Geosci. Model Dev., 5, 1045–1060, 2012
www.geosci-model-dev.net/5/1045/2012/
doi:10.5194/gmd-5-1045-2012

Geoscientific Model Development

Models of soil organic matter decomposition: the SOILR package, version 1.0

C. A. Sierra, M. Müller, and S. E. Trumbore

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

Correspondence to: C. A. Sierra (csierra@bgc-jena.mpg.de)

Received: 29 March 2012 – Published in Geosci. Model Dev. Discuss.: 2 May 2012
Revised: 2 August 2012 – Accepted: 4 August 2012 – Published: 24 August 2012

Abstract. Soil organic matter decomposition is a very important process within the Earth system because it controls the rates of mineralization of carbon and other biogeochemical elements, determining their flux to the atmosphere and the hydrosphere. SOILR is a modeling framework that contains a library of functions and tools for modeling soil organic matter decomposition under the R environment for computing. It implements a variety of model structures and tools to represent carbon storage and release from soil organic matter. In SOILR, organic matter decomposition is represented as a linear system of ordinary differential equations that generalizes the structure of most compartment-based decomposition models. A variety of functions is also available to represent environmental effects on decomposition rates. This document presents the conceptual basis for the functions implemented in the package. It is complementary to the help pages released with the software.

1 Introduction

Soil organic matter decomposition is a fundamental process within the Earth system (Swift et al., 1979; Schlesinger, 1997; Jacobson, 2000). Through this process, carbon and other biogeochemical elements fixed by plants in the process of photosynthesis are transferred to the atmosphere and the hydrosphere in mineral form. This release of biogeochemical elements is fundamental for other processes in the Earth system, such as the global energy balance, with important consequences for climate. Soil organic matter decomposition is also a basic process for the availability of biogeochemical elements necessary for plant growth, therefore it has important consequences for agriculture and humanity.

Given the importance of soil organic matter decomposition, many models have been developed describing its dynamics (Manzoni and Porporato, 2009), but only few attempts have been made to synthesize them (e.g. Paustian et al., 1997; Wu and McGechan, 1998; Manzoni and Porporato, 2009). Although more than 250 different models of soil organic matter decomposition have been proposed since the 1930s, most of these models share common mathematical structures (Manzoni and Porporato, 2009). This suggests that it is possible to develop models that can generalize most of the models already proposed. In fact, Ågren and Bosatta (1998) have made important contributions to a general theory of organic matter decomposition with the development of the continuous quality theory.

In most models, soil organic matter is usually characterized by compartments with homogeneous decomposition rates, which in the continuous quality theory are approximated by a continuous function between a rank variable denoted as quality and the decomposition rate. The continuous quality approach introduces a high level of generality, but it also introduces limitations in terms of finding analytical solutions for complex representations of organic matter heterogeneity (Sierra et al., 2011). For this reason, the continuous quality theory has only been implemented to describe the average dynamics of soil organic matter decomposition. Furthermore, the description of microbial dynamics in the continuous quality theory lacks the generality needed to encompass different mathematical representations of microbial-substrate interactions.

A general theory of soil organic matter decomposition can benefit greatly from a synthesis of the different modeling approaches already proposed. It would help to identify model

structures that have been used frequently with certain degree of success to represent observed data.

Recently, Held (2005) has called to attention a growing gap between high-end simulations and theoretical understanding in climate modeling, which we believe also applies to Earth system modeling in general. Current models are highly complex and use sophisticated algorithms to represent different processes within the Earth system. However, it is difficult to obtain a basic understanding of system behavior from these models due to their complexity. Furthermore, these models include only one single set of functions to represent a specific process, which is equivalent to proposing one single hypothesis to explain system structure and function. Therefore, Held (2005) proposed the development of hierarchical models in which detailed models can be reduced in hierarchies that help to better understand system dynamics. On the one hand, the general model has a high level of abstraction and helps to elucidate basic properties of the system. On the other hand, detailed models are specific realizations of the general models that can help to predict system behavior under specific conditions, such as climate change or emission scenarios.

In this document we introduce SOILR, a modeling framework to represent the process of soil organic matter decomposition in terrestrial ecosystems. It was developed under the idea of hierarchical models that synthesize different approaches to represent the decomposition process. The current version is built under the mathematical formalism of linear dynamical systems to represent, in a very general form, soil organic matter as a state variable with time dependent inputs, outputs, and internal transformations.

A dynamical system, in a broad sense, is a system that evolves in time through the iterated application of an underlying dynamical rule (Jost, 2005). To describe the evolution of a dynamical system over time, it is necessary to represent the actual state of the system and a mathematical rule that dictates the change of state. There are many different ways to represent both the state of the soil system and its transition rules. SOILR provides the basic framework to accommodate different representations of state or system structure and its dynamics. This is accomplished by a library of different numerical functions that can represent many different possibilities of soil organic matter dynamics.

This document presents the main structural characteristics of SOILR and the quantitative tools that can be used to represent different soil biogeochemical processes. The first version of this tool is focused on organic matter decomposition, and other versions of the package will include nutrient dynamics and isotopic composition.

1.1 General information about SOILR and R

The modeling framework we describe in this document is implemented in the R environment for computing (R Development Core Team, 2011). However, numerical ecosystem models are frequently developed in low-level programing languages such as C or Fortran. There are many advantages of using these low-level languages, specially in terms of computational efficiency; however, they are difficult to learn for scientists not formally trained in programing. At a different side of the spectrum of programing languages is the R environment; a high-level language in which ease of use may compromise efficiency. For many applications in ecosystem modeling, the nature and size of the problems are usually not large enough for this to be an important issue. Ease of use however, has been a major constraint for a wide adoption of models in ecosystem science.

Another important issue for model development is accessibility. Models coded in licensed software impose limitations in accessibility and future developments of new tools and models. Open-source software is ideal for guaranteeing that code can be freely distributed, used, and modified by everyone.

SOILR was developed in the R environment for computing to provide simple access of soil organic matter decomposition models in an open-source platform where the code is freely accessible. To see source code and examples of the functions implemented in SOILR the user only needs to type the name of the function in the R command shell. To obtain more detailed documentation, the user can simply type a question mark (?) followed by a function name.

R allows the integration of concepts from functional and object oriented programing and can interface with many other low-level programing languages (Chambers, 2008). R also allows the development of software using concepts of *literate programing* (Knuth, 1984), allowing the production of code and documentation within the same environment. Models and data analyzes in R are therefore easily reproducible.

As an open-source tool, SOILR is also open for contributions by users interested in improving the existing code, make corrections on bugs in the code, add new functions, improve efficiency and functionality, etc.

2 Theoretical framework

2.1 Brief history of SOM modeling

Although early models of soil organic matter decomposition employed geometric series or difference equations (e.g. Nikiforoff, 1936; Jenny et al., 1949), the predominant mathematical formalism since the 1940s is that of ordinary differential equations (Manzoni and Porporato, 2009). Representing decomposition of chemical substances by differential equations was introduced much earlier than that (Van't Hoff, 1884). However, within the ecological disciplines, Olson (1963) presented the first comprehensive treatment of mathematical models of organic matter decomposition, popularizing the model

$$\frac{dX}{\mathrm{d}t} = L - kX, \tag{1}$$

where X is either oven-dry weight, organic carbon, or energy in organic matter; L is the income of organic matter; and k a decay constant.

Equation (1) treats soil organic matter as one single compartment with an overall decomposition rate representative of all substances within the soil matrix. It has been commonly noted that soil organic matter is heterogeneous, and the single exponential model of decomposition fails to account for this heterogeneity (Minderman, 1968; Swift et al., 1979). Earlier, Henin et al. (1959) proposed a model to account for the different rates of decomposition of labile and stable material, also considering the process of humification, i.e. the transfer of material from the labile to the stable pool. This model can be expressed as

$$\begin{aligned}\frac{dX_1}{\mathrm{d}t} &= L - k_1 X_1 \\ \frac{dX_2}{\mathrm{d}t} &= \alpha k_1 X_1 - k_2 X_2,\end{aligned} \tag{2}$$

where X_1 represents the labile pool and X_2 the stable pool. The parameter α represents the humification or transfer rate. A different version of a two-pool model has been widely used for studies of litter decomposition, in which the system of equations takes the form (Minderman, 1968; Means et al., 1985)

$$\begin{aligned}\frac{dX_1}{\mathrm{d}t} &= \gamma L - k_1 X_1 \\ \frac{dX_2}{\mathrm{d}t} &= (1-\gamma) L - k_2 X_2.\end{aligned} \tag{3}$$

In this case, the two pools decompose independently from one another and the amount of litter inputs L is partitioned between the pools according to the parameter γ.

Different variations of these models can be found in the literature, with different number of pools and transfer among compartments.

Two numerical compartment models have become standard in representing organic matter decomposition, these are the RothC (Jenkinson and Rayner, 1977; Jenkinson et al., 1990) and the Century (Parton et al., 1987) models. These two models have been used successfully to represent soil carbon dynamics at different spatial and temporal scales (Paul and Clark, 1996; Paustian et al., 1997). Although these models were developed on the grounds of pragmatism rather than based on strict mathematical formalisms (Bolker et al., 1998), they can be easily translated into systems of differential equations with the general model (Bolker et al., 1998; Paustian et al., 1997)

$$\begin{aligned}\frac{dX_1}{\mathrm{d}t} &= f_1(\theta_1 k_1 X_1, \cdots, \theta_m k_m X_m) \\ &\vdots \\ \frac{dX_m}{\mathrm{d}t} &= f_m(\theta_1 k_1 X_1, \cdots, \theta_m k_m X_m),\end{aligned} \tag{4}$$

where θ is a parameter set modifying the decomposition rate k, and m the total number of compartments representing the system.

In general, the number of compartments in this type of models is less than 10 (Manzoni and Porporato, 2009), and the decomposition rate constant may be a function of temperature, moisture, and/or other edaphic conditions.

We make use of this mathematical abstraction (Eq. 4), to propose a general model of soil organic matter decomposition.

2.2 A general model of soil organic matter decomposition

Models of soil organic matter decomposition are, in their large majority, specific cases of linear dynamical systems (Bolker et al., 1998; Manzoni and Porporato, 2009; Luo and Weng, 2011). Making use of this property, we propose a model that generalizes the majority of all previously proposed compartment models. This general model is given by

$$\frac{d\boldsymbol{C}(t)}{\mathrm{d}t} = \boldsymbol{I}(t) + \mathbf{A}(t)\boldsymbol{C}(t), \tag{5}$$

where $\boldsymbol{C}(t)$ is a $m \times 1$ vector of carbon stores in m pools at a given time t; $\mathbf{A}(t)$ is a $m \times m$ square matrix containing time-dependent decomposition rates for each pool and transfer coefficients between pools; and $\boldsymbol{I}(t)$ is a time-dependent column vector describing the amount of inputs to each pool m.

The matrix $\mathbf{A}(t)$ is particularly important because it defines both the model structure and the extrinsic effects on decomposition and transfer rates. For this reason we rewrite Eq. (5) as

$$\frac{d\boldsymbol{C}(t)}{\mathrm{d}t} = \boldsymbol{I}(t) + \xi(t)\mathbf{A}\boldsymbol{C}(t), \tag{6}$$

where $\xi(t)$ is a time-dependent scalar containing the extrinsic or environmental effects on decomposition rates. Notice that the matrix $\mathbf{A}$ contains now constant coefficients defining model structure.

From this general Eq. (6), it is possible to derive a large variety of structures for compartment models.

2.3 The matrix A and model structure

Organic matter decomposition can be represented with a large variety of model structures and levels of connectivity

among compartments (Swift et al., 1979; Bruun et al., 2008; Manzoni and Porporato, 2009). Different model structures are determined by the matrix **A** in linear dynamical systems (Bolker et al., 1998; Manzoni et al., 2009). For instance, the parallel or pure decay structure (Fig. 1) in compartment models is defined by a diagonal matrix of the form

$$\mathbf{A} = \begin{pmatrix} -k_1 & 0 & \cdots & 0 \\ 0 & -k_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -k_m \end{pmatrix},$$

where the entries in the diagonal represent the decomposition rate k_j for each compartment j. A required condition is that all $k_j \geq 0$.

Compartments connected in series (Fig. 1) can be represented with a matrix of the form

$$\mathbf{A} = \begin{pmatrix} -k_1 & 0 & 0 & \cdots & 0 \\ a_{2,1} & -k_2 & 0 & \cdots & 0 \\ 0 & a_{3,2} & -k_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -k_m \end{pmatrix},$$

where the entries $a_{i,j}$ are the transfer coefficients of material from pool j to pool i. A required condition is that all $a_{i,j} \geq 0$.

Similarly, feedback between adjacent compartments (Fig. 1) is defined by a matrix of the form

$$\mathbf{A} = \begin{pmatrix} -k_1 & a_{1,2} & 0 & 0 & \cdots & 0 \\ a_{2,1} & -k_2 & a_{2,3} & 0 & \cdots & 0 \\ 0 & a_{3,2} & -k_3 & a_{3,4} & \cdots & 0 \\ 0 & 0 & a_{4,3} & -k_4 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & -k_m \end{pmatrix}. \quad (7)$$

More complex model structures are created by replacing zero entries in the matrix **A**, representing transfers between different compartments, i and j.

An important characteristic of the entries $a_{i,j}$ is that they are proportional to the decomposition rate, i.e. $a_{i,j} = \alpha_{i,j} k_i$, where $\alpha_{i,j}$ represents the proportion of the decomposition rate that is transferred to pool i from pool j. Furthermore, $0 \leq \alpha_{i,j} \leq 1$, and the column sum $\sum_i \alpha_{i,j} \leq 1$, with

$$r_j = 1 - \sum_i \alpha_{i,j} \quad (8)$$

representing the proportion of the decomposed material that gets released from the system from pool j.

2.4 The environmental term $\xi(t)$

The majority of organic matter decomposition models include functions $f(x)$ that modify decomposition rates according to a set of time-varying environmental conditions

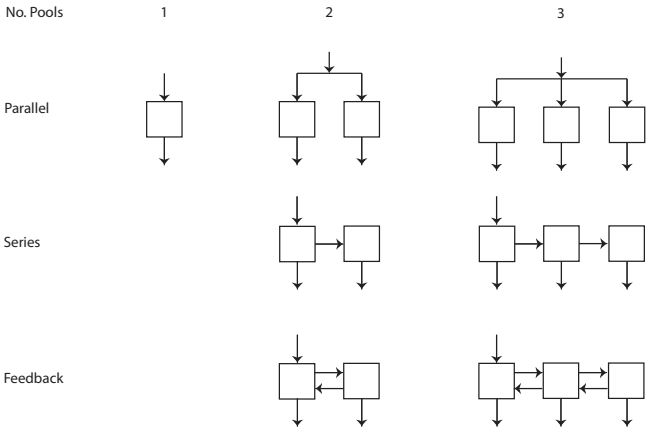

Fig. 1. Basic model structures implemented in SOILR. Squares represent the compartments, and arrows represent inputs and outputs to and from the compartments. These model structures are special cases of the matrix **A**.

$\{x_1(t) \ldots x_n(t)\}$, such as temperature, moisture, evapotranspiration, etc. (Burke et al., 2003; Adair et al., 2008). In our model (Eq. 6), the representation of these environmental effects is done with the term $\xi(t)$, which is the result of the evaluation of the function or set of functions $f(x_i(t))$, yielding a scalar value that can be directly multiplied to the matrix **A**. In this case,

$$\xi(t) = f(x_1(t), \ldots, x_n(t)). \quad (9)$$

The values of $f(x_i(t))$ are determined by different functions that depend on temperature, precipitation, and other environmental variables. Time-dependence is, therefore, introduced with time series of these environmental variables as input in the model. SOILR contains a library of functions that calculate environmental effects on decomposition rates based on functions reported for different models (Table 1).

More complex functions are also introduced in SOILR but are not included in Table 1 due to space limitations. The documentation and help files of SOILR contain a more detailed description of all functions.

2.5 Initial conditions

The linear dynamical system represented by Eq. (6), has many different solutions, but we are only interested in the solution that satisfies

$$\boldsymbol{C}(t = 0) = \boldsymbol{C_0}, \quad (10)$$

where $\boldsymbol{C_0}$ is a $m \times 1$ vector with the value of carbon content in the different compartments i. $\boldsymbol{C_0}$ must be specified in SOILR to run any possible model structure.

2.6 The vector of inputs

Inputs to the system from above and belowground components are represented by the vector $\boldsymbol{I}(t)$. This vector can also

Table 1. Functions implemented in SOILR to represent the effects of temperature T, and moisture W on decomposition rates.

$f(x)$	Terms	Function name	Source
$f(T) =$			
$Q_{10}^{(T-10)/10}$	T: mean temperature	`fT.Q10`	
$\frac{47.9}{1+\exp(\frac{106}{T+18.3})}$	T: monthly temperature (°C)	`fT.RothC`	Jenkinson et al. (1990)
$\left(\frac{T_{\max}-T}{T_{\max}-T_{\text{opt}}}\right)^{0.2}\exp\left(\frac{0.2}{2.63}\left(1-\left(\frac{T_{\max}-T}{T_{\max}-T_{\text{opt}}}\right)^{2.63}\right)\right)$	T, $T_{\max}$ T_{opt}: monthly average, maximum, and optimal temperature	`fT.Century1`	Burke et al. (2003)
$3.439\exp\left(\frac{0.2}{2.63}\left(1-\left(\frac{T_{\max}-T}{T_{\max}-T_{\text{opt}}}\right)^{2.63}\right)\left(\frac{T_{\max}-T}{T_{\max}-T_{\text{opt}}}\right)^{0.2}\right)$	T, $T_{\max}$ T_{opt}: monthly average, maximum, and optimal temperature	`fT.Century2`	Adair et al. (2008)
$0.8\exp(0.095T_s)$	T_s: Soil temperature	`fT.Daycent1`	Kelly et al. (2000)
$0.56+(1.46\arctan(\pi 0.0309(T_s-15.7)))/\pi$	T_s: Soil temperature	`fT.Daycent2`	Parton et al. (2001); Grosso et al. (2005)
$0.198+0.036T$	T: monthly temperature	`fT.linear`	Adair et al. (2008)
$\exp\left(308.56\left(\frac{1}{56.02}-\frac{1}{(T+273)-227.13}\right)\right)$	T: monthly temperature	`fT.LandT`	Lloyd and Taylor (1994)
$\exp(-3.764+0.204T(1-0.5T/36.9))$	T: mean temperature	`fT.KB`	Kirschbaum (1995)
$\exp((\ln(Q_{10})/10)(T-20))$	T: mean temperature. Q_{10}: temperature coefficient	`fT.Demeter`	Foley (2011)
$\exp(-(T/(T_{\text{opt}}+T_{\text{lag}}))^{T_{\text{shape}}})Q_{10}^{(T-10)/10}$	T, $T_{\max}$ T_{opt}: monthly average, maximum, and optimal temperature	`fT.Standcarb`	Harmon and Domingo (2001)
$f(W) =$			
$\frac{1}{1+30\exp(-8.5W)}$	$W = P/\text{PET}$, P: monthly precipitation, PET: monthly potential evapotranspiration	`fW.Century`	Parton et al. (2001); Adair et al. (2008)
$\left(\frac{W-b}{a-b}\right)^{d((b-a)/(a-c))}\left(\frac{W-c}{a-c}\right)^{d}$	W: water filled pore space. a, b, c, d: empirical coefficients	`fW.Daycent1`	Kelly et al. (2000)
$5(0.287)+(\arctan(\pi 0.009(RWC-17.47))/\pi)$	W: volumetric water content	`fW.Daycent2`	Grosso et al. (2005)
$0.25+0.75(M/M_{\text{sat}})$	M: soil moisture. M_{sat}: saturated soil mositure	`fW.Demeter`	Foley (2011)
$(1-\exp(-(3/M_{\min})(M+a)))^{b}\exp(-(M/(M_{\max}+c))^{d})$	M, $M_{\min}$, $M_{\max}$: average, minimum and maximum moisture content in litter pool. a, b, c, d: empirical coefficients	`fW.Standcarb`	Harmon and Domingo (2001)

be expressed as

$$\boldsymbol{I}(t) = I(t)\begin{pmatrix}\gamma_1\\ \vdots\\ \gamma_i\\ \vdots\\ \gamma_m\end{pmatrix} \tag{11}$$

where $I(t)$ is a time-dependent scalar representing the total amount of inputs and the coefficients γ_i represent the partitioning among the different pools. In this representation $0 \le \gamma_i \le 1$, and $\sum_{i=1}^{m}\gamma_i = 1$.

2.7 Carbon release

A variable of interest in modeling soil organic matter decomposition is the amount of carbon leaving the system over time either in the form of CO_2 gas or as dissolved organic carbon. We represent this flux with the general term $\boldsymbol{r}$, which is given by

$$\boldsymbol{r} = \mathbf{R}\boldsymbol{C}(t), \tag{12}$$

where $\boldsymbol{r}$ is a $m\times 1$ vector containing the instantaneous release of carbon for all pools, and $\mathbf{R}$ is a $m\times m$ diagonal matrix with the release coefficients r_j in its diagonal calculated from (8).

2.8 Analytical solution

Analytical solutions to Eq. (5) are implemented in SOILR only with the purpose of testing the performance of the numerical methods. However, we can only test cases under certain simplifications of the general model of Eq. (5). In particular, for a homogeneous system with constant coefficients, which is analogous to the decomposition of a single cohort

of organic matter (Ågren and Bosatta, 1998); Eq. (5) simplifies to

$$\frac{d\boldsymbol{C}(t)}{\mathrm{d}t} = \mathbf{A}\boldsymbol{C}(t). \tag{13}$$

With initial conditions as in Eq. (10), the analytical solution to this problem is given by

$$\boldsymbol{C}(t) = e^{\mathbf{A}(t-t_0)}\boldsymbol{C_0}. \tag{14}$$

If $\boldsymbol{I}(t)$ is not identically zero, then the solution of the linear system

$$\frac{d\boldsymbol{C}(t)}{\mathrm{d}t} = \boldsymbol{I}(t) + \mathbf{A}\boldsymbol{C}(t),$$

with initial conditions as in Eq. (10), is given by

$$\boldsymbol{C}(t) = e^{\mathbf{A}(t-t_0)}\boldsymbol{C_0} + \int_{t_0}^{t} e^{\mathbf{A}(t-\tau)}\boldsymbol{I}(\tau)d\tau. \tag{15}$$

3 Numerical implementation

The solution to the dynamical system described by Eq. (6) is discretized over time, with h denoting the time step and n the number of steps. The time step h may or may not be constant. Initial conditions are given at time $t_0 = 0$. The solution to the system is then given by

$$\boldsymbol{C}_{n+1} = \boldsymbol{C}_n + D_r[f'(\boldsymbol{C}_n), h], \tag{16}$$

where $D_r[f'(\boldsymbol{C}_n), h]$ is an r order finite difference approximation to the system of ODEs of Eq. (6) for each time step h (LeVeque, 2007); in other words, an ODE solver.

The choice for the ODE solver is flexible in SOILR. Currently, we provide the option to use a simple Euler forward method or an interface to the `deSolve` package of Soetaert et al. (2010). The function `deSolve.lsoda.wrapper` in SOILR, is a wrapper to the function `lsoda` in package `deSolve`.

3.1 The `Model` class

The general numerical model described by Eq. (6) is represented in SOILR by an object oriented design centering around the class `Model`. Nearly all computations possible in SOILR are facilitated by `Model` itself or its building blocks. The class defines the information all `Model` objects contain and how those objects behave when (generic) functions are applied to them (Chambers, 2008). Any model constructed with SOILR provides then a stable and consistent interface that encapsulates details about its implementation. This interface provide a number of advantages to the user:

1. A level of abstraction close to the mathematical description of the model being implemented, focusing on the scientific content rather than on technical details.
2. Stability of code generated by the user based on SOILR output. We believe that the more essential and less redundant the interface, the less likely it is to change in future revisions.
3. Safeguards against unreasonable (and therefore probably unintentional) input, from simple dimensional checks to mathematical consistency of all arguments of the general model described in Sects. 2.2 through 2.7.
4. Smaller programs that are easier to read.

Developers also benefit from this implementation design by:

1. Freedom to change the implementation behind the scenes without breaking user code.
2. Easier maintenance, testing, bug tracking, and exception handling by reduced duplication

Objects of class `Model` are initialized in SOILR by various functions, which are listed with the command `?Model`. Most of them provide shortcuts for the construction of standard models such as those in Fig. 1. These functions usually do not require the user to consider the real building blocks of a `Model` object, but rather infer the needed information from objects more common in `R` such as data frames. However, all these high-level functions call `GeneralModel`, which is the most general constructor. The function `GeneralModel` initializes the `Model` object after applying various validity checks, but does not run the model itself, which instead is done by specific functions (methods) applied to objects of class `Model`.

The function `GeneralModel` takes five arguments representing a specific case of Eq. (6).

1. A vector `t` which contains the time values where the solution to the ODE system is sought. It can be of any length but must be of class `"numeric"`.
2. An object of class `"TimeMap"` which implements the matrix **A** of Eq. (5) as a function of time (including its domain) thus allowing the same level of abstraction as the most general ODE solvers (Soetaert et al., 2010). The class `TimeMap` is native to SOILR and simply extends a common `R` function to a mathematical valid function definition that includes the time domain where the function is defined. Thus it contains the necessary information to prevent extrapolations beyond the range of input data. Details about `TimeMap` are discussed in Appendix A.
3. A vector of class `"numeric"` containing the initial values $\boldsymbol{C_0}$ of the ODE system.
4. Again an object of class `"TimeMap"` containing the inputs $\boldsymbol{I}(t)$ to the system as a vector valued function of time. The length of this vector must be equal to the dimension of the matrix **A**, and is checked as well as the time domain.

5. A string choosing the solver to be used.

Once a new object of class `Model` is intialized from a call to `GeneralModel`, or one of its various wrappers, the new object can be queried by several generic functions. For example, to obtain the amount of carbon over time solving the system of ODEs, any object of class `Model` (and therefore all its subclasses) can be used as argument of the function `getC`. The call to this function returns a $n \times m$ matrix with the amount of carbon for each pool m at each time step n. Similarly, to obtain the amount of carbon release over time, the function `getReleaseFlux` can be called with any object of class `Model` as argument. The result will be a $n \times m$ matrix with the amount of released carbon for each pool m at each time step n. It is also possible to use the typical operators `[ ]` and `$` for our `Model` class given access to methods even easier. For example, `getC(mod)` is equivalent to `mod$C` and something like:

```
df=as.data.frame(cbind(getTime(mod),
    getC(mod)))
```

can also be expressed as

```
df=mod[c("time", "C")]
```

This avoids the necessity of the somewhat dangerous use of the `@` operator in user code, which we strongly discourage, since it attaches the code to implementational details that may change in the future.

The implementation of specific models as subclasses of `Model` with methods for the generic functions will allow the integration of new functionality without major modifications to our current implementation. For example, once nutrient cycling and isotope dynamics are incorporated into SOILR, new methods will be developed independently without major modifications to the current implementation of carbon stocks and release.

The vignette `GeneralModel` provided with SOILR presents some further insight in the implementation of specific model structures with the help of the general tool-set provided by the `Model` class. The examples therein may also serve as templates to implement new model structures as desired by the user. Examples on how to use these model structures as a function are presented in Sect. 4.

3.2 Version control system, unit testing, and automatic documentation

The development of SOILR is aided by a significant amount of existing open- source software. To solve the ordinary differential equations produced by our framework, we rely on the well tested and documented `deSolve` package developed by Soetaert et al. (2010). We also use the open-source symbolic python library SymPy (SymPy Development Team, 2008) to compute analytical solutions for the models for which this is possible. The analytical solutions obtained from SymPy are used to automatically create unit tests for SOILR. To constantly run these tests, we use another open-source software, the `Runit` package (Burger et al., 2009). The tests are distributed with the release version of SOILR and thus add to the transparency of its development. In addition, we use `Sweave` (Leisch, 2002, 2003) and the `inlinedocs` package (Hocking et al., 2012) to produce documentation and encourage a literate programing style (Knuth, 1984). As a version control system, we use Mercurial (O'Sullivan, 2009) and the Trac (Edgewall Software, 2011) online project management tool, which includes ticket system, wiki, and online access to our source code.

3.3 Documentation

There are different types of documentation for SOILR. The first source of information is this document, which introduces the science and some general technical details. A second source of information is the documentation to each function provided within the package itself. To view this documentation, the user only needs to open R and type `help.start()`. This will open a help window on a web-browser. There the user only needs to go to Packages/SoilR to view a list of all the functions implemented. Clicking on each function will show details about the arguments of each function and examples on how to use them. For specific functions, the user can also just type the name of the function preceded by the question mark on the R command shell. For example, typing `?TwopParallelModel` in the R command shell will open a help window with the description of the function. To view the source code of the function, the user only needs to type the name of the function (without the question mark) on the R command shell.

A third source of information are the so called Package Vignettes. These are short documents illustrating the use of the package for specific purposes. Currently, we provide one vignette with version 1.0 of SOILR. This vignette illustrates the implementation of any model structure within SOILR. For future versions, we will provide vigenettes about fitting specific model structures to data and how to use SOILR for modeling radiocarbon.

3.4 Installing and loading SOILR

SOILR can be obtained from the Comprehensive R Archive Network (CRAN), the official repository for R packages with mirrors in places all over the world. Packages stored in CRAN can be downloaded directly from an R session. It can also be obtained from R-Forge, a repository for package developers. To install SOILR from CRAN, the user simply needs to type in the R command shell `install.packages("SoilR")`. To install from R-Forge, the statement is `install.packages("SoilR", repos="http://R-Forge.R-project.org")`. After installing the package, simply type

`library(SoilR)` and the package is loaded into your R session.

4 Examples

In this section we present examples on how to run some of the functions implemented in SOILR based on the theoretical framework presented previously. Additional details about the implementations of each function and instructions on how to implement new model structures are presented in the vignette 'Implementing Compartment Models in SOILR: the `GeneralModel` Function' provided with the package. To view this vignette, simply type `vignette("GeneralModel", package="SoilR")` in the R command shell.

4.1 Implementation of a two pool model with connection in series: the ICBM model

One of the first models ever proposed to represent soil organic matter dynamics was a two-pool model with connection in series (cf. Eq. 2, Henin et al., 1959). More than 50 yr later, Andren and Katterer (1997) proposed the ICBM model, which is practically the same model proposed earlier by Henin et al. (1959), but including a term for temperature and moisture dependence of decomposition rates. The set of differential equations of the ICBM model are given by

$$\frac{dC_1}{\mathrm{d}t} = I - k_1 \xi C_1$$
$$\frac{dC_2}{\mathrm{d}t} = \alpha k_1 \xi C_1 - k_2 \xi C_2, \quad (17)$$

where α is a humification or transfer coefficient and ξ a parameter representing external effects on decomposition rates. In the ICBM model, C_1 represents a "young" pool and C_2 an "old" pool. This set of equations can be rewritten using our model formulation, which gives

$$\frac{d\boldsymbol{C}}{\mathrm{d}t} = I \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \xi \begin{pmatrix} -k_1 & 0 \\ \alpha k_1 & -k_2 \end{pmatrix} \begin{pmatrix} C_1 \\ C_2 \end{pmatrix}, \quad (18)$$

with initial conditions $(C_{1,0}, C_{2,0})^{\mathbf{T}}$, and where

$$\mathbf{A} = \begin{pmatrix} -k_1 & 0 \\ a_{2,1} & -k_2 \end{pmatrix},$$

with $a_{2,1} = \alpha k_1$, $\gamma = 1$, and $\xi(t) = \xi$.

The function `ICBMModel` in SOILR implements this model structure requiring as its arguments: (1) a vector of any length with the points in time when we are interested in finding a solution, (2) a column vector of decomposition rates $(k_1, k_2)^{\mathbf{T}}$, (3) the value of α, (4) the value of ξ, (5) a column vector with the initial amount of carbon at the beginning of simulation $(C_{1,0}, C_{2,0})^{\mathbf{T}}$, and (6) the mean annual carbon input to the soil I. Andren and Katterer (1997) provided values for these arguments from a 35-yr field experiment manipulating carbon and nitrogen inputs to an agricultural soil in Sweden. For the case of a treatment in which the soil was left as bare fallow without N or C inputs, the ICBM model can be parameterized as

$$\frac{d\boldsymbol{C}}{\mathrm{d}t} = 0 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + 1.32 \begin{pmatrix} -0.8 & 0 \\ 0.13 \times 0.8 & -0.00605 \end{pmatrix} \begin{pmatrix} C_1 \\ C_2 \end{pmatrix}, \quad (19)$$

with initial conditions $(0.3, 3.96)^{\mathbf{T}}$.

Assuming SOILR is already installed, it is only necessary to write the following lines of code to run the ICBM model

```
library(SoilR)
times=seq(0,20,by=0.1)
Bare=ICBMModel(t=times, ks=c(k1=0.8,
         k2=0.00605), h=0.13, r=1.32,
          c0=c(C10=0.3,C20=3.96), In=0)
```

The call to `ICBMModel` simply initialize, the model and checks for consistency on its arguments. In this example, the decomposition rates are given in units of year^{-1} and the initial amounts of carbon in units of kg C m^{-2}.

To obtain the amount of carbon over time it is necessary to invoke the function `getC` storing the output into an object. For example, to store the amount of carbon from the object `Bare` the user can type

```
CtBare=getC(Bare).
```

This new object, `CtBare`, is a matrix with 2 columns (2 pools) and 201 rows (201 points in time, from 0 to 20 in increments of 0.1). To obtain the total amount of carbon, i.e. the sum of the pools, the R function `rowSums` can be used. For example, plotting the total amount of carbon over time as well as the carbon on each pool only requires these lines of code

```
plot(times, rowSums(CtBare),type="l",
   ylim=c(0,5), ylab="Topsoil carbon
   mass (kg m-2)",xlab="Time (years)",
   lwd=2)
lines(times, CtBare[,1],lty=2)
lines(times,CtBare[,2],lty=3,col=2,lwd=2)
legend("topright",c("Total carbon",
  "Carbon in pool 1", "Carbon in pool 2"),
  lty=c(1,2,3),col=c(1,1,2),lwd=c(2,1,2),
  bty="n")
```

If the total amount of carbon is needed for further calculations, the output of `rowSums()` can be stored in an object with any name.

We implemented the different N and C treatments reported in Andren and Katterer (1997) from the set of parameters reported by those authors. The code necessary to reproduce Fig. 2 in Andren and Katterer (1997) is provided as an example with the function `ICBMModel` and can be accessed by

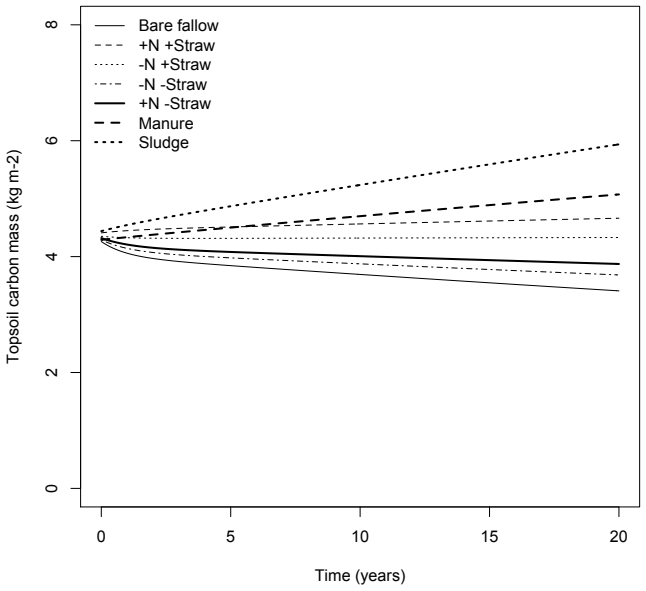

Fig. 2. Model predictions with the version of the ICBM model implemented in SOILR. This graph reproduces Fig. 2 in Andren and Katterer (1997). This figure can be reproduced typing `example(ICBMModel)` or `attr(ICBMModel,"ex")` in SOILR.

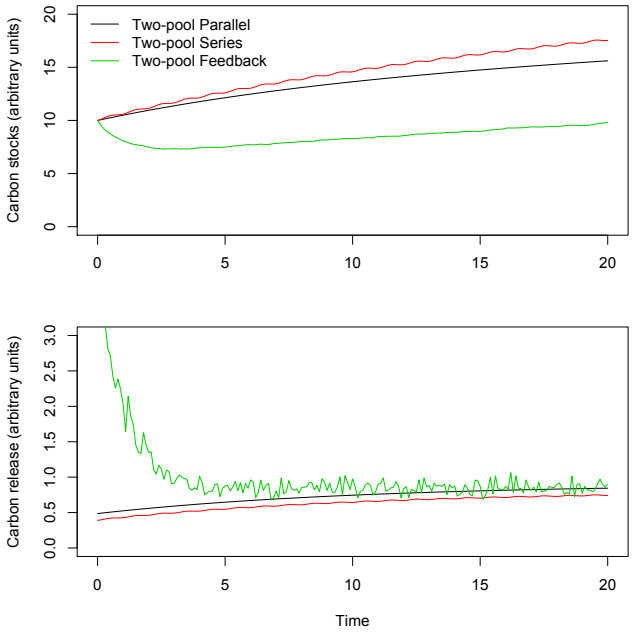

Fig. 3. Examples of three different representations of a two-pool model with different model structures and environmental effects on decomposition rates. The upper panel shows carbon stocks and the lower panel carbon release. Additional details about the implementation are given in the text.

typing `example(ICBMModel)`, or from the html help in R. This code produces Fig. 2, which is identical to Fig. 2 in Andren and Katterer (1997).

4.2 Alternative two-pool models

The ICBM model described in the previous section, although useful, does not offer too much flexibility in terms of the input arguments. For example, the litter inputs to the system could vary over time as well as the temperature and moisture effects on decomposition rates. In addition, there are other possibilities to implement a two pool model depending on the type of connection between pools (Fig. 1).

The parallel pool model structure can be implemented with the function `TwopParallelModel`, while a more general version of a series model structure can be implemented with the function `TwopSeriesModel`. Similarly, the feedback model structure can be implemented with the function `TwopFeedbackModel`. The inputs of litter and the modification of decomposition rates by external factors can be either constant or a function of time. Furthermore, the functions presented in Table 1, and their combinations, can be used as arguments in the different model structures providing a large variety of options to model decomposition with just two pools. In fact, the same flexibility can be obtained with any number of pools with the application of the more general function `GeneralModel`.

As an example, we show the differences obtained by running three different versions of a two-pool model with the same amount of carbon at the beginning of the simulation, similar rates of litter inputs, and equal decomposition rates (Fig. 3). As an illustration, we also ran the simulations with different options for the time dependence of the litter inputs and the decomposition rates. In the first simulation, we ran a model with a structure of parallel compartments. In this simulation, the litter inputs and decomposition rates were constant, but the decomposition rates were modified by average values of temperature and moisture according to the functions proposed in the Daycent model (Table 1). For the second simulation, we ran a two-pool model with connection in series introducing temporal variability in the amount of inputs using a sine function that artificially represents an annual cycle. In the third simulation, we ran a two-pool model with connection in series among compartments. The amount of litter inputs over time were calculated using random numbers over time. In this simulation, we also produced random numbers of temperature and moisture and applied the functions to modify decomposition rates according to the Century and the Demeter models (Fig. 3). The code to reproduce Fig. 3 is provided in the example of the function `TwopFeedbackModel`.

These simulations, without being necessarily realistic, simply show that small differences in model structure can produce very different predictions, even when the main parameters of the model remain unchanged. The simulations also serve to illustrate different possibilities in the use of the

basic functions of SOILR to represent the process of organic matter decomposition over time.

To implement more sophisticated models with a higher degree of complexity, it is possible to specify a larger amount of pools with complex functions representing the dependence of litter inputs, decomposition rates, and transfer between pools with other external variables, such as temperature, moisture, soil texture, nutrient status, among many other.

4.3 Implementation of the RothC model

RothC is a popular and widely used model for predicting organic matter dynamics over time. Although earlier versions of the model included five active pools and one inert pool (Jenkinson and Rayner, 1977), more recent versions only include four active pools plus the inert pool (Jenkinson et al., 1990). RothC is implemented within SOILR and we provide details here about this implementation to illustrate the use and potential implementation of any other model. RothC can be described by the following set of differential equations

$$\begin{aligned}
\frac{dC_1}{\mathrm{d}t} &= \gamma I - k_1 C_1 \\
\frac{dC_2}{\mathrm{d}t} &= (1-\gamma) I - k_2 C_2 \\
\frac{dC_3}{\mathrm{d}t} &= \alpha_{3,1} k_1 C_1 + \alpha_{3,2} k_2 C_2 - k_3 C_3 + \alpha_{3,3} k_3 C_3 + \alpha_{3,4} k_4 C_4 \\
\frac{dC_4}{\mathrm{d}t} &= \alpha_{4,1} k_1 C_1 + \alpha_{4,2} k_2 C_2 + \alpha_{4,3} k_3 C_3 - k_4 C_4 + \alpha_{4,4} k_4 C_4 \\
\frac{dC_5}{\mathrm{d}t} &= 0
\end{aligned} \tag{20}$$

where C_1 represents the decomposable plant material (DPM) pool, C_2 the resistant plant material (RPM) pool, C_3 the microbial biomass (BIO) pool, C_4 the humified organic matter (HUM) pool, and C_5 the inert organic matter pool (IOM). This set of equations can be rewritten in the form

$$\frac{d\boldsymbol{C}}{\mathrm{d}t} = I \begin{pmatrix} \gamma \\ 1-\gamma \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} -k_1 & 0 & 0 & 0 & 0 \\ 0 & -k_2 & 0 & 0 & 0 \\ \alpha_{3,1}k_1 & \alpha_{3,2}k_2 & -k_3(1-\alpha_{3,3}) & \alpha_{3,4}k_4 & 0 \\ \alpha_{4,1}k_1 & \alpha_{4,2}k_2 & \alpha_{4,3}k_3 & -k_4(1-\alpha_{4,4}) & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \end{pmatrix}$$

or as

$$\frac{d\boldsymbol{C}}{\mathrm{d}t} = I \begin{pmatrix} \gamma \\ 1-\gamma \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} -k_1 & 0 & 0 & 0 & 0 \\ 0 & -k_2 & 0 & 0 & 0 \\ a_{3,1} & a_{3,2} & -k_3 + a_{3,3} & a_{3,4} & 0 \\ a_{4,1} & a_{4,2} & a_{4,3} & -k_4 + a_{4,4} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \end{pmatrix} \tag{21}$$

The values of the decomposition rates are constant and given by: $k_1 = 10$, $k_2 = 0.3$, $k_3 = 0.66$, and $k_4 = 0.02$ (Jenkinson et al., 1990; Coleman and Jenkinson, 1999). The value of the transfer coefficients is determined by a function of soil texture. For the microbial biomass pool, transfer coefficients are calculated as

$$a_{3,j} = k_j \frac{0.46}{x+1} \tag{22}$$

where x is a value that determines the proportion of decomposed material that is respired as CO_2 and is given by

$$x = 1.67(1.85 + 1.60\exp(-0.0786\, p\mathrm{Clay})) \tag{23}$$

where pClay is percent clay in mineral soil (Jenkinson et al., 1990). Similarly, the transfer coefficients for the humified pool are given by

$$a_{4,j} = k_j \frac{0.54}{x+1}. \tag{24}$$

The partitioning of incoming plant material is determined by the ratio DPM/RPM, which in RothC is set as 1.44. Therefore, $\gamma = 0.59$. Now, the basic structure of the RothC model can be written as

$$\frac{d\boldsymbol{C}}{\mathrm{d}t} = I \begin{pmatrix} 0.59 \\ 0.41 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} -10 & 0 & 0 & 0 & 0 \\ 0 & -0.3 & 0 & 0 & 0 \\ 1.02 & 0.03 & -0.59 & 0.01 & 0 \\ 1.19 & 0.04 & 0.08 & -0.02 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \end{pmatrix}. \tag{25}$$

The annual amount of inputs is set in the RothC model as $I = 1.7\,\mathrm{Mg\,C\,ha^{-1}\,yr^{-1}}$.

With this parameterization, it is possible to run the RothC model without varying environmental effects on decomposition rates and observe how the system approaches steady-state for the different pools (Fig. 4). Parameter values, initial conditions, and litter inputs can be changed easily within SOILR to compare different predictions from this model.

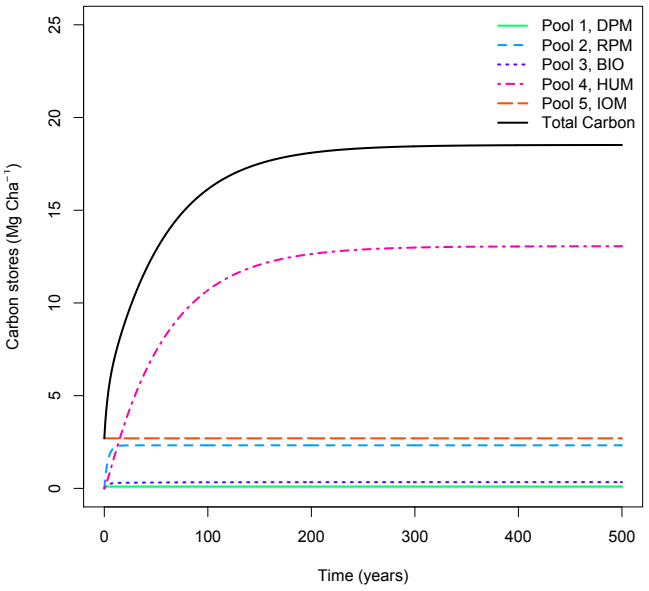

Fig. 4. Carbon accumulation for the different pools included in the RothC model. DPM: the decomposable plant material pool, RPM: resistant plant material, BIO: microbial biomass pool, HUM: humified organic matterpool, and IOM: inert organic matter pool.

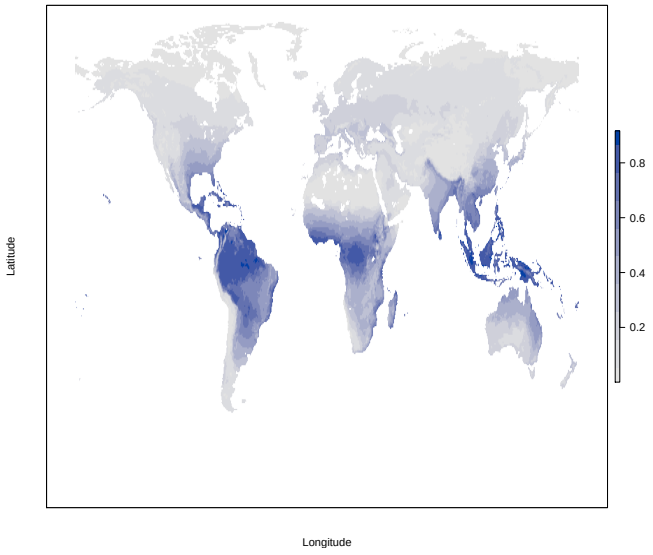

Fig. 5. Climate decomposition index (CDI) calculated as the product of a function of temperature (`fT.Century1`) and a function of precipitation and potential evapotranspiration (`fW.Century`) using monthly data from the WATCH dataset (Weedon et al., 2011).

4.4 Applications for large-scale modeling

R can handle a large variety of data structures, which offers interesting possibilities for the application of the functions implemented in SOILR. One important type of data structure is spatial data with global coverage. There is a large number of R packages to import and manipulate spatial data, which facilitates the technical aspects for running SOILR functions at the global scale.

As an example, we show here a simple calculation of climate decomposition indexes (CDIs) (Adair et al., 2008) using a global dataset of temperature, precipitation, and potential evapotranspiration available at 0.5 degree resolution on NetCDF files from the WATCH dataset (Weedon et al., 2011). The input dataset contained monthly average temperature, precipitation and evapotranspiration for the period 1958 to 2001. We calculated the CDIs as

$$\mathrm{CDI} = \xi(t) = f(T)f(W) \tag{26}$$

with $f(T)$ implemented by the function `fT.Century1`, and $f(W)$ by the function `fW.Century` as described in Table 1. Results can be easily plotted on a map (Fig. 5) and exported again to NetCDF files. Different combinations of the functions described in Table 1 can be used to calculate CDIs (Adair et al., 2008) and evaluate different hypotheses about the response of decomposition rates to moisture, temperature, and other variables.

5 Discussion

Many models of soil organic matter decomposition have been proposed previously, and there even exist some open-source tools to implement some of these models (e.g. Easter et al., 2007). We have developed a tool for implementing and running a large variety of these models with the idea of facilitating comparison among multiple models in an easy to use interface. In this section, we discuss some of the advantages and disadvantages of our approach.

5.1 Parameter space and structural domain

Models of organic matter decomposition are basically hypothetical abstractions about the structure and dynamics of soil organic matter. The multitude of models previously developed suggests that there exists a large number of hypotheses about the structure and functioning of soil organic matter, but there is basically little consensus on whether one model structure (hypothesis) would have more support on observations than other model structures (Manzoni and Porporato, 2009).

The majority of modeling studies have focused on finding the set of parameter values of a particular model that best fit some observed data. This approach is useful and has provided much insight on understanding the rates of soil organic matter decomposition. However, from the perspective of assessing different hypotheses about the structure and dynamics of soil organic matter, parameter estimation can only give a narrow view of the more complex spectrum resulting from

the combination of structure domain and the parameter space of models.

SOILR provides the possibility of assessing both model structure and parameter values broadening the spectrum of ideas that can be assessed within one single analytical framework.

We believe this approach complements well, and could be even more powerful than previous approaches to assess performance of model structure with inter-comparisons among different modeling groups (e.g. Melillo et al., 1995; Wu and McGechan, 1998; Cramer et al., 2001; Randerson et al., 2009). Multi-model inter-comparison projects do not necessarily cover the whole domain of model structures, and may be subject to important issues, such as independence of code, bias of the whole model ensemble, inappropriate metrics to define model performance, etc (Knutti et al., 2009; Knutti, 2010). SOILR can partially help to overcome some of these problems for assessing the performance of model structure through the option of using alternative functions to represent the same process. Philosophically, this approach is also similar to testing multiple working hypotheses.

5.2 Model hierarchies and functional programing

The gap between simulations and understanding described by Held (2005) is currently exacerbated by the continuous increase in detail and complexity of simulation models. Held (2005) suggests that a way forward to close the gap between simulations and understanding is by the development of model hierarchies in which large-scale complex models are particular cases of general models that are more amenable for understanding of system structure and behavior.

SOILR can also be viewed as a system for hierarchical modeling of soil processes. Consider for example, the environmental or external effects on decomposition rates, which here are denoted by the term $\xi(t)$. In its more simple and general case, the external effects can be simply a constant ($\xi(t) = c$) that allows the understanding of model behavior without changes in environmental conditions. Simulations can then be run with a changing environment, for example with variable soil moisture ($\xi(t) = f(W(t))$). Soil moisture could depend on other variables, such as precipitation and potential evapotranspiration ($W(t) = f(P(t), \mathrm{PET}(t))$), which in turn can be dependent on other functions. In this form, a hierarchy of models is build with the dependence of different functions on other functions.

In terms of programing, this concept of model hierarchies can be easily implemented in a functional programing style. A function that performs certain tasks can have as its arguments other functions that perform other tasks. These functions can be independent among each other so many different functions that perform the same task can be available within the same modeling environment. This is one of our goals with SOILR, to provide a modeling environment that serves as a repository of different functions that can perform the same task so their performance can be easily compared. It also allows building soil organic matter decomposition models in a hierarchical framework because functions can have a large number of dependences on other functions creating a bridge between simple general models and detailed modeling constructions under the same basic principles.

5.3 Scope and limitations

Our approach to soil organic matter decomposition modeling with SOILR may have some limitations. One potential limitation is the use of a high-level programing language that can have issues in terms of computational efficiency. This could be an important problem if specific models structures with a large number of computations per time step are applied to a large number of points such as a large spatial grid. Other programing languages such as C or Fortran may be more suitable for these tasks, although a good programing style can avoid many issues of computational efficiency in R. It is important to keep in mind that the scope of SOILR is more for the exploration of different model structures and hypotheses about soil processes rather than for large computational tasks; however, some parallelization tools within R could be used for this purpose. We recommend that once a specific model structure is identified as useful for a large computation, the entire model object is translated to other language or optimized for running in R. We are exploring these possibilities to include in future releases of SOILR.

Another potential limitation is the incompatibility of our conceptual framework to represent the decomposition process as a non-linear dynamical system. Non-linear dynamics are important for representing microbial processes such as priming, and are very relevant for simulations at short time scales (Wutzler and Reichstein, 2008; Manzoni and Porporato, 2007, 2009). Non-linear models however, can be easily implemented in R with the `deSolve` package. We are considering two possibilities to include non-linear dynamical systems within our theoretical framework. One possibility is to provide a framework to specify and linearize non-linear systems and solve the linear equations as presented here. Another possibility is to include a set of non-linear models solving them directly with `deSolve`. We aim for further releases to include this additional functionality.

6 Conclusions

We have developed a modeling environment for representing the process of soil organic matter decomposition. This tool, SOILR, is an open- source package for the implementation and testing of different representations of the process of soil organic matter decomposition. The main characteristic of SOILR is its hierarchical structure in which we describe a general model that can accommodate any possible model structure of a multi-pool model of decomposition.

More detailed models can be implemented to simulate specific controls on the decomposition process. This allows for testing of multiple working hypotheses about the structure and functioning of soils and their behavior over time. SOILR not only allows for exploring dynamics on the parameter space of a model but also on the structural domain.

This first version of SOILR allows simulations of organic matter decomposition and future versions will include representations of other biogeochemical elements such as nitrogen and phosphorus as well as their isotopic composition. A module for parameter fitting will also be included in future releases.

Appendix A

Implementation details and design decisions regarding SOILR

A1 The choice of the object-oriented programing (OOP) system

While in object oriented languages the implementation of object oriented design principles is usually hardwired, in R it is a choice between at least three different approaches, thus forcing the user to explain why a particular system was used. In our case the choice of S4 is a compromise. On one hand there is very limited support in S4 for real encapsulation, expressed most notably by the possibility to read and even write the values of slots from outside the class by the `slot` function or the @ operator, respectively. This default behavior is actually not even trivially changed. On the other hand there are some important advantages:

1. Compatibility with the functional paradigm of R and its associated benefits such as lazy evaluation, concurrency and in the future maybe even automatic parallelization, which depends on the absence of side effects (unavoidable with a reference based system like R5).

2. The integration with R's standard library, which increases

   a. the probability of future maintenance, and

   b. acceptance by other R users.

3. The relative seamless integration to common R workflows without in depth knowledge of object oriented programing.

Considering that if R is capable to implement an OOP system by means of on board tools it will probably be flexible enough to change the default behavior of this OOP system with on board tools as well. SOILR is a rather small package and we think S4 is and appropriate choice for our needs.

A2 Comparison between SIMECOL and SOILR

SIMECOL (Petzoldt and Rinke, 2007) an R package for the implementation of ecological models, is an alternative to our own implementation. Here we compare the two approaches in terms of their design.

A2.1 Scope

SOILR is focused solely on soil organic matter decomposition, while SIMECOL aims to be a general framework for ecological modeling in R. SIMECOL represents an abstraction level somewhere between SOILR and R's S4 system. Hence, the obvious way for SOILR to make use of SIMECOL would be to implement our `Model` class as a subclass of SIMECOL 's `simObj`. One motivation to do so consists in the possibility to inherit functionality from the ancestor which is usually additionally shared by other subclasses thus enabling easier communication between them. Whether subclassing is the appropriate strategy thus depends on the amount of inheritable code and the number of subclasses. We found that within SOILR, `Model` would be the only one subclass of SIMECOL's model with nearly no code shared owing to differences in functionality and design goals (see below).

A2.2 Functionality

One major difference with SIMECOL's `simObj` is that SOILR's `Model` initialize method requires objects of class `TimeMap` or its subclasses, thus guaranteeing that the different objects conforming a model share a common support in the time domain; even if the creation of `TimeMap` objects usually takes place far away from the initialization of a `Model` object since this usually happens behind the scenes, e.g. by conversion of `data.frames` the user provides as input. This is a significant difference from SIMECOL's approach. Implementing this functionality in a subclass of SIMECOL's model would be misleading and would not conform well with our design goals.

A2.3 Design goals

The main aim of SOILR's `Model` class is to provide a fence between interface and implementation. While the interface should change as little as possible to protect code using the class, we want to have as much freedom as possible to change its implementation. We strive to achieve this by avoiding every possible syntactic distinction between references to object data and object methods from the outside. With R's S4 system, one (and maybe the only) way to do so is to wrap all access to object data with get and set methods. This is one aspect of a concept often referred to as *encapsulation* and regarded as one of the key components of OOP. In languages such as Smalltalk (Kay, 1993) or Ruby (Matsumoto and Ishituka, 2002), fully dedicated to OOP, this approach is mandatory: A method call (a message send to an object) is

the one and only way for objects to communicate with each other.

The S4 approach in R is much less strict and allows direct manipulation of the data belonging to an object by means of the operator @ (or the slot function respectively), thus revealing the implementation of an object to the outside code using this object. User code based on @ will break if the implementation changes and the desired slot disappears, while a get method may be able to retrieve the desired information from other data. The use of the slot operator from outside should therefore be avoided (Genolini, 2008; Gentleman, 2003). Imagine for instance a `circle` object `c` that *stores* its radius and *computes* the diameter when `getDiameter(c)` is called. If the radius is accessed solely by calls of `getRadius(c)` the implementation of the circle class can later be changed to store the diameter and compute the radius *without* any affect to outside code using the class.

Applications to SOILR's `Model` occur if one considers the increasing number of constructors with redundant parameter sets. The distinction between parameters and derived variables may even become difficult. Differing from SIMECOL we unify access to `Model` elements and avoid a syntactic difference of parameters and derived values as expressed by SIMECOL's functions `params` and `out`. In addition to the `get...` functions we provide for all data intended to be accessed from outside, access through the typical operators `[]` and `$` for our `Model` class. This gives access to all data the model can provide, either during the initialization or computed by the methods of the class. In fact, `[]` and `$` internally use the `get...` methods and may perform any number of checks and computations to provide a consistent output. This facility is powerful enough to avoid the use of @ completely for our class. This is exactly our intention, since the implementation of the class may change, accessing its elements via @ is strongly discouraged. While we will maintain the defined interface via `[]`, `$` and the `get...` functions we cannot guarantee that code relying on @ will work in future versions of SOILR.

Another difference with SIMECOL is also related to the `[]` operator. It enables the user to create the desired output on demand, thus avoiding duplication between output objects. If the user asks for `"C"` she/he gets `"C"` and not `"time"` and `"C"`, which sometimes may not be intended if the user is concerned about issues of memory or duplication. However, it may be very useful to have both in other situations when the `"times"` argument to the constructor that created the model in the first place is not available anymore. With `[]` the user can decide which fields the output should contain.

An additional intended feature of our `[]` operator is to always produce `data.frames` and not a `"Result"` object that would (or at least should) be more protective of its data against alterations post simulation than a `data.frame`. If we provided code to post-process results we would certainly implement such a `"Result"` class to make sure that our post-processing is based on valid data. Since at the moment we don't, we also do not force a structured output object to the user.

At the moment we do not even provide output as `deSolve` objects, since every structured output is part of the interface, and may be relied on by user code. Although we are happy and thankful to use `deSolve` extensively within SOILR, making it a part of the interface would be a promise to the user that we will keep on supporting it, which is a much bigger commitment.

Until now the responsibility beyond the scope of class `Model` stays with the user.

Supplementary material related to this article is available online at: http://www.geosci-model-dev.net/5/1045/2012/gmd-5-1045-2012-supplement.zip.

Acknowledgements. Financial support for the development of this project was provided by the Max Planck Society. We would like to thank Ulrich Weber and Martin Jung for providing global climate data used in the example. Maarten Braakhekke provided helpful comments on a previous version of this manuscript.

The service charges for this open access publication have been covered by the Max Planck Society.

Edited by: S. Arndt

References

Adair, E., Parton, W., Del Grosso, S., Silver, W., Harmon, M., Hall, S., Burke, I., and Hart, S.: Simple three-pool model accurately describes patterns of long-term litter decomposition in diverse climates, Glob. Change Biol., 14, 2636–2660, doi:10.1111/j.1365-2486.2008.01674.x, 2008.

Ågren, G. and Bosatta, E.: Theoretical ecosystem ecology: understanding element cycles, Cambridge University Press, 1998.

Andren, O. and Katterer, T.: ICBM: The Introductory Carbon Balance Model for Exploration of Soil Carbon Balances, Ecol. Appl., 7, 1226–1236, http://www.jstor.org/stable/2641210, 1997.

Bolker, B. M., Pacala, S. W., and Parton, W. J.: Linear analysis of soil decomposition: insights from the CENTURY model, Ecol. Appl., 8, 425–439, doi:10.1890/1051-0761(1998)008[0425:LAOSDI]2.0.CO;2, 1998.

Bruun, S., Six, J., Jensen, L., and Paustian, K.: Estimating turnover of soil organic carbon fractions based on radiocarbon measurements, Radiocarbon, 47, 99–113, 2008.

Burger, M., Juenemann, K., and Koenig, T.: RUnit: R Unit test framework, R package version 0.4, 25, 2009.

Burke, I., Kaye, J., Bird, S., Hall, S., McCulley, R., and Sommerville, G.: Evaluating and testing models of terrestrial biogeochemistry: the role of temperature in controlling decomposition, Models in ecosystem science, Princeton University Press, Princeton, New Jersey, USA, 225–253, 2003.

Chambers, J.: Software for data analysis: Programing with R, Springer Verlag, 2008.

Coleman, K. and Jenkinson, D.: ROTHC-26.3 A model for the turnover of carbon in soils, Rothamsted Research, 1999.

Cramer, W., Bondeau, A., Woodward, F. I., Prentice, I. C., Betts, R. A., Brovkin, V., Cox, P. M., Fisher, V., Foley, J. A., Friend, A. D., Kucharik, C., Lomas, M. R., Ramankutty, N., Sitch, S., Smith, B., White, A., and Young-Molling, C.: Global response of terrestrial ecosystem structure and function to CO_2 and climate change: results from six dynamic global vegetation models, Glob. Change Biol., 7, 357–373, doi:10.1046/j.1365-2486.2001.00383.x, 2001.

Easter, M., Paustian, K., Killian, K., Williams, S., Feng, T., Al-Adamat, R., Batjes, N., Bernoux, M., Bhattacharyya, T., Cerri, C., Cerri, C., Coleman, K., Falloon, P., Feller, C., Gicheru, P., Kamoni, P., Milne, E., Pal, D., Powlson, D., Rawajfih, Z., Sessay, M., and Wokabi, S.: The GEFSOC soil carbon modelling system: A tool for conducting regional-scale soil carbon inventories and assessing the impacts of land use change on soil carbon, Agr. Ecosyst. Environ., 122, 13–25, doi:10.1016/j.agee.2007.01.004, 2007.

Edgewall Software: The Trac User and Administration Guide 0.12, available at: http://trac.edgewall.org/wiki/TracGuide (last access: 12 July 2011), 2011.

Foley, J.: An equilibrium model of the terrestrial carbon budget, Tellus B, 47, 310–319, 2011.

Genolini, C.: A (Not So) Short Introduction to S4, available at: http://cran.r-project.org/doc/contrib/Genolini-S4tutorialV0-5en.pdf (last access: 22 August 2012), 2008.

Gentleman, R.: S4 Classes in 15 pages, more or less, available at: http://www.stat.auckland.ac.nz/S-Workshop/Gentleman/S4Objects.pdf (last access: 22 August 2012), 2003.

Grosso, S. D., Parton, W., Mosier, A., Holland, E., Pendall, E., Schimel, D., and Ojima, D.: Modeling soil CO_2 emissions from ecosystems, Biogeochem., 73, 71–91, doi:10.1007/s10533-004-0898-z, 2005.

Harmon, M. and Domingo, J.: A user's guide to STANDCARB version 2.0: a model to simulate the carbon stores in forest stands, Department of Forest Science, Oregon State University, Corvallis, Oregon, 2001.

Held, I. M.: The Gap between Simulation and Understanding in Climate Modeling, B. Am. Meteorol. Soc., 86, 1609–1614, doi:10.1175/BAMS-86-11-1609, 2005.

Henin, S., Monnier, G., and Turc, L.: Un aspect de la dynamique des matieres organiques du sol, Comptes rendus hebdomadaires des séances de l'Académie des sciences, 248, 138–141, 1959.

Hocking, T. D., Wutzler, T., Ponting, K., and Grosjean, P.: Sustainable, Extensible Documentation Generation using inlinedocs, J. Stat. Softw., in press, 2012.

Jacobson, M.: Earth system science: from biogeochemical cycles to global change, Academic Press, 2000.

Jenkinson, D. S. and Rayner, J. H.: The Turnover of Soil Organic Matter in Some of the Rothamsted Classical Experiments, Soil Sci., 123, 298–305, 1977.

Jenkinson, D. S., Andrew, S. P. S., Lynch, J. M., Goss, M. J., and Tinker, P. B.: The Turnover of Organic Carbon and Nitrogen in Soil, Philosoph. Trans., 329, 361–368, 1990.

Jenny, H., Gessel, S. P., and Bingham, F. T.: Comparative Study of Decomposition Rates of Organic Matter in Temperate and Tropical Regions, Soil Sci., 68, 419–432, 1949.

Jost, J.: Dynamical systems: examples of complex behaviour, Springer Verlag, Berlin, 2005.

Kay, A.: The early history of Smalltalk, ACM, available at: http://www.smalltalk.org/smalltalk/TheEarlyHistoryOfSmalltalk_Abstract.html (last access: 22 August 2012), 1993.

Kelly, R. H., Parton, W. J., Hartman, M. D., Stretch, L. K., Ojima, D. S., and Schimel, D. S.: Intra-annual and interannual variability of ecosystem processes in shortgrass steppe, J. Geophys. Res., 105, 20093–20100, doi:10.1029/2000JD900259, 2000.

Kirschbaum, M. U.: The temperature dependence of soil organic matter decomposition, and the effect of global warming on soil organic C storage, Soil Biol. Biochem., 27, 753–760, doi:10.1016/0038-0717(94)00242-S, 1995.

Knuth, D. E.: Literate Programming, Comput. J., 27, 97–111, 1984.

Knutti, R.: The end of model democracy?, Climatic Change, 102, 395–404, doi:10.1007/s10584-010-9800-2, 2010.

Knutti, R., Furrer, R., Tebaldi, C., Cermak, J., and Meehl, G. A.: Challenges in Combining Projections from Multiple Climate Models, J. Climate, 23, 2739–2758, doi:10.1175/2009JCLI3361.1, 2009.

Leisch, F.: Sweave, part I: Mixing R and LaTeX, R News, 2, 28–31, 2002.

Leisch, F.: Sweave, part II: Package vignettes, R News, 3, 21–24, 2003.

LeVeque, R.: Finite difference methods for ordinary and partial differential equations, Society for Industrial and Applied Mathematics, Philadelphia, 2007.

Lloyd, J. and Taylor, J. A.: On the Temperature Dependence of Soil Respiration, Funct. Ecol., 8, 315–323, 1994.

Luo, Y. and Weng, E.: Dynamic disequilibrium of the terrestrial carbon cycle under global change, Trends Ecol. Evolut., 26, 96–104, doi:10.1016/j.tree.2010.11.003, 2011.

Manzoni, S. and Porporato, A.: A theoretical analysis of nonlinearities and feedbacks in soil carbon and nitrogen cycles, Soil Biol. Biochem., 39, 1542–1556, 2007.

Manzoni, S. and Porporato, A.: Soil carbon and nitrogen mineralization: Theory and models across scales, Soil Biol. Biochem., 41, 1355–1379, doi:10.1016/j.soilbio.2009.02.031, 2009.

Manzoni, S., Katul, G. G., and Porporato, A.: Analysis of soil carbon transit times and age distributions using network theories, J. Geophys. Res., 114, doi:10.1029/2009JG001070, 2009.

Matsumoto, Y. and Ishituka, K.: Ruby programming language, Addison Wesley Publishing Company, 2002.

Means, J. E., Cromack Jr., K., and MacMillan, P. C.: Comparison of decomposition models using wood density of Douglas-fir logs, Can. J. Forest Res., 15, 1092–1098, doi:10.1139/x85-178, 1985.

Melillo, J., Borchers, J., and Chaney, J.: Vegetation/ecosystem modeling and analysis project: Comparing biogeography and geochemistry models in a continental-scale study of terrestrial ecosystem responses to climate change and CO_2 doubling, Global Biogeochem. Cy., 9, 407–438, 1995.

Minderman, G.: Addition, Decomposition and Accumulation of Organic Matter in Forests, J. Ecol., 56, 355–362, 1968.

Nikiforoff, C.: Some general aspects of the chernozem formation, Soil Sci. Soc. Amer. J., 1, 333–342, 1936.

Olson, J. S.: Energy Storage and the Balance of Producers and Decomposers in Ecological Systems, Ecology, 44, 322–331, 1963.

O'Sullivan, B.: Mercurial: the definitive guide, Oreilly & Associates Inc, available at: http://mercurial.selenic.com (last access: 22 August 2012), 2009.

Parton, W., Schimel, D., Cole, C., and Ojima, D.: Analysis of factors controlling soil organic matter levels in Great Plains grasslands, Soil Sci. Soc. Amer. J., 51, 1173–1179, 1987.

Parton, W. J., Morgan, J. A., Kelly, R. H., and Ojima, D. S.: Modeling soil C responses to environmental change in grassland systems, in: The potential of US grazing lands to sequester carbon and mitigate the greenhouse effect, edited by: Follett, R., Kimble, J., and Lal, R., CRC, 371–398, 2001.

Paul, E. and Clark, F.: Soil microbiology and biochemistry, Academic Press, San Diego, USA, 1996.

Paustian, K., Ågren, G., and Bosatta, E.: Modelling litter quality effects on decomposition and soil organic matter dynamics, in: Driven by nature: plant litter quality and decomposition, edited by: Cadisch, G. and Giller, K. E., CABI Publishing, UK, 313–335, 1997.

Petzoldt, T. and Rinke, K.: simecol: An Object-Oriented Framework for Ecological Modeling in R, Journal of Statistical Software, 22, 1–31, available at: http://www.jstatsoft.org/v22/i09, 2007.

R Development Core Team: R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, available at: http://www.R-project.org/, ISBN 3-900051-07-0, 2011.

Randerson, J. T., Hoffman, F. M., Thornton, P. E., Mahowald, N. M., Lindsay, K., Lee, Y.-H., Nevison, C. D., Doney, S. C., Bonan, G., Stöckli, R., Covey, C., Running, S. W., and Fung, I. Y.: Systematic assessment of terrestrial biogeochemistry in coupled climate–carbon models, Glob. Change Biol., 15, 2462–2484, doi:10.1111/j.1365-2486.2009.01912.x, 2009.

Schlesinger, W.: Biogeochemistry: An Analysis of Global Change, Academic Press, 1997.

Sierra, C. A., Harmon, M. E., and Perakis, S. S.: Decomposition of heterogeneous organic matter and its long-term stabilization in soils, Ecol. Monogr., 81, 619–634, doi:10.1890/11-0811.1, 2011.

Soetaert, K., Petzoldt, T., and Setzer, R. W.: Solving Differential Equations in R: Package deSolve, Journal of Statistical Software, 33, 1–25, available at: http://www.jstatsoft.org/v33/i09, 2010.

Swift, M., Heal, O., and Anderson, J.: Decomposition in terrestrial ecosystems, University of California Press, available at: http://books.google.de/books?id=hSWMkhdSfPAC, 1979.

SymPy Development Team: SymPy–Python library for symbolic mathematics, Tech. rep., Technical report (since 2006), available at: http://code.google.com/p/sympy/(last access: November 2009), 2008.

Van't Hoff, J.: Etudes de dynamique chimique, Frederick Muller & Co., Amsterdam, 1884.

Weedon, G. P., Gomes, S., Viterbo, P., Shuttleworth, W. J., Blyth, E., Österle, H., Adam, J. C., Bellouin, N., Boucher, O., and Best, M.: Creation of the WATCH Forcing Data and Its Use to Assess Global and Regional Reference Crop Evaporation over Land during the Twentieth Century, J. Hydrometeorol., 12, 823–848, doi:10.1175/2011JHM1369.1, 2011.

Wu, L. and McGechan, M.: A Review of Carbon and Nitrogen Processes in Four Soil Nitrogen Dynamics Models, J. Agr. Eng. Res., 69, 279–305, doi:10.1006/jaer.1997.0250, 1998.

Wutzler, T. and Reichstein, M.: Colimitation of decomposition by substrate and decomposers – a comparison of model formulations, Biogeosciences, 5, 749–759, doi:10.5194/bg-5-749-2008, 2008.