Key Points:

- enter point 1 here
- enter point 2 here
- enter point 3 here

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

2]Markus Müller 4]Holger Metzler 3]Verónika Ceballos-Núñez 2]Konstiantin Viatkin 2]Yiqi Luo [1]Max Planck Institute for Biogeochemistry, Hans-Knöll-Str. 10, 07745 Jena, Germany [2]Cornell [3] [4]

Corresponding author: Markus Müller, `markus.mueller.1.g@googlemail.com`

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. Firstly the scientific challenge is to find the common diagnostics by which we can compare models. The technical challenge is how to implement them in a way to make them applicable to many models and thirdly how to describe models compactly enough to facilitate a maintainable collection of reasonable size and flexible enough to formulate models in the many different ways in which they appear in the literature. To address the first question we enhance the common diagnostics of pool contents and fluxes by the the computations of age and transit time distributions for whole models and especially subsystems like vegation or soil which is essential to be able to compare models with conceptually different pools, i.e. two ecosystem models will likely both have a vegetation part , however consist it might consist of two pools (i.e. leaf and root) in one but three pools (i.e. leaf, wood, root) in the other. To address the technical challenge we avoid duplication in two very rigorous and novel ways: We provide common eqivalent building blocks for models implemented in symbolic math via `sympy` as special types, and a set of type annotated functions operating on these types as building blocks which can be combined by a graph library to compute every possible result, reachable by any recursive combination of these functions. This allows us to formulate models compactly and flexibly in different equivalent ways i.e. via fluxes, flowdiagrams, or matrices and at the same time to avoid the implementation of many of similar functions for the same result. Thus we can also not only query and compare what is provided, as in the model description records of a conventional data base, but also what is **computable** from them. The combination of these capabilities is expressed in the package `bgc_md2` (Biogeochemical Model Database), which apart from the tools mentioned above also represents a small (30+) collection of models, motivated by our own work on the terrestrial carbon cycle. As a proof of concept of this approach we use `bgc_md2` to reconstruct four models of the trendy9 model intercomparison, symbolicly, guess some parameters run them with the trendy9 driver data and compare them with respect to transient mean ages and transit times of carbon trough the vegetation and soil subsystems. This presentation concentrates on the software and only links to the mathematical foundations which are published already or in preparation.

Plain Language Summary

Enter your Plain Language Summary here or delete this section. Here are instructions on writing a Plain Language Summary: https://www.agu.org/Share-and-Advocate/Share/Community/Plain-language-summary

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 prop-

erties 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

The most parsimonious and general description of a compartmental system is a graph in the mathematical sense, which is a tupel of two sets, the set of nodes, and the set of edges. In the particular graphs that describe compartmental systems the compartments or pools form the nodes and the fluxes between them and the exterior the edges. The fluxes are allowed to be functions of time and the contents of the pools. This is usually referred to as 'well mixed' or 'kinetically homogeneous' compartments. If we assume an arbitrary ordering of the pools we can represent the mass (content of the pools) by an ordered tuple $\mathbf{x}$ 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. $x \in \mathbb{R}^n_+$. Likewise the mass, that the system receives from outside is represented by a tuple of mass input fluxes (mass over time) $u \in \mathbb{R}^n_+$. It has been shown in (**?**) that for smoth enough fluxes, mass is transferred among compartments and released back to the external environment can be 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 $u^* = u(x^*)$ If the inverse B^{*-1} exists transit time and age distributions can be computed (**?**). and therefore also nonlinear autonomous systems at equilbrium can be analyzed with the by the `LAPM` package. Note however,that (3) is useless to determine x^* and no such x^* might exist for some systems while others may have multiple fixed points 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 The `Python` packages

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 implements 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 uses `LAPM` .

3.3 `ComputabilityGraphs`

This is a helper package, specifically developed for use in `bgc_md` but also usable in separation. It implements the class `CMTVS` which stands for **C**onnected **M** ulti **T**yped **V**ariable **S**et. Instances consist of a set of variables with unique type and a set of type annotated functions that exclusively use these types in their signature (as arguments or return values). This combination can be used to build graphs that we exploit mainly in the followng ways.

1. To compute which types of information (target types) are obtainable given the set of provided variables. Since the `CMTVS` knows the types of all its variables and the signatures of all available functions, it can iteratively add the result types of all applicable functions until no more result types can be reached. This is a forward graph search.
2. To find out what type of information is i.e. variable of which type are *missing* to obtain a target variable (backward search). Fig. **??** shows the bipartite dependency tree for a quite complex numerical result. printed by the `CMTVS` instance.
3. To compute the actual result of a targeted type. If a result type is in the computable set determined by 1 then The search tree created under 2 can be traversed in reverse starting at the given nodes and ending in the final result.

Using 1 and 3 together a `CMTVS` can add get methods for computable results dynamically. This is very useful for for the user interface in interactive python sessions, including jupyter notebooks since on pressing the tab key, the python interpreter suggest available method calls as autocompletion option for any object followed by a "." For an `CMTVS` object these methods become more numerous automatically as more and more information is added to it. Apart from the user interface 1 is also useful for queries. If we have a (large) set of different `CMTVS` objects and want to compare them with respect to a certain propertie, we should first find out for which of them this propertie is actually available.

3.4 The biogeochemical model database `bgc_md2`

This is the central package and can be seen as a frontend to `CompartmentalSystems` and `LAPM` facilitated by `ComputabilityGraphs` . It can be used as a library [1] that provides:

1. Datatypes defining **building blocks and comparable results** specifically tailored to compartmental models. e.g. `CompartmentalMatrix`, `InternalFluxesBySymbol`, `NumericVegetationCarbonMeanBackwardTransitTimeSolution` ... Many of these types are based on a symbolic mathematical representation, using `SymPy` , so that many symbolic results are computable without the need to know parameters or data to run the models. Using the underlying graph representation as sets of pools and fluxes, we can reorder pools and thereby automatically transform the compartmental matrices, group them into different subsets (e.g. vegetation, soil, litter, carbon, nitrogen ...), substitute pool names, get mathematical expression for cumulative fluxes e.g. from vegetation to soil or simply plot the graphs. Using this symbolic transformations we can compare models that might have looked very different initially, simply due to arbitrary choices of pool names or their even more arbitrary order. comment:
green brown graph or link to the compare TICO notebook
2. Type annotated functions operating on those types (forming the edges of the graph where the Datatypes are nodes)
3. 30+ vegetation, soil or ecosystem models for carbon and nitrogen cycling as reusable python modules using the building blocks in a flexible way.
4. An interface to *many algorithms* in `CompartmentalSystems` to compute diagnostic variables for *many models* in `bgc_md2`.

It is impossible to fully exhibit the potential of this approach without examples. To this end we created some illustrative `jupyter` notebooks that are availble via binder without the need to install the packages. This paragraph contains only pointers to those much more elaborate notebooks and some example plots from them. The examples demonstrate how `bgc_md2`:

1. Symplifies and unifies the creation of new models from scratch while using the symbolic and graphic diagnostic capabilities to inspect it while we build it and use `ComputabilityGraphs` to point out what missing information we have to provide to make desired results computable. This is demonstrated in
2. How a model that is already part of the database can be inspected w.r.t complex diagnostics like the transient mean transit time that would take years to implement in stand alone model code but are now available for all models with sufficient information. This is demonstrated in
3. aided by `ComputabilityGraphs` 's ability to compute which properties are computable allows to query the collection of models that are already part of `bgc_md`. This is demonstrated in
4. Several Models for which numeric parameter values and driver data are available can be compared w.r.t. abstract properties. We chose the mean age and transit time of the vegetation and soil subsystems since they are defined for all models while the concrete pools of the models differ and not only have different names but also different meaning.

[1] rather than a framework, since there is no 'inversion of control'

4 Conclusions

Open Research Section

This section MUST contain a statement that describes where the data supporting the conclusions can be obtained. Data cannot be listed as "Available from authors" or stored solely in supporting information. Citations to archived data should be included in your reference list. Wiley will publish it as a separate section on the paper's page. Examples and complete information are here: https://www.agu.org/Publish with AGU/Publisusing SymPy'sh/Author Resources/Data for Authors

Acknowledgments
Enter acknowledgments here. This section is to acknowledge funding, thank colleagues, enter any secondary affiliations, and so on.

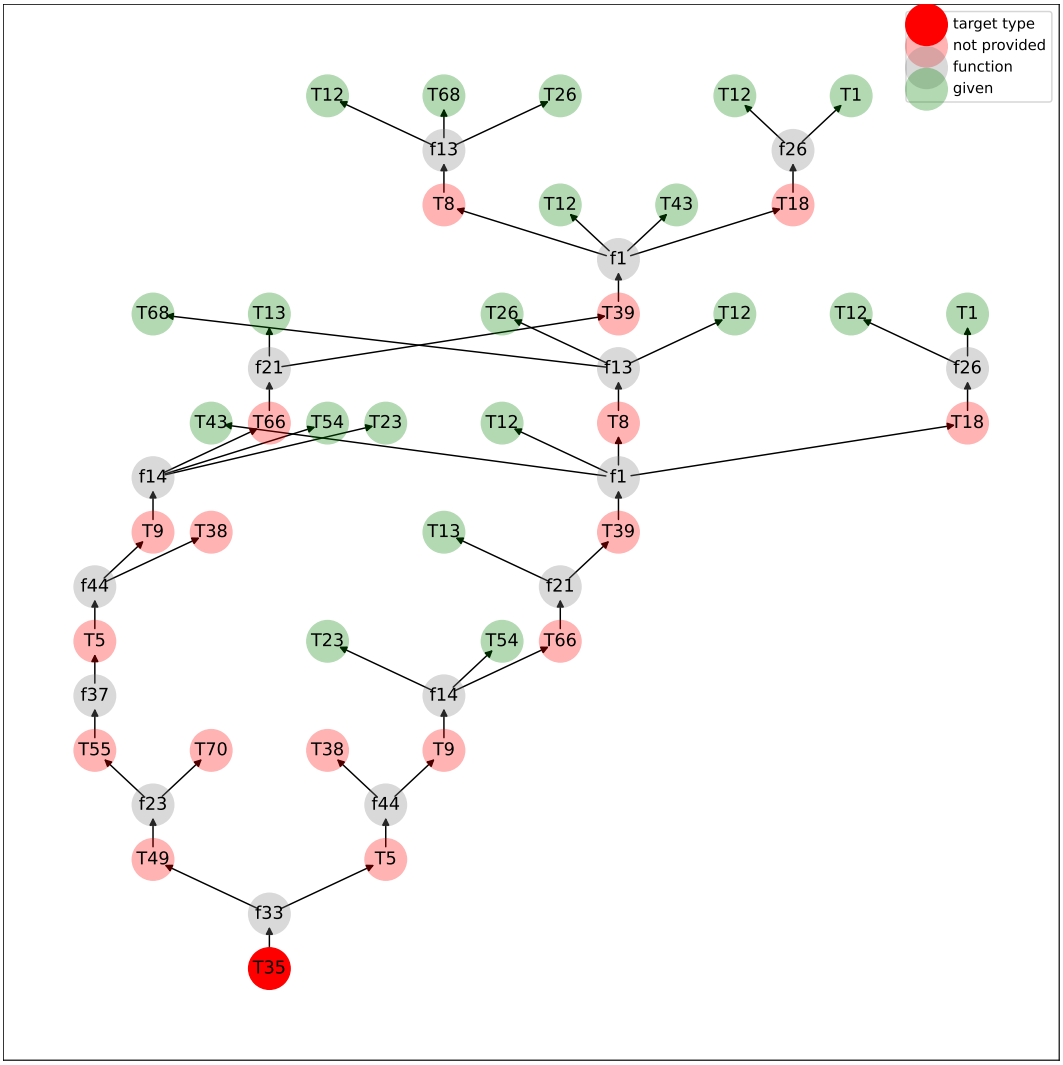

Figure 1: Dependency graph

The red node T35 at the bottom represents the target that we want to compute: `NumericVegetationCarbonMeanBackwardTransitTimeSolution` The green nodes are the building blocks that we have provided in the model source file. The light red nodes show things that are not provided but have to be computed via the function represented by the grey nodes. This is possible if all their (ultimate recursive) dependencies are green nodes. In this example this is the case except for nodes T70 and T38 `StartConditionMaker` and `VegetationCarbonStateVariableTuple` respectively

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

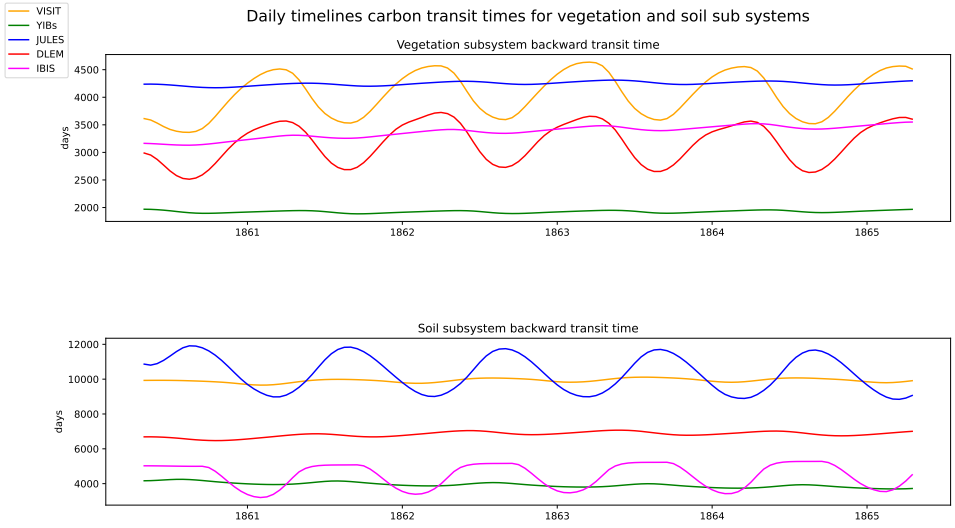

Figure 2: Figure above: Comparing the (real = transient) backward transit time through subsystems, accross different models.

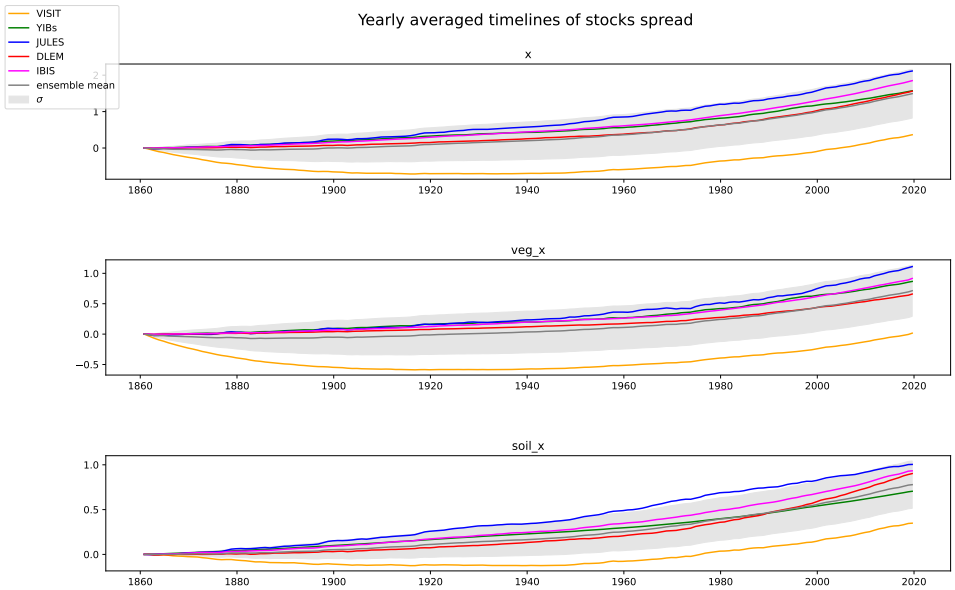

Figure 3: Figure above: Total Carbon stock and subsystem stock development for different models. `bgc_md` only needs to be told which pools belong to which subsystem.

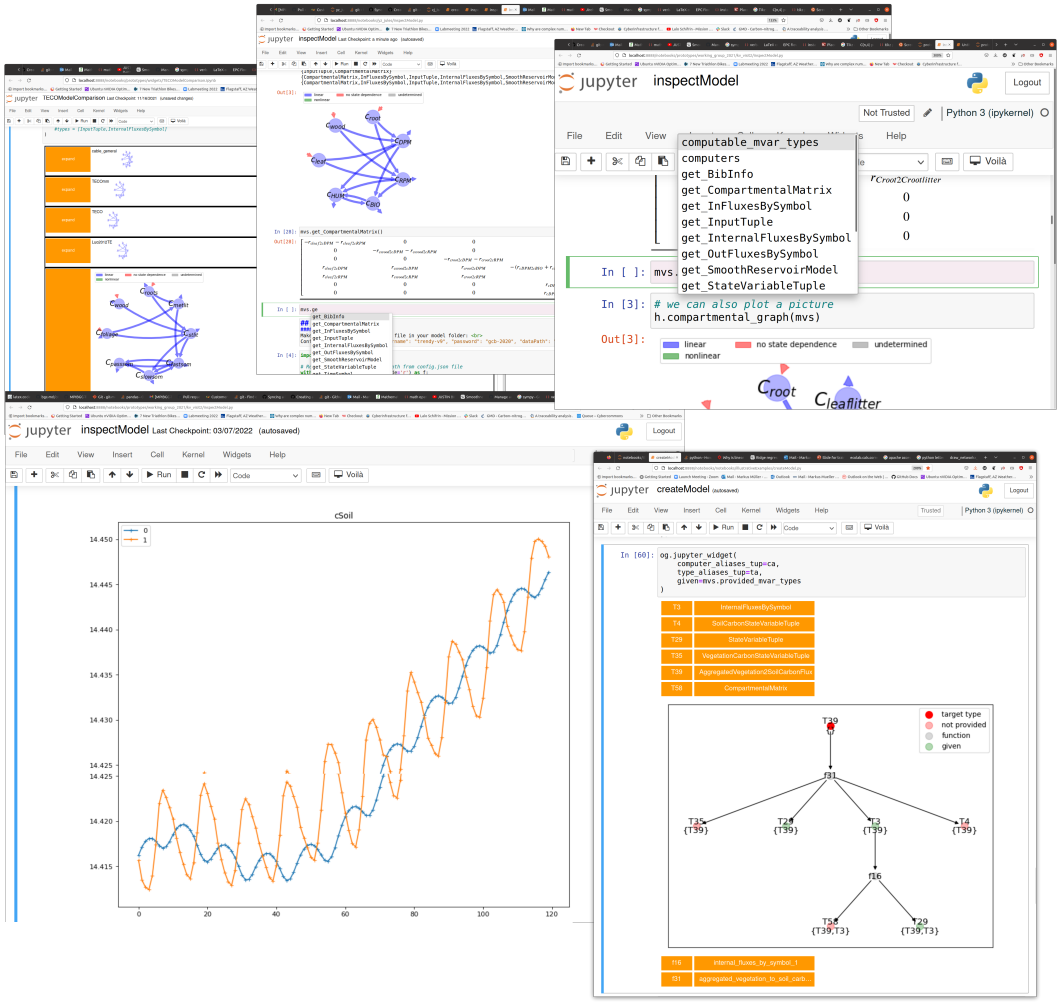

Figure 4: Figure description, top row, left to right: Interactive jupiter widget with a table of models (orange buttons can be clicked to expand or collape a more detailed view of the particular model), Model inspection with pool connection graph, which can be derived from the symbolic description along 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)