C-Stability Java library

Version 1.0

User guide and documentation

Julien Sainte-Marie

September 15, 2021

Contents

Chapter 1

User guide

Contents

This chapter describes how to use the Java library implementing the model C-STABILITY. It is assumed that the reader already know the theoretical framework of the model. Otherwise, see chapter 2 for a complete description of the model theory.

1.1 Content of the archive cstability.zip

- Documentation_of_C-STABILITY_v.1.0.pdf : file of this documentation
- cstability.jar : source code
- cstability.bat : file to use C-STABILITY on Windows system
- cstability.sh : file to use C-STABILITY on Linux/MacOs systems
- configuration_file_template.csv : an example of configuration file
- generate_figures.py : an util to make some figure of the results

1.2 Software requirements

1.2.1 Java

C-STABILITY library is implemented with Java SE 8 (there is no warranty for above versions) with a licence LGPL (see chapter A). It may be used on machines under Windows, Linux or Mac OS X. To use the library ensure that java 1.8 is installed on your computer. To check the correct version of Java is installed and available, open a Terminal and check the answer to this command,

```
java -version
```

You should get this kind of answer (note the "1.8.x"),

```
java version "1.8.0_101"
Java(TM) SE Runtime Environment (build 1.8.0_101-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.101-b13, mixed mode)
```

If java is not found on your system or the version is not 1.8.x, please download and install the correct version of java for your operating system.

1.2.2 Python3

A Python3 (`https://www.python.org/`) script named generate_figures.py is stored in cstability.zip. The script requires the installation of the Python3 library Matplotlib (`https://matplotlib.org/stable/index.html`).

1.3 Configuration file

1.3.1 Formatting rules

A configuration file is a *.csv file containing the specifications of the simulation. The file has to be edited according to the following rules:

- empty lines and lines starting with "#" are skipped by the program.
- simple values are defined with "=" and a blank separator.
- complex lines starts with a specific flag and all fields are separated by a tabulation noted "\t" in documentation.

A template configuration file *configuration_file_template.csv* is given in the archive.

1.3.2 File structure

The configuration file is composed of sections ordered specifically,

1. Simulation timeline, see section 1.3.4.1,

2. Numerical scheme and methods, see section 1.3.4.2,
3. Biochemical classes, see section 1.3.4.3,
4. Substrate pools, see section 1.3.4.4,
5. Enzyme traits, see section 1.3.4.5,
6. Microbe signature, see section 1.3.4.6,
7. Microbe enzymatic production, see section 1.3.4.7,
8. Microbe assimilation, see section 1.3.4.8,
9. Microbe mortality, see section 1.3.4.9,
10. Substrate pool transfers, see section 1.3.4.10,
11. Substrate pools initialization, see section 1.3.4.11,
12. Substrate pools temporal inputs, see section 1.3.4.12,
13. Microbe initial state, see section 1.3.4.13,
14. Observations, see section 1.3.4.14.

Please note that the order of the sections must be respected otherwise the simulation will fail.

1.3.3 Available functions

The configuration file defines in particular several model functions. A set of generic functions can be used in different sections. We give here their general formulations and are specified in the next sections each time they can be called. All functions f have the same structure in C-STABILITY. They always have parameters (p), context (c), state (s) and variables (v) as inputs. The latter are determined by the context of use and are not precised by the user. For more information about model structure see section 3.1.

1.3.3.1 Constant

The mathematical formulation is,

$$f(p, c, s, v) = \alpha.$$

Formally we have,

```
constant(α)
```

where $\alpha \in \mathbb{R}$. By example,

```
constant(0.3)
```

1.3.3.2 Linear

The mathematical formulation is,

$$f(p, c, s, v) = \alpha v.$$

Formally we have,

```
linear(α)
```

where $\alpha \in \mathbb{R}$. By example,

```
linear(0.4)
```

1.3.3.3 UniformLinear

The mathematical formulation is,

$$f(p, c, s, v_1, v_2) = \alpha \; 1_{\mathcal{D}}(v_1) \; v_2.$$

Formally we have,

```
uniformLinear(α,D)
```

where $\alpha \in \mathbb{R}$ and $\mathcal{D}$ is an interval. By example,

```
uniformLinear(1;[0,0.4])
```

1.3.3.4 Gaussian

The mathematical formulation is,

$$f(p, c, s, v) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\left(\frac{v-\mu}{\sigma}\right)^2}.$$

Formally we have,

```
gaussian(μ;σ)
```

where $\mu \in \mathbb{R}$ the mean and $\sigma \in \mathbb{R}^+$ the standard deviation. By example,

```
gaussian(1;0.4)
```

1.3.4 File sections

1.3.4.1 Simulation timeline

This file section should contain the time unit written timeUnit (string), the initial date written initialDate (integer) and the final date written finalDate (integer) of the simulation. By example we have,

```
timeUnit = day
initialDate = 0
finalDate = 200
```

1.3.4.2 Numerical scheme and methods

This file section describes the numerical rules used to solve the model equation. Depending on the precision needed, simulation could be resources consuming. The section contains a time discretization written userTimeStep (double) in]0,1[, a polymerization discretization written userPolymerizationStep (double), and a numerical integration method for distributions written integrationMethod (String) chosen between INTEGRATION_TRAPEZE, INTEGRATION_RECTANGLE_LEFT and INTEGRATION_RECTANGLE_RIGHT. By example we have,

```
userTimeStep = 0.1
userPolymerizationStep = 0.01
integrationMethod = INTEGRATION_TRAPEZE
```

1.3.4.3 Biochemical classes

This file section should contains at least two biochemical classes. The line format is

```
BIOCHEMICAL_CLASS \t bcName \t polymerization
```

where bcName (String) is the name of the biochemical class and polymerization is the interval $[p^{min}_{bcName}, p^{max}_{bcName}]$ used to describe polymerization. By example we have,

```
BIOCHEMICAL_CLASS \t cellulose \t [0,2]
```

1.3.4.4 Substrate pools

This file section should contain at least an accessible pool per biochemical class. The line format is

```
POOL_ACCESSIBILITY \t bcName \t accessibility
```

where bcName (String) has to be defined previously and accessibility is a list of accessibility status available among the following proposition, ACCESSIBLE, INACCESSIBLE_AGGREGATION, INACCESSIBLE_MINERAL_ASSOCIATION, and INACCESSIBLE_MINERAL_ASSOCIATION. By example we have,

```
POOL_ACCESSIBILITY \t cellulose \t [ACCESSIBLE,INACCESSIBLE_AGGREGATION]
```

1.3.4.5 Enzyme traits

1.3.4.5.1 Line format This section permits to define the simulation enzymes traits. The line format is

```
ENZYME_TRAITS \t enzName \t bcName \t depolymerizationRateFunction
                                    ...\t kernelFunction \t kernelIntegrationMethod
```

where enzName (String) is the name of the enzyme, bcName (String) is biochemical class to which the enzyme is associated, depolymerizationDomain (Interval) is the polymerization interval of the biochemical class targeted by the enzyme, depolymerizationRateFunction (String) is the function defining the depolymerization rate of the enzyme (eq. (2.11)), kernelFunction (String) is the function defining the transformation kernel of the enzyme (eq. (2.9)), and kernelIntegrationMethod (String) is the numerical method used to compute enzymatic transformation (section 3.2.3) and has to be chosen between STANDARD_KERNEL_INTEGRATION and INTEGRAL_KERNEL_INTEGRATION. Available functions are given in sections 1.3.4.5.2 and 1.3.4.5.3. By example we have,

```
ENZYME_TRAITS \t cellulolase \t cellulose \t [0,2] \t uniformLinear([0,2];1.8)
                        ...\t kernelAlpha([0,2];5) \t INTEGRAL_KERNEL_INTEGRATION
```

1.3.4.5.2 Available depolymerization rate functions Depolymerization rate functions have arguments of type $f(p, c, s, v_1, v_2)$ where $v_1 = p_{bcName}, v_2 = \pi_{enz}(t)$.

1.3.4.5.2.1 UniformLinear Formally we have,

```
uniformLinear(α,𝒟)
```

and in eq. (2.11) gives,

$$\tau_{enz}(p_{bcName}, t) = \alpha \; 1_{\mathcal{D}}(p_{bcName}) \; \pi_{enz}(t).$$

1.3.4.5.3 Available kernel functions Kernel functions have arguments of type $f(p, c, s, v_1, v_2)$ where $v_1 = p_{bcName}, v_2 = p'_{bcName}$.

1.3.4.5.3.1 KernelAlpha Formally we have,

```
kernelAlpha(𝒟;α)
```

where $\mathcal{D} = [p_{min}^{bcName}, p_{max}^{bcName}]$ is an interval and $\alpha \in \mathbb{R}^+$, by example we have,

```
kernelAlpha([0,1];2.1)
```

The mathematical formulation associated to eq. (2.9) is for all $p, p' \in [p_{min}^{bcName}, p_{max}^{bcName}]$,

$$\mathcal{K}_{enz}(p, p') = 1_{\left\{p_{min}^{bcName} < p \leq p' < p_{max}^{bcName}\right\}}(\alpha + 1)\frac{(p - p_{min}^{bcName})^{\alpha}}{(p' - p_{min}^{bcName})^{\alpha+1}}.$$

In addition, the primitive according to p'_{bcName} is available and for all $p, p' \in [p_{min}^{bcName}, p_{max}^{bcName}]$,

$$\mathbb{K}_{enz}(p, p') = -1_{\left\{p_{min}^{bcName} < p \leq p' < p_{max}^{bcName}\right\}}\frac{(\alpha + 1)}{\alpha}\left(\frac{p - p_{min}^{bcName}}{p' - p_{min}^{bcName}}\right)^{\alpha}.$$

1.3.4.6 Microbe signature

1.3.4.6.1 Line format This section describes the microbes involved in the simulation and their biochemical and polymerization composition. The line format is

```
SIGNATURE \t micName \t bcName \t proportion \t polymerizationFunction
```

where micName (string) is the name of the microbial community considered, bcName (string) is the name of the biochemical class composing the microbe, proportion (double in [0,1]) is the proportion of the components of the considered biochemical class composing the microbes, and polymerizationFunction (String) is the function describing the polymerization of biochemical elements considered (section 2.1.3). Available functions are given in section 1.3.4.6.2. By example we have,

```
SIGNATURE \t cellulose_decomposer \t microbe_sugar
            ...\t 1 \t  gaussianTruncatedNormalized(microbe_sugar;1.5;0.1;[0,2])
```

1.3.4.6.2 Available function Polymerization functions have arguments of type $f(p, c, s, v)$ where $v = p_{bcName} \in [p_{bcName}^{min}, p_{bcName}^{max}]$.

1.3.4.6.2.1 GaussianTruncatedNormalized Formally we have,

```
gaussianTruncatedNormalized(bcName;μ;σ;𝒟)
```

where $bcName$ is the biochemical class, $\mu \in \mathbb{R}$ the mean, $\sigma \in \mathbb{R}^+$ the standard deviation and $\mathcal{D}$ is the interval of truncation e.g.

```
gaussianTruncatedNormalized(microbe_sugar;1;0.4;[0,2])
```

The mathematical formulation associated to eq. (2.4) is,

$$s_{mic,bcName}(p_{bcName}) = 1_{\mathcal{D}}(p_{bcName})\frac{e^{-\left(\frac{p_{bcName}-\mu}{\sigma}\right)^2}}{\int_{\mathcal{D}} e^{-\left(\frac{p-\mu}{\sigma}\right)^2} dp}.$$

The parameter $bcName$ is mandatory to explicit the integration rule to compute the integral.

1.3.4.7 Microbe enzymatic production

1.3.4.7.1 Line format This section describes how enzymes are produced by the microbes. The line format is,

```
ENZYME_PRODUCTION \t micName \t producedEnzyme \t productionRateFunction
```

where micName (String) is the name of the microbial community considered, producedEnzyme (String) is the name of the enzymes produces by the microbes, productionRateFunction (String) is the function describing the production rate of the enzyme (eq. (2.10) and section 1.3.3 for available functions). By example we have,

```
ENZYME_PRODUCTION \t cellulose_decomposer \t cellulase \t linear(1.)
```

1.3.4.7.2 Available function Enzyme production functions have arguments of type $f(p, c, s, v)$ where $v = C_{mic}$.

1.3.4.7.2.1 Linear Formally we have,

```
linear(α)
```

and gives in eq. (2.10),

$$\pi_{enz}^{mic}(C_{mic}(t)) = \alpha \; C_{mic}(t).$$

1.3.4.8 Microbe assimilation

1.3.4.8.1 Line format This section describes how microbes assimilate C from the accessible substrate pools. Assimilation of each microbe has to be defined. The line format is,

```
ASSIMILATION \t micName \t bcName
                            ...\t uptakeFluxFunction \t carbonUseEfficiencyFunction
```

where micName (String) is the name of the microbial community considered, bcName (String) is the biochemical class where carbon is taken up by microbes, uptakeFluxFunction (String) is the function describing the rate of uptake of carbon (eq. (2.6) and section 1.3.3 for available functions), carbonUseEfficiencyFunction (String) is the function describing the carbon use efficiency of the taken up carbon (eq. (2.7) and section 1.3.3 for available functions). By example we have,

```
ASSIMILATION \t cellulose_decomposer \t cellulose
                                ...\t uniformLinear([0,0.4];5.) \t constant(0.4)
```

1.3.4.8.2 Available uptake flux function Microbial uptake flux functions have arguments of type $f(p, c, s, v_1, v_2)$ where $v_1 = p_{bcName}, v_2 = \chi_{bcName}$.

1.3.4.8.2.1 UniformLinear Formally we have,

```
uniformLinear(α,𝒟)
```

and gives in eq. (2.6),

$$F_{mic,bcName}^{upt}(\chi_{bcName}^{ac}, p_{bcName}, t) = \alpha \; 1_{\mathcal{D}}(p_{bcName}) \; \chi_{bcName}^{ac}(t).$$

1.3.4.8.3 Available carbon use efficiency functions Carbon use efficiency functions have arguments of type $f(p, c, s, v_1, v_2)$ where $v_1 = p_{bcName}, v_2 = \chi_{bcName}$.

1.3.4.8.3.1 Constant Formally we have,

```
constant(α)
```

and gives in eq. (2.7),

$$e_{mic,bcName}(t) = \alpha.$$

1.3.4.9 Microbe mortality

1.3.4.9.1 Line format This section describes the mortality rate of the microbes. The mortality of each microbe has to be defined. The line format is,

```
MORTALITY \t micName \t mortalityFunction
```

where micName (String) is the name of the microbial community considered, mortalityFunction (String) is the function describing the mortality rate of microbes (eq. (2.8) and section 1.3.3 for available functions). By example we have,

```
MORTALITY \t cellulose_decomposer \t linear(0.02)
```

1.3.4.9.2 Available function Microbe mortality functions have arguments of type $f(p, c, s, v)$ where $v = C_{mic}$.

1.3.4.9.2.1 Linear Formally we have,

```
linear(α)
```

and gives in eq. (2.8),

$$m_{mic}\,(t, C_{mic}) = \alpha\ C_{mic}(t).$$

1.3.4.10 Substrate pool transfers

1.3.4.10.1 Line format This section is optional if the simulations contains only ACCESSIBLE pools. For a same biochemical class, exchange between accessible and inaccessible pool can occur. The line format is,

```
POOL_TRANSFER \t bcName \t origin \t arrival \t transferFunction
```

where bcName (String) is the biochemical class considered, origin (String) is the accessibility of the origin pool, arrival (String) is the accessibility of the arrival pool, transferFunction (String) is the function describing the transfer rate between the pools (eqs. (2.15) and (2.16)). Available functions are given in section 1.3.4.10.2. By example we have,

```
POOL_TRANSFER \t cellulose \t INACCESSIBLE_EMBEDMENT
                        ...\t ACCESSIBLE \t enzymaticLinearTransfer(lignolase;1)
```

1.3.4.10.2 Available transfer function Pool transfer functions have arguments of type $f(p, c, s, v)$ where $v = \chi_{bcName}(p_{bcName}, t)$.

1.3.4.10.2.1 EnzymaticLinearTransfer Formally we have,

```
enzymaticLinearTransfer(enz; α)
```

where enz is the enzyme name and $\alpha \in \mathbb{R}^+$ by example we have,

```
enzymaticLinearTransfer(cellulase;1)
```

The mathematical formulation associated to eqs. (2.15) and (2.16) is,

$$F^{loc}_{in,bcName}(p_{bcName}, t) = \alpha\ \omega_{enz}\ \chi_{bcName}(p_{bcName}, t)$$

where ω_{enz} is defined in eq. (2.12) and is stored in model state s.

1.3.4.11 Substrate pools initialization

1.3.4.11.1 Line format This section contains the initial biomass of each substate pool. The line format is,

```
POOL_INITIALIZATION \t bcName \t accessibility \t carbonPolymerization
```

where bcName (String) is the biochemical class considered, accessibility (String) is the accessibility of the pool, carbonPolymerization (String) is the function describing the amount and polymerization of carbon. Available functions are given in section 1.3.4.11.2. By example we have,

```
POOL_INITIALIZATION \t cellulose \t ACCESSIBLE
            ...\t gaussianTruncatedProportionalized(cellulose;0.95;1.5;0.1;[0,2])
```

1.3.4.11.2 Available function Substrate polymerization functions have arguments of type $f(p, c, s, v)$ where $v = p_{bcName} \in [p^{min}_{bcName}, p^{max}_{bcName}]$.

1.3.4.11.2.1 GaussianTruncatedProportionalized Formally we have,

```
gaussianTruncatedProportionalized(bcName;α;μ;σ;𝒟)
```

where bcName is the biochemical class, $\alpha \in \mathbb{R}^+$ is the proportion coefficient, $\mu \in \mathbb{R}$ the mean, $\sigma \in \mathbb{R}^+$ the standard deviation and $\mathcal{D}$ is the interval of truncation e.g.

```
gaussianTruncatedProportionalized(lipid;95,1;0.4;[0,2])
```

The mathematical formulation associated is,

$$\chi_{bcName}(p_{bcName}, t_0) = \alpha\ 1_{\mathcal{D}}(p_{bcName})\ \frac{e^{-\left(\frac{p_{bcName}-\mu}{\sigma}\right)^2}}{\int_{\mathcal{D}} e^{-\left(\frac{p-\mu}{\sigma}\right)^2} dp}.$$

The parameter $bcName$ is mandatory to explicit the integration rule to compute the integral.

1.3.4.12 Substrate pools temporal inputs

This section should contains the C composition of specified pools.

1.3.4.12.1 Line format

```
POOL_INPUT \t bcName \t accessibility \t carbonInputFunction
```

where bcName (String) is the biochemical class, accessibility (String) is the accessibility of the pool considered, carbonInputFunction (String) is the function describing the amount and polymerization of the carbon input. By example we have,

```
POOL_INPUT \t cellulose \t ACCESSIBLE
    ...\t constantInput(0.1:gaussianTruncatedNormalized(cellulose;1.5;0.1;[0,2]))
```

1.3.4.12.2 Available function

1.3.4.12.2.1 constantInput

Formally we have,

```
constantInput(r:f)
```

where r (double>0) is the rate of the input in $g_C.t^{-1}$, and f describe the polymerization of the input. By example we have,

```
constantInput(0.3:gaussianTruncatedNormalized(cellulose;1.5;0.1;[0,2]))
```

The mathematical formulation associated is,

$$i_{bcName}^{accessibility}(p_{bcName}, t) = r \ f(p_{bcName}).$$

1.3.4.13 Microbe initial state

This section contains the initial biomass of each microbe. The line format is,

```
MICROBE_INITIALIZATION \t micName \t mass
```

where micName (String) is the name of the microbial community considered, and mass (double>0) is the initial amount of carbon composing microbial community. By example we have,

```
MICROBE_INITIALIZATION \t cellulose_decomposer \t 0.05
```

1.3.4.14 Observations

This section contains the observations needed by the user at the end of the simulation. If no observations are specified, all observers are generated. Depending on the type of variable needed, the line format is specified. Five types are available for state, pool, pool transfer, microbes and enzymes variables.

1.3.4.14.1 State observer

```
STATE_OBSERVER \t observableVariable \t datesToObserve
```

where observableVariable is the observed variable (chosen among "respiration"), and datesToObserve is the set of dates to observes and can be formated as follow,

- $[0, 2, 5]$ to observe date 0, 2 and 5

- $[1:2:11]$ to observe date between 1 and 11 with a step of 2
- $[1:2:14,25,33]$ which is a combination of the two previous formats

By example we have,

```
STATE_OBSERVER \t respiration \t [1:2:14,25,33]
```

1.3.4.14.2 Pool observer

```
POOL_OBSERVER \t [bcName,accessibility] \t observableVariable \t datesToObserve
```

where bcName (String) is the biochemical class considered, accessibility (String) is the accessibility of the pool considered, observableVariable is the observed variable (chosen among "mass", "mass_distribution"), and datesToObserve is define as in section 1.3.4.14.1. By example we have,

```
POOL_OBSERVER \t [cellulose,ACCESSIBLE] \t mass \t [17]
```

1.3.4.14.3 Pool transfer observer

```
POOL_TRANSFER_OBSERVER \t [bcName,originPool,arrivalPool]
                                        ...\t observableVariable \t datesToObserve
```

where bcName (String) is the biochemical class considered, originPool (String) is the accessibility of the origin pool, arrivalPool (String) is the accessibility of the arrival pool, observableVariable is the observed variable (chosen among "flux_distribution"), and datesToObserve is define as in section 1.3.4.14.1. By example we have,

```
POOL_TRANSFER_OBSERVER \t [cellulose,INACCESSIBLE_EMBEDMENT,ACCESSIBLE]
                                                ...\t flux_distribution \t [0,1]
```

1.3.4.14.4 Microbe observer

```
MICROBE_OBSERVER \t micName \t observableVariable \t datesToObserve
```

where micName (String) is the name of the considered microbes, observableVariable is the observed variable (chosen among "mass", "uptake_flux_distribution_map", "carbon_use_efficiency_distribution_map", "respiration", "mortality_flux"), and datesToObserve is define as in section 1.3.4.14.1. By example we have,

```
MICROBE_OBSERVER \t cellulose_degrader \t uptake_flux_distribution \t [1:2:9]
```

1.3.4.14.5 Enzyme observer

```
ENZYME_OBSERVER \t enzName \t observableVariable \t datesToObserve
```

where enzName is the name of the considered enzymes, observableVariable is the observed variable (chosen among "depolymerization_rate_distribution, "activity_distribution"), and datesToObserve is define as in section 1.3.4.14.1. By example we have,

```
ENZYME_OBSERVER \t cellulolysis \t activity_distribution \t [0,1]
```

1.4 Running a simulation

1.4.1 On Windows

To start C-STABILITY under Windows, if cstability\ is in the 'C:\Users\John' directory,

```
C:\> cd Users\John\cstability
C:\Users\John\cstability> cstability.bat configFilePath\configFileName
```

1.4.2 On Linux/MacOS

To start C-STABILITY under Linux or MacOSX, if cstability/ is in the '/home/John' directory

```
/$ cd /home/John/cstability
/home/John/cstability/$ sh cstability.sh configFilePath/configFileName
```

1.5 Simulation outputs

1.5.1 Result files

Observers defined in configuration file (section 1.3.4.14) are saved in a new directory created in the directory where configuration file is placed. The result directory is named after the configuration file name.

1.5.2 Graphics outputs

A Python3 script named generate_figures.py is stored in cstability.zip and can be used to generate figures for C-STABILITY outputs. The script requires the installation of the Python3 library Matplotlib. To use the script, use the terminal and execute the following command line to generate a figure for each file in the directory of observations files,

```
python3 generate_figures.py observationDirectory
```

The resulting figures are saved in a sub-directory of observation-files directory named figures. The script has two optional arguments:

- observation-files: Names of specific observation-files to plot.
- file-extension: Extension of figure files, default *.png

To have more details, use the command,

```
python3 generate_figures.py --help
```

Chapter 2

Model description

Contents

Carbon Substrate Targeted AccessiBILITY (C-STABILITY) is a general model of soil organic matter (SOM) dynamics. The model is driven by microbial activity and combines compartmental and continuous modelling approaches. The substrate is split into biochemical classes, separating pools accessible to enzymes from inaccessible ones. Within each pool a continuous approach is implemented to describe the level of substrate polymerization. The model distinguishes between substrate accessibility to microbe uptake, only possible for small oligomers, and substrate accessibility to enzymes regulated by organic matter spatial arrangement within the soil matrix or interactions with other soil components. Over time the enzyme-accessible substrate is fragmented and depolymerized until it eventually becomes accessible to microbe uptake. Part of the taken up substrate is assimilated by decomposers and its biochemistry is modified by decomposer anabolism. One of the major features of the model is the description of the enzymatic polymer breakage process. The model has been first published in Sainte-Marie et al. (2021).

2.1 Organic matter representation

2.1.1 Biochemical classes and carbon pools

The description of SOM in C-STABILITY consists of several subdivisions. First, pools of living microbes (noted C_{mic}) are separated from the substrate (noted C_{sub}). Several groups of living microbes can be considered simultaneously (e.g. bacteria, fungi, etc.) and C-STABILITY classes them into functional communities. Second, SOM is also separated between several biochemical classes (noted $*$), e.g. cellulose, lignin, lipid, protein, microbial sugar... Third for each biochemical class, substrate accessible to its enzymes (noted ac) is separated from substrate which is inaccessible (noted in) due to specific physicochemical conditions, e.g. interaction between different molecules, inclusion in aggregates, sorption on mineral surfaces, etc.

2.1.2 Substrate polymerization

Polymerization is a driver of interactions between substrate and living microbes and a continuous description of the degree of organic matter polymerization (noted p_*) is provided for each of these pools, as a distribution. The polymerization axis is oriented from the lowest to the highest degree of polymerization. A right-sided distribution corresponds to a highly polymerized substrate whereas a left-sided distribution corresponds to monomer or small oligomer forms. For each biochemical class $*$ ($*$ = cellulose, lignin, lipid, etc.), the polymerization range is identical for both accessible and inaccessible pools. p_*^{min} and p_*^{max} are the minimum and maximum degrees of polymerization of the biochemical class $*$.

The total amount of C (g_C) in the accessible and the inaccessible pools of any biochemical class is,

$$C_*^{ac} = \int_{p_*^{min}}^{p_*^{max}} \chi_*^{ac}(p_*)dp_*, \tag{2.1}$$

$$C_*^{in} = \int_{p_*^{min}}^{p_*^{max}} \chi_*^{in}(p_*)dp_*, \tag{2.2}$$

where χ_*^{ac}, χ_*^{in} ($g_C.p^{-1}$) are the polymerization distributions. Finally, the total substrate C pool is defined as the sum of all biochemical pools,

$$C_{sub} = \sum_* C_*^{in} + C_*^{ac}. \tag{2.3}$$

2.1.3 Microbial pool and signature

Each microbial group (denoted mic) produces new organic compounds from the assimilated C. After death, the composition of the necromass returning to each biochemical pool $*$ of SOM is assumed to be constant, accessible and is depicted with a set of distributions $s_{mic,*}$, named signature. Each distribution $s_{mic,*}$ (p_*^{-1}) describes the polymerization of the dead microbial compounds returning to the pool $*$. The signature is normalized and unitless to ensure mass conservation, i.e. if we note that,

$$S_{mic,*} = \int_{p_*^{min}}^{p_*^{max}} s_{mic,*}(p_*)dp_*, \tag{2.4}$$

then we have

$$\sum_* S_{mic,*} = 1. \tag{2.5}$$

2.2 Organic matter dynamics

Three processes drive OM dynamics: (i) microbial uptake, biotransformation and mortality, (ii) enzymatic activity, and (iii) changes in local physicochemical conditions. All of these processes are generally considered with a daily time step (noted d). Other time step could be considered.

2.2.1 Microbes turnover

Microbial uptake of substrate is only possible for molecules having a very small degree of polymerization. When C is taken up, a fraction is respired and the remaining is metabolized, and biotransformed into microbial molecules that return to the substrate upon microbe death. Each microbial group has a specific signature that describes its composition in terms of biochemistry and polymerization.

Accessibility to microbe uptake is described by the interval (also called domain) $\mathcal{D}^u_{mic,*} \subset [p^{min}_*, p^{max}_*]$ for each biochemical class $*$, which corresponds to small substrate compounds, monomers, dimers or trimers smaller than 600 Daltons, that microbes are able to take up. For each accessible pool of substrate, the term

$$F^{upt}_{mic,*}(\chi^{ac}_*, p_*, t), \tag{2.6}$$

describes how the microbes take up the substrate available in $\mathcal{D}^u_{mic,*}$. The carbon use efficiency,

$$e_{mic,*}(p_*), \tag{2.7}$$

is the ratio between microbe assimilated and taken up C. Depending on carbon use efficiency, taken up C is respired or assimilated and biotransformed into microbial metabolites. This induces a change in the biochemistry and polymerization described by microbe signature section 2.1.3. Finally, microbial necromass returns to the substrate pools with a specific mortality m_{mic}, which depends on the microbial C quantity,

$$F^{nec}_{mic,*}(p_*, t) = m_{mic}\,(t, C_{mic})\, s_{mic,*}(p_*). \tag{2.8}$$

2.2.2 Enzymatic activity

Enzymes have a depolymerization role, which enables the transformation of highly polymerized substrate into fragments accessible to microbes. They are specific to biochemical classes. They are not individually reported, but rather as a family of enzymes contributing to the depolymerization of a biochemical substrate (e.g. combined action of endoglucanase, exoglucanase, betaglucosidase, etc., on cellulose will be reported as cellulolytic action). Accessibility to enzymes occurs in the $\mathcal{D}_{enz}$ domain. Over time the substrate accessible to enzymes is depolymerized and its distribution shifts towards $\mathcal{D}^u_{mic,*}$ (section 2.2.1) where it eventually becomes accessible to microbe uptake. The overall functioning of each enzyme family (noted enz) is described by two terms: a depolymerization rate τ_{enz} providing the number of broken bonds per time unit and a kernel accounting for the type of substrate cleavage $\mathcal{K}_{enz}$. The term F^{act}_{enz} $(g_C.p^{-1}_*.d^{-1})$ represents the change in polymerization of χ^{ac}_* due to enzyme activity for all $p_* \in \mathcal{D}_{enz} \subset [p^{min}_*, p^{max}_*]$,

$$F^{act}_{enz}(\chi^{ac}_*, p_*, t) = -\tau_{enz}(p_*, t)\chi^{ac}_*(p_*, t) + \int_{\mathcal{D}_{enz}} \mathcal{K}_{enz}(p_*, p'_*)\tau_{enz}(p'_*, t)\chi^{ac}_*(p'_*, t)dp'_*. \tag{2.9}$$

The depolymerization rate, τ_{enz} (d^{-1}), is related to the enzymes' producers by specific productions rates π^{mic}_{enz} for each microbes producing the enzyme,

$$\pi_{enz}(t) = \sum_{mic} \pi^{mic}_{enz}(C_{mic}(t)). \tag{2.10}$$

$$\tau_{enz}(p_*, t) = \tau_{enz}(p_*, \pi_{enz}(t)). \tag{2.11}$$

We can also define the enzymatic activity ω_{enz} ($g_C.d^{-1}$),

$$\omega_{enz}(t) = \int_{\mathcal{D}_{enz}} \tau_{enz}(p_*, t)\chi_*^{ac}(p_*, t)dp_*. \tag{2.12}$$

The $\mathcal{K}_{enz}$ (p_*^{-1}) kernel provides the polymerization change from p'_* to p_*. To satisfy the mass balance, the kernel verifies for all $p'_* \in \mathcal{D}_{enz}$,

$$\int_{\mathcal{D}_{enz}} \mathcal{K}_{enz}(p_*, p'_*)dp_* = 1. \tag{2.13}$$

Then F_{enz}^{act} does not change the total C mass but only the polymerization distribution,

$$\int_{\mathcal{D}_{enz}} F_{enz}^{act}(\chi, p_*, t)dp_* = 0. \tag{2.14}$$

2.2.3 Changes in local physicochemical conditions

Changes in local substrate conditions drive exchanges between substrate accessible and inaccessible to enzymes (e.g. aggregate formation and break). The polymerization of a substrate inaccessible to its enzymes remains unchanged over time. A specific event changing the accessibility to enzymes (e.g. aggregate disruption or desorption from mineral surfaces) is modeled with a flux from the inaccessible to the accessible pool. Transfer between these pools is described by the $F_{ac,*}^{loc}$ term for each biochemistry $*$,

$$F_{ac,*}^{loc}(p_*, t) = \tau_{tr}^{ac}\chi_*^{in}(p_*, t). \tag{2.15}$$

where $\tau_{tr,*}^{ac}$ (d^{-1}) is rate of local condition change toward accessibility. Transfer in the opposite way (e.g. aggregate formation, association with mineral surfaces) is described by the $F_{in,*}^{loc}$ term,

$$F_{in,*}^{loc}(p_*, t) = \tau_{tr}^{in}\chi_*^{ac}(p_*, t) \tag{2.16}$$

where $\tau_{tr,*}^{in}$ (d^{-1}) is rate of local condition change toward inaccessibility.

2.2.4 Organic matter input

We defined time dependent distributions for carbon input fluxes. There are denoted i_*^{ac} and i_*^{in} ($g_C.p_*^{-1}.d^{-1}$) for both accessible and inaccessible pools of biochemical classes $*$. The carbon input flux per biochemical class $*$, expressed in $g_C.d^{-1}$, is

$$I_*(t) = \int_{p_*^{min}}^{p_*^{max}} \left(i_*^{in}(p_*, t) + i_*^{ac}(p_*, t)\right) dp_*, \tag{2.17}$$

and the total input flux is,

$$I(t) = \sum_* I_*(t). \tag{2.18}$$

2.2.5 General dynamics equations

The distribution dynamics for each biochemical class $*$ is obtained from eqs. (2.8), (2.9), (2.11) and (2.15) to (2.17). For all $p_* \in [p_*^{min}, p_*^{max}]$,

$$\begin{aligned}\frac{\partial \chi_*^{ac}}{\partial t}(p_*,t) &= F_{ac,*}^{loc}(p_*,t) - F_{in,*}^{loc}(p_*,t) \\ &\quad + \sum_{\text{enz}} F_{enz}^{act}(\chi_*^{ac},p_*,t) \\ &\quad + \sum_{\text{mic}} \left(F_{mic,*}^{nec}(p_*,t) - F_{mic,*}^{upt}(\chi_*^{ac},p_*,t)\right) \\ &\quad + i_*^{ac}(p_*,t),\end{aligned} \tag{2.19}$$

$$\begin{aligned}\frac{\partial \chi_*^{in}}{\partial t}(p_*,t) &= F_{in,*}^{loc}(p_*,t) - F_{ac,*}^{loc}(p_*,t) \\ &\quad + i_*^{in}(p_*,t).\end{aligned} \tag{2.20}$$

By replacing some terms by their literal expression, we have

$$\begin{aligned}\frac{\partial \chi_*^{ac}}{\partial t}(p_*,t) &= \tau_{tr}^{ac}\chi_*^{in}(p_*,t) - \tau_{tr}^{in}\chi_*^{ac}(p_*,t) \\ &\quad + \sum_{\text{enz}} \left(-\tau_{enz}(p_*,t)\chi_*^{ac}(p_*,t) + \int_{\mathcal{D}_{enz}} \mathcal{K}_{enz}(p_*,p'_*)\tau_{enz}(p'_*,t)\chi_*^{ac}(p'_*,t)dp'_*\right) \\ &\quad + \sum_{\text{mic}} \left(m_{mic}\left(t,C_{mic}\right)s_{mic,*}(p_*) - F_{mic,*}^{upt}(\chi_*^{ac},p_*,t)\right) \\ &\quad + i_*^{ac}(p_*,t),\end{aligned} \tag{2.21}$$

$$\begin{aligned}\frac{\partial \chi_*^{in}}{\partial t}(p_*,t) &= \tau_{tr}^{in}\chi_*^{ac}(p_*,t) - \tau_{tr}^{ac}\chi_*^{in}(p_*,t) \\ &\quad + i_*^{in}(p_*,t).\end{aligned} \tag{2.22}$$

The dynamics of the total substrate is ruled by,

$$\frac{dC_{sub}(t)}{dt} = \sum_* \int_{p_*^{min}}^{p_*^{max}} \left(\frac{\partial \chi_*^{ac}}{\partial t}(p_*,t) + \frac{\partial \chi_*^{in}}{\partial t}(p_*,t)\right) dp_*. \tag{2.23}$$

The dynamics of microbial C_{mic} is obtained by,

$$\frac{dC_{mic}}{dt}(t) = -m_{mic}(t,C_{mic}) + \sum_* \int_{\mathcal{D}_{mic,*}^u} F_{mic,*}^{upt}(\chi_*^{ac},p_*,t)e_{mic,*}(p_*)\chi_*^{ac}(p_*,t)dp_*, \tag{2.24}$$

and the CO_2 flux ($\mathrm{g}_C.\mathrm{d}^{-1}$) produced by the microbes is given by,

$$F_{CO_2}(t) = C_{mic}(t)\sum_* \int_{\mathcal{D}_{mic,*}^u} F_{mic,*}^{upt}(\chi_*^{ac},p_*,t)\left(1 - e_{mic,*}(p_*)\right)\chi_*^{ac}(p_*,t)dp. \tag{2.25}$$

2.3 Conceptual schemes

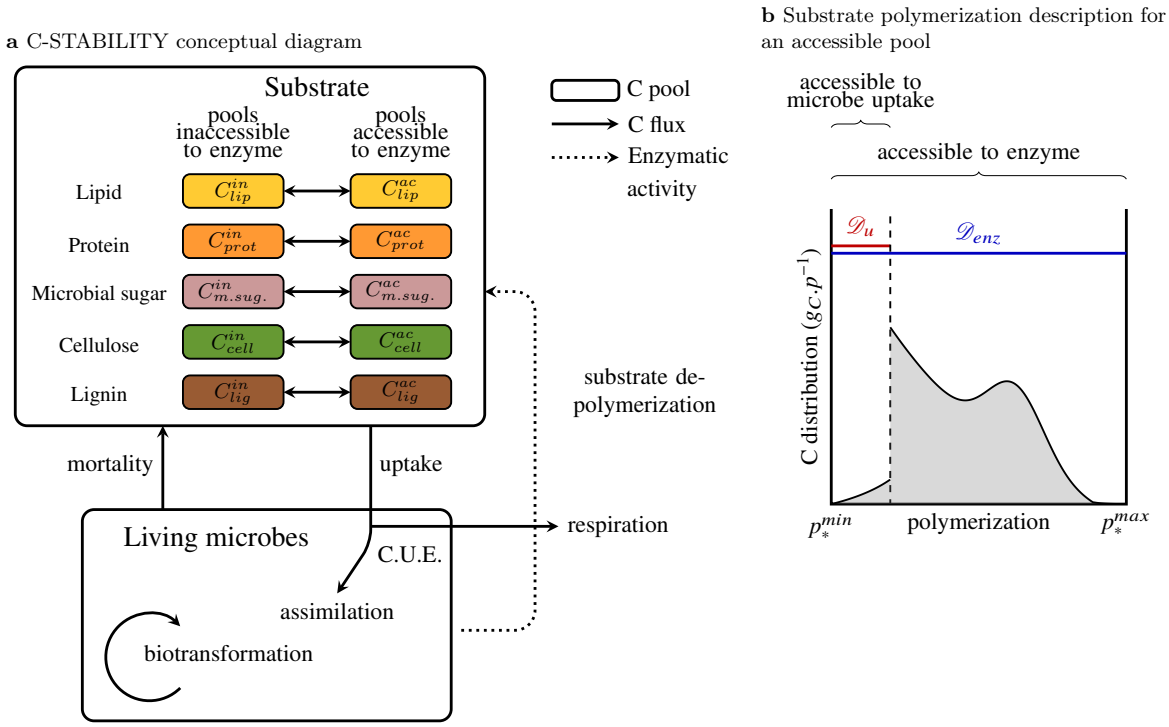

Figure 2.1: C-STABILITY model framework. **a** Conceptual diagram presenting interactions and exchanges between living microbes and organic substrate. Colored boxes stand for substrate C pools. The colors represent the different biochemical classes, and forms accessible to enzymes are separated from inaccessible ones. Plain arrows represent C fluxes, the dotted arrow represents enzymatic activity. Microbial uptake is only possible for small oligomers, which are produced by enzymes. C.U.E. means carbon use efficiency. Assimilated C is biotransformed by microbes and returns to substrate through mortality. **b** Substrate polymerization (noted p) is described for each C pool. Polymerization is used to define how living microbes get access to substrate uptake. A continuous distribution in $g_C.p^{-1}$ reports the polymerization level of the C substrate of each biochemical class (denoted $*$). p_*^{min} and p_*^{max} are respectively the minimum and maximum levels of polymerization. The polymerization axis is oriented from the lowest polymerization level on the left to the highest polymerization level on the right. Here an accessible pool is presented. The same definition of polymerization stands for pool accessible and inaccessible to enzymes. The blue domain ($\mathcal{D}_{enz}$) identifies polymers accessible to enzymes. The red domain $\mathcal{D}_u$ identifies monomers and small oligomers accessible to microbe uptake. The amount of C corresponds to the area below the curve. Reprint from Sainte-Marie et al. (2021).

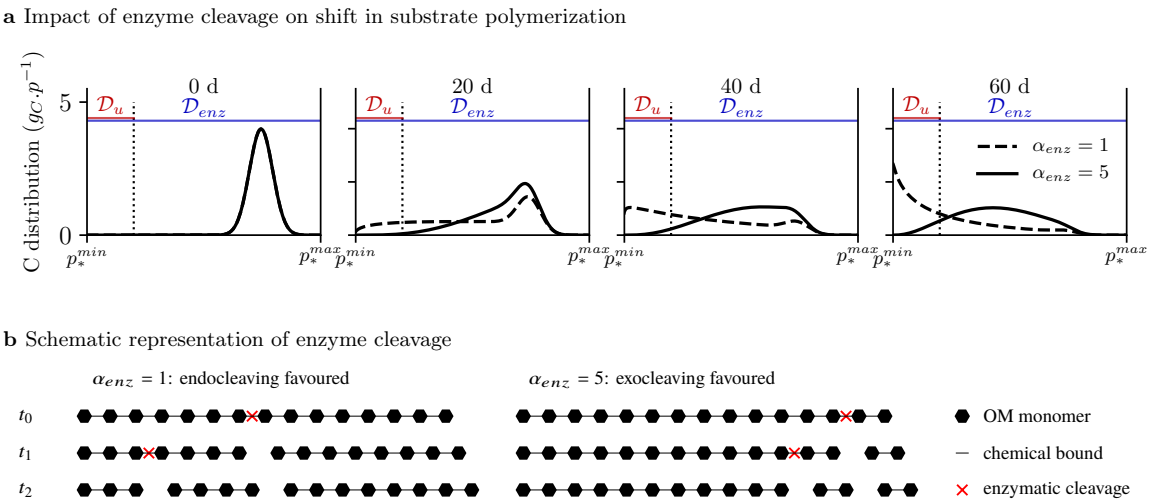

Figure 2.2: Change in substrate polymerization over time induced by enzymes for an accessible biochemical class (denoted $*$) in absence of microbial uptake. **a** A continuous distribution reports substrate polymerization, noted p. p_*^{min} and p_*^{max} are the minimum and maximum levels of polymerization. The $\mathcal{D}_{enz}$ domain (blue) indicates substrate accessible to its enzymes. The $\mathcal{D}_u$ domain (red) indicates where the substrate is accessible for microbe uptake. Here the microbe uptake rate is nil, which explains the substrate accumulation in $\mathcal{D}_u$. **b** Two substrate cleavage factor α_{enz} values are tested. α_{enz}=1 is typical of the action of endocleaving enzymes, which randomly disrupt any substrate bond and generate oligomers, inducing a rapid shift towards the $\mathcal{D}_u$ microbial uptake domain. α_{enz}=5 is typical of exocleaving enzymes attacking the substrate end-members to release small oligomers, inducing a slower shift towards $\mathcal{D}_u$. Reprint from Sainte-Marie et al. (2021).

2.4 Notations

Notations with description and units, $*$ stands for biochemical classes, *mic* stands for microbes and *enz* stands for enzymes.

Time variable		
t		day
Polymerization description		
p_*	polymerization variable for biochemical class $*$	a.u.
p_*^{min}	minimum value of polymerization variable p_*	p_*
p_*^{max}	maximum value of polymerization variable p_*	p_*
Substrate pools		
C_{sub}	total carbon of non-living substrate	g_C
C_{mic}	carbon of living microbe mic	g_C
C_*	total carbon of biochemical class $*$	g_C
C_*^{ac}	accessible carbon of biochemical class $*$	g_C
C_*^{in}	inaccessible carbon of biochemical class $*$	g_C
χ_*^{ac}	distribution of accessible carbon of biochemical class $*$	$g_C.p_*^{-1}$
χ_*^{in}	distribution of inaccessible carbon of biochemical class $*$	$g_C.p_*^{-1}$
Microbes signature		
$S_{mic,*}$	proportion of the dead microbial compounds returning to the pool $*$	-
$s_{mic,*}$	distribution of the dead microbial compounds returning to the pool $*$	p_*^{-1}
Carbon inputs		
I	total carbon input flux	$g_C.t^{-1}$
i_*^{ac}	distribution of accessible C input fluxes for biochemical class $*$	$g_C.p_*^{-1}.t^{-1}$
i_*^{in}	distribution of inaccessible C input fluxes for biochemical class $*$	$g_C.p_*^{-1}.t^{-1}$
Microbes turnover		
$\mathcal{D}_{mic,*}^u$	accessibility interval (or domain) for biochemical class $*$	-
$F_{mic,*}^{upt}$	specific microbe uptake flux for biochemical class $*$	$g_C.p_*^{-1}.t^{-1}$
$e_{mic,*}$	carbon use efficiency for biochemical class $*$	-
m_{mic}	mortality rate of microbe	$g_C.t^{-1}$
$F_{mic,*}^{nec}$	specific mortality flux for biochemical class $*$	$g_C.p_*^{-1}.t^{-1}$
Enzymatic activity		
F_{enz}^{act}	enzymatic activity flux	$g_C.p_*^{-1}.t^{-1}$
$\mathcal{D}_{enz}$	accessibility interval (or domain) for biochemical class $*$	-
τ_{enz}	depolymerization rate	t^{-1}
π_{enz}^{mic}	specific depolymerization rates	t^{-1}
$\mathcal{K}_{enz}$	dispersion kernel	p_*^{-1}
Changes in local physicochemical conditions		
$\tau_{tr,*}^{ac}$	rate of local condition change toward accessibility	t^{-1}
$\tau_{tr,*}^{in}$	rate of local condition change toward inaccessibility	t^{-1}

Bibliography

Sainte-Marie, J., Barrandon, M., Saint-André, L., Gelhaye, E., Martin, F., and Derrien, D. (2021). C-stability an innovative modeling framework to leverage the continuous representation of organic matter. *Nature communications*, 12(1):1–13.

Chapter 3

Model implementation

Contents

3.1 Model implementation structure

C-STABILITY simulations are organised with a formalism close to the one proposed by Bayol (2016). Let be $N \in \mathbb{N}^*$ the number of model iterations. Then, the entire simulation is obtained by computing the system $\forall n \in [\![1, N]\!]$,

$$\begin{cases} s_0 \text{ given} & \\ s_n = F_{n-1}(p, c_{n-1}, s_{n-1}) & \forall n \in [\![1, N]\!] \\ o_n = G_n(p, s_n) & \forall n \in [\![1, N]\!] \end{cases} \tag{3.1}$$

where

- s_n is the state of the model which contains the model variables, in particular s_0 is the initial state of the system,
- F_n is the simulation function implementing the dynamical equations of C-STABILTY,
- p are the simulation parameters (e.g. biochemical classes polymerization, microbes and enzymes traits...),
- c_n is the simulation context (e.g. climate and soil variables over the time, substrate input),
- o_n is an observation of the state s_n,
- G_n is an observation function which extract specific values of the state s_n.

3.2 Numerical schemes

3.2.1 Discretization

Each unit of time is divided in n_t sub-intervals. Then, $\Delta_t = \frac{1}{n_t}$ and the number of iterations computed by the model between 0 and T units of time is $n_t(T+1)$. For each biochemical class $*$, the interval of polymerization $[p^*_{min}, p^*_{max}]$ is regularly divided in n^*_p sub-intervals $[p^*_i, p^*_{i+1}]$. Then we define $n^*_p + 1$ points such that for all $i \in [\![0, n^*_p]\!]$, $p^*_i = p^*_{min} + i\Delta^*_p$ where $\Delta^*_p = \frac{p^*_{max} - p^*_{min}}{n^*_p}$.

3.2.2 Microbe evolution

For each microbe, the time evolution of its constitutive C is computed with an explicit Euler scheme,

$$C_{mic}(t + \Delta_t) \simeq C_{mic}(t) + \Delta_t \frac{dC_{mic}}{dt}(t). \tag{3.2}$$

3.2.3 Substrate evolution

For each biochemical class $*$, substrate is evaluated on the grid of the interval of polymerization $[p^*_{min}, p^*_{max}]$. By using eqs. (2.21) and (2.22) and the Euler method for time evolution, we have for all $i \in [\![0, n^*_p]\!]$,

$$\chi^{ac}_*(p^*_i, t + \Delta_t) \simeq \chi^{ac}_*(p^*_i, t) + \Delta_t \frac{\partial \chi^{ac}_*}{\partial t}(p^*_i, t), \tag{3.3}$$

$$\chi^{in}_*(p^*_i, t + \Delta_t) \simeq \chi^{in}_*(p^*_i, t) + \Delta_t \frac{\partial \chi^{in}_*}{\partial t}(p^*_i, t). \tag{3.4}$$

In the following, each element of the terms $\frac{\partial \chi^{ac}_*}{\partial t}(p^*_i, t)$ and $\frac{\partial \chi^{in}_*}{\partial t}(p^*_i, t)$ is considered.

$$\begin{aligned}
\frac{\partial \chi_*^{ac}}{\partial t}(p_i^*,t) &= \tau_{tr}^{ac}\chi_*^{in}(p_i^*,t) - \tau_{tr}^{in}\chi_*^{ac}(p_i^*,t) + i_*^{ac}(p_i^*,t) \\
&\quad + \sum_{\text{enz}} \left(-\tau_{enz}(p_i^*,t)\chi_*^{ac}(p_i^*,t) + \int_{\mathcal{D}_{enz}} \mathcal{K}_{enz}(p_i^*,p_*')\tau_{enz}(p_*',t)\chi_*^{ac}(p_*',t)dp_*' \right) \\
&\quad + \sum_{\text{mic}} \left(m_{mic}\left(t, C_{mic}\right) s_{mic,*}(p_i^*) - F_{mic,*}^{upt}(\chi_*^{ac}, p_i^*, t) \right), \\
\frac{\partial \chi_*^{in}}{\partial t}(p_i^*,t) &= \tau_{tr}^{in}\chi_*^{ac}(p_i^*,t) - \tau_{tr}^{ac}\chi_*^{in}(p_i^*,t) + i_*^{in}(p_i^*,t).
\end{aligned}$$

Here $\mathcal{D}_{enz} = [p_{min}^*, p_{max}^*]$. We propose here two methods to compute the integral,

$$\int_{p_{min}^*}^{p_{max}^*} \mathcal{K}_{enz}(p_i^*,p_*')\tau_{enz}(p_*',t)\chi_*^{ac}(p_*',t)dp_*$$

according to polymerization discretization. Other terms do not require specific methods. To simplify the following, we note $\phi = \tau * \chi$ and we consider the computation of the integral, $\int_{p_{min}^*}^{p_{max}^*} \mathcal{K}(p,p')\phi(p')dp'$ and we note $\Phi = \left(\phi(p_0^*), \phi(p_1^*), \ldots, \phi(p_{n_p^*}^*)\right)^T$.

3.2.3.1 Regular approach

The integral is approximated according to the selected method for distribution integration in simulation parameters (i.e. rectangle left, rectangle right, trapeze). Here we only present the result with the trapeze method. For all $p \in [p_{min}^*, p_{max}^*]$,

$$\begin{aligned}
\int_{p_{min}^*}^{p_{max}^*} \mathcal{K}(p,p')\phi(p')dp' &= \sum_{j=0}^{n_p^*-1} \int_{p_j^*}^{p_{j+1}^*} \mathcal{K}(p,p')\phi(p')dp', \\
&\simeq \sum_{j=0}^{n_p^*-1} \frac{\Delta_p^*}{2}\left(\mathcal{K}(p,p_j^*)\phi(p_j^*) + \mathcal{K}(p,p_{j+1}^*)\phi(p_{j+1}^*)\right), \\
&\simeq \Delta_p^* \frac{\mathcal{K}(p,p_0^*)}{2}\phi(p_0^*) + \Delta_p^* \sum_{j=1}^{n_p^*-1} \mathcal{K}(p,p_j^*)\phi(p_j^*) + \Delta_p^* \frac{\mathcal{K}(p,p_{n_p^*}^*)}{2}\phi(p_{n_p^*}^*).
\end{aligned}$$

Finally, for all $i \in [\![0, n_p^*]\!]$,

$$\int_{p_{min}}^{p_{max}} \mathcal{K}(p_i^*,p')\phi(p')dp' \simeq \mathbb{K}_i\Phi, \tag{3.5}$$

where $\mathbb{K}_i = \left(\Delta_p^* \frac{\mathcal{K}(p_i^*,p_0^*)}{2}, \Delta_p^*\mathcal{K}(p_i^*,p_1^*), \ldots, \Delta_p^*\mathcal{K}(p_i^*,p_{n_p^*-1}^*), \Delta_p^* \frac{\mathcal{K}(p_i^*,p_{n_p^*}^*)}{2}\right)$.

3.2.3.2 Integral approach

We assume that the analytical expression of $\mathcal{L}(p,p') = \int \mathcal{K}(p,p')dp'$ is known. For all $j \in [\![0, n_p^* - 1]\!]$, we note $\mathcal{L}_j(p) = \int_{p_j^*}^{p_{j+1}^*} \mathcal{K}(p,p')dp' = \mathcal{L}(p,p_{j+1}^*) - \mathcal{L}(p,p_j^*)$. Then, for all $p \in [p_{min}^*, p_{max}^*]$,

$$\begin{aligned}
\int_{p_{min}^*}^{p_{max}^*} \mathcal{K}(p,p')\phi(p')dp' &= \sum_{j=0}^{n_p^*-1} \int_{p_j^*}^{p_{j+1}^*} \mathcal{K}(p,p')\phi(p')dp', \\
&\simeq \sum_{j=0}^{n_p^*-1} \frac{\phi(p_j^*) + \phi(p_{j+1}^*)}{2} \int_{p_j^*}^{p_{j+1}^*} \mathcal{K}(p,p')dp', \\
&\simeq \sum_{j=0}^{n_p^*-1} \frac{\phi(p_j^*) + \phi(p_{j+1}^*)}{2} \mathcal{L}_j(p), \\
&\simeq \frac{\mathcal{L}_0(p)}{2}\phi(p_0^*) + \sum_{j=1}^{n_p^*-1} \frac{\mathcal{L}_{j-1}(p) + \mathcal{L}_j(p)}{2}\phi(p_j^*) + \frac{\mathcal{L}_{n_p^*-1}(p)}{2}\phi(p_{n_p^*}).
\end{aligned}$$

Finally, for all $i \in [\![0, n_p^*]\!]$,

$$\int_{p_{min}}^{p_{max}} \mathcal{K}(p_i^*,p')\phi(p')dp' \simeq \mathbb{L}_i\Phi, \tag{3.6}$$

where $\mathbb{L}_i = \left(\frac{\mathcal{L}_0(p_i^*)}{2}, \frac{\mathcal{L}_0(p_i^*)+\mathcal{L}_1(p_i^*)}{2}, \ldots, \frac{\mathcal{L}_{n_p^*-2}(p_i^*)+\mathcal{L}_{n_p^*-1}(p_i^*)}{2}, \frac{\mathcal{L}_{n_p^*-1}(p_i^*)}{2} \right)$.

Bibliography

Bayol, B. (2016). *Système informatique d'aide à la modélisation mathématique basé sur un langage de programmation dédié pour les systèmes dynamiques discrets stochastiques. Application aux modèles de croissance de plantes.* PhD thesis, Université Paris-Saclay (ComUE).

Appendices

Appendix A

Licence LGPL v.3

GNU LESSER GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

0. Additional Definitions.

As used herein, "this License" refers to version 3 of the GNU Lesser General Public License, and the "GNU GPL" refers to version 3 of the GNU General Public License.

"The Library" refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

An "Application" is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

A "Combined Work" is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the "Linked Version".

The "Minimal Corresponding Source" for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The "Corresponding Application Code" for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work.

1. Exception to Section 3 of the GNU GPL.

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL.

2. Conveying Modified Versions.

If you modify a copy of the Library, and, in your modifications, a

facility refers to a function or data to be supplied by an Application
that uses the facility (other than as an argument passed when the
facility is invoked), then you may convey a copy of the modified
version:

   a) under this License, provided that you make a good faith effort to
   ensure that, in the event an Application does not supply the
   function or data, the facility still operates, and performs
   whatever part of its purpose remains meaningful, or

   b) under the GNU GPL, with none of the additional permissions of
   this License applicable to that copy.

  3. Object Code Incorporating Material from Library Header Files.

  The object code form of an Application may incorporate material from
a header file that is part of the Library. You may convey such object
code under terms of your choice, provided that, if the incorporated
material is not limited to numerical parameters, data structure
layouts and accessors, or small macros, inline functions and templates
(ten or fewer lines in length), you do both of the following:

   a) Give prominent notice with each copy of the object code that the
   Library is used in it and that the Library and its use are
   covered by this License.

   b) Accompany the object code with a copy of the GNU GPL and this license
   document.

  4. Combined Works.

  You may convey a Combined Work under terms of your choice that,
taken together, effectively do not restrict modification of the
portions of the Library contained in the Combined Work and reverse
engineering for debugging such modifications, if you also do each of
the following:

   a) Give prominent notice with each copy of the Combined Work that
   the Library is used in it and that the Library and its use are
   covered by this License.

   b) Accompany the Combined Work with a copy of the GNU GPL and this license
   document.

   c) For a Combined Work that displays copyright notices during
   execution, include the copyright notice for the Library among
   these notices, as well as a reference directing the user to the
   copies of the GNU GPL and this license document.

   d) Do one of the following:

0) Convey the Minimal Corresponding Source under the terms of this
License, and the Corresponding Application Code in a form
suitable for, and under terms that permit, the user to
recombine or relink the Application with a modified version of
the Linked Version to produce a modified Combined Work, in the
manner specified by section 6 of the GNU GPL for conveying
Corresponding Source.

1) Use a suitable shared library mechanism for linking with the
Library. A suitable mechanism is one that (a) uses at run time
a copy of the Library already present on the user's computer
system, and (b) will operate properly with a modified version
of the Library that is interface-compatible with the Linked
Version.

e) Provide Installation Information, but only if you would otherwise
be required to provide such information under section 6 of the
GNU GPL, and only to the extent that such information is
necessary to install and execute a modified version of the
Combined Work produced by recombining or relinking the
Application with a modified version of the Linked Version. (If
you use option 4d0, the Installation Information must accompany
the Minimal Corresponding Source and Corresponding Application
Code. If you use option 4d1, you must provide the Installation
Information in the manner specified by section 6 of the GNU GPL
for conveying Corresponding Source.)

5. Combined Libraries.

You may place library facilities that are a work based on the
Library side by side in a single library together with other library
facilities that are not Applications and are not covered by this
License, and convey such a combined library under terms of your
choice, if you do both of the following:

a) Accompany the combined library with a copy of the same work based
on the Library, uncombined with any other library facilities,
conveyed under the terms of this License.

b) Give prominent notice with the combined library that part of it
is a work based on the Library, and explaining where to find the
accompanying uncombined form of the same work.

6. Revised Versions of the GNU Lesser General Public License.

The Free Software Foundation may publish revised and/or new versions
of the GNU Lesser General Public License from time to time. Such new
versions will be similar in spirit to the present version, but may
differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy's public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.