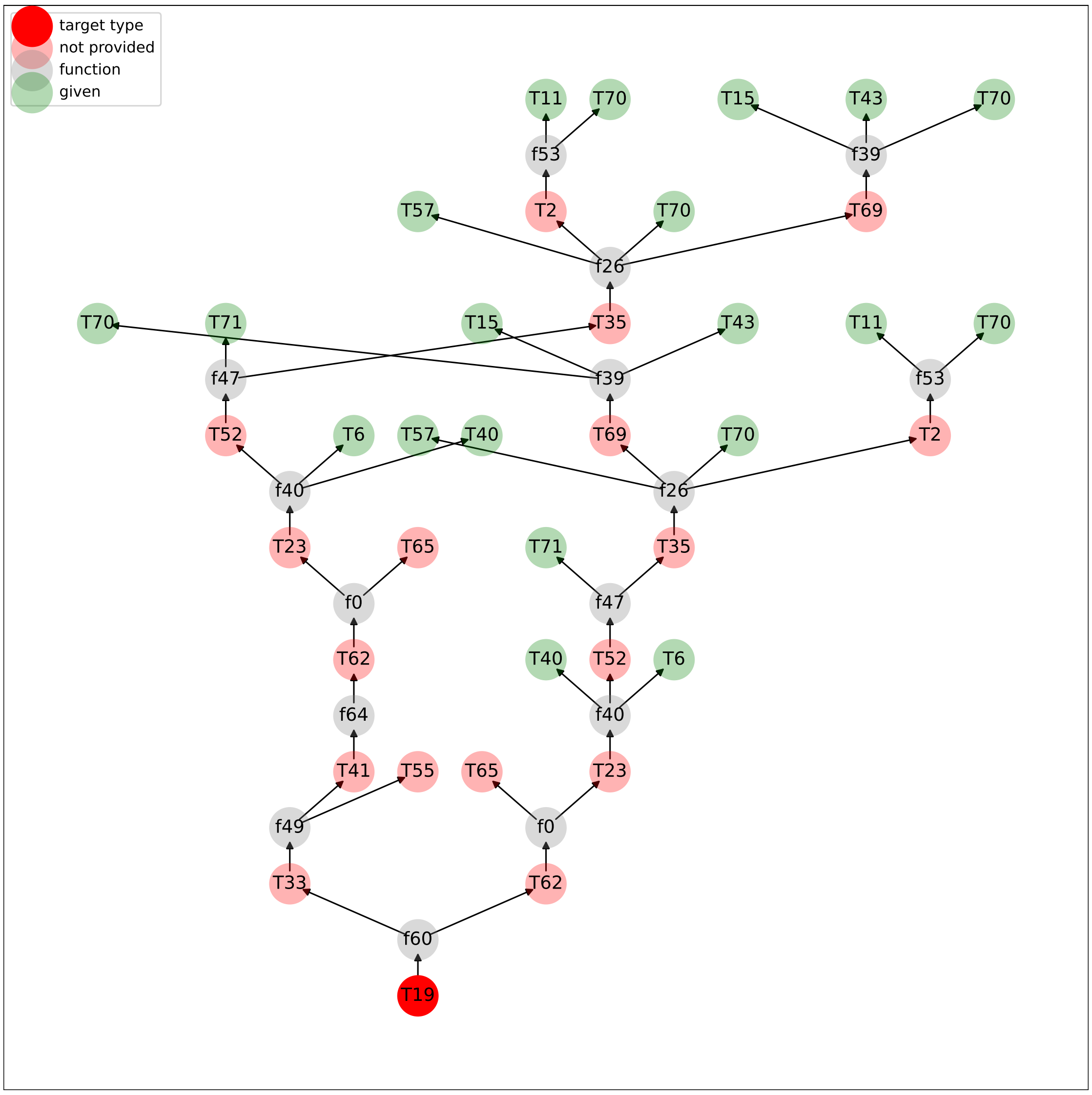


Figure 3: Figure above: Total Carbon stock and subsystem stock development for different models. `bgc_md` only needs to be told which pools belong to which subsystem.



T1	InFluxesBySymbol
T5	VegetationCarbonSmoothModelRun
T8	CompartmentalMatrix
T9	SmoothModelRun
T12	StateVariableTuple
T13	NumericParameterization
T18	InputTuple
T23	NumericSimulationTimes
T26	OutFluxesBySymbol
T35	NumericVegetationCarbonMeanBackwardTransitTimeSolution
T38	VegetationCarbonStateVariableTuple
T39	SmoothReservoirModel
T43	TimeSymbol
T49	NumericVegetationCarbonStartMeanAgeTuple
T54	NumericStartValueArray
T55	NumericParameterizedVegetationCarbonSmoothReservoirModel
T66	NumericParameterizedSmoothReservoirModel
T68	InternalFluxesBySymbol
T70	StartConditionMaker