Comparing and Updating R Packages using MCMC Algorithms for Linear Inverse Modeling of Metabolic Networks

Valerie Girardin, Théo Grente, Nathalie Niquil, Philippe Regnault

Comparing and Updating R packages using MCMC Algorithms for Linear Inverse Modeling of Metabolic Networks

Valérie Girardin[1*†], Théo Grente[1,2†], Nathalie Niquil[3†], Philippe Regnault[4†]

[1*]Laboratoire de Mathématiques Nicolas Oresme, Université de Caen Normandie, CNRS-6139, BP5186, Caen, 14032, France.
[2]France Énergies Marines, Avenue Alexis de Rochon, Plouzané, 29280, France.
[3]Laboratoire Biologie des Organismes et Ecosystemes Aquatiques, Université de Caen Normandie, CNRS-8067, Esplanade de la Paix, Caen, 14000, France.
[4]Laboratoire de Mathématiques de Reims, Université de Reims Champagne-Ardenne, CNRS-9008, BP1039, Reims, 51687, France.

*Corresponding author(s). E-mail(s): valerie.girardin@unicaen.fr;
Contributing authors: theo.grente@unicaen.fr;
nathalie.niquil@unicaen.fr; philippe.regnault@univ-reims.fr;
†These authors contributed equally to this work.

Abstract

Gathered under the name of metabolic networks, trophic, biochemical, and urban networks are here handled as a single field. In the Linear Inverse Modeling framework, these highly complex objects of research are all mathematically represented by weighted oriented graphs whose vertices are compartments and edges the flows (or flux) of matter or energy.

All the flows that satisfy realistic constraints belong to very anisotropic high dimensional polytopes that cannot be analytically determined. Sampling the polytope yields a set of possible scenarios for the metabolic network. Different Monte Carlo Markov Chain (MCMC) algorithms together with their most recent implementations are scrutinized, leading to design an updated R package called {`samplelim`}. Comparison of the most recent implementations in terms of both computation time and sampling performances follows a methodology involving

acknowledged and new statistical diagnostics and indexes. Application on real data metabolic networks of the three types shows that {samplelim} gathers the best properties of previous implementations of the MCMC algorithms.
Code Repositories: The source code of the package {samplelim} is publicly available from its GitHub repository.*
The code to reproduce the computations in the manuscript has its own private GitHub repository.† In case of publication, it will be made public and a DOI will be attached to it.

Keywords: Metabolic, trophic, biochemical and urban networks; Linear inverse modeling; High-dimensional polytope; Mirror and Billard walks; R package.

MSC Classification: 92-04 , 92-08 , 92-10 , 65C05 , 60J20

1 Introduction

According to Encyclopedia Britannica, metabolism is "the sum of the chemical reactions that take place within each cell of a living organism and that provide energy for vital processes and for synthesizing new organic material". However, metabolic systems cover many different concepts used in different domains.

The three main functions of metabolism in biology are: the conversion of energy in food into energy available to run cellular processes; the conversion of food into building blocks for proteins, lipids, nucleic acids, and some carbohydrates; the elimination of metabolic wastes. The development of molecular biology during the 1990's led to the characterization of first partial and then complete genomes in microbes, that opened a new area of research on microbial metabolic modeling; see [7]. The linear constraints on the flows within microbial cells are physico-chemical constraints –e.g. diffusion rates, topo-biological constraints –e.g. derived from the crowding of molecules inside cells, environmental constraints –e.g. nutrient availability, and regulatory constraints –e.g. enzyme regulation; see [30]. This type of model spread out of microbial biology with the development of overall sequencing of metazoan species and in particular of humans; see [8]. The networks of metabolic flows within cells are called either metabolic networks in [1], metabolic reaction networks in [7] or biochemical networks in [28]; to avoid any confusion, we will here refer to them as the latter.

The Encyclopedia of Earth Science enlarges the definition: "Metabolism refers to the use of energy for the production and assimilation of food and for locomotion, maintenance, growth, reproduction, and other processes that characterize life". With this definition, the concept extends from biology to ecology, and allows the definition not only of cell or organism metabolism, but also of ecosystem metabolism. In ecosystems, the main metabolic processes are production and respiration. Because most of the production is ingested by consumers, the metabolic system can be represented as a network of flows from preys to predators, including waste production and respiration, that defines the food web.

*{samplelim} source code: https://github.com/pregnault/samplelim
†Manuscript code: https://github.com/TheoGrente/limcomp

The definition of metabolic networks also extends to economy, giving rise to the concept of urban metabolism, for studying material and energy exchanges in cities, with the metaphor of megapolis seen as bodies. The first discussions on urban metabolism go a long way back, to Karl Marx in 1881; see [38]. The concept came back in economic studies in the 1960's in the context of the growing pollution of air and water in USA cities. Since then, the literature on the subject developed and in 2008, a devoted congress highlighted urban metabolism as having a worldwide influence; see [18]. The study of urban metabolism facing strong environmental issues is particularly relevant in China, in a context of rapidly growing megapolis; see [38] and the references therein. A trend of studies merged economical approaches with concepts and methods derived from ecology; see, e.g. [5]. Indeed, networks combining energy and matter flow circulation present common features with trophic networks. The other way round, ecological network analysis was strongly influenced by econometry with the application of Leontief input-output analysis, first in [15]. These intertwined approaches of network analysis led in particular to the recent project UNCNET; see [14, 20].

In the three above types of metabolic networks, even if the goals may appear as different, flows can be represented by interactions, and the issuing mathematical models for the networks are graphs whose vertices are the compartments and edges the interactions. In order to specify the unknown flow values, a class of methods involving Linear Inverse Modeling (LIM) was developed; see [26] and the references therein. It relies on the principle of steady state mass balance, i.e., the sum of the inflows and outflows through the components of the system equals the rate of change in their standing stocks, most often considered negligible; the nature of the "mass" depends on the type of system. This yields a set of linear equations known as the mass balance equations (MBE). Through acquired knowledge, from dynamics field measurements, laboratory experiments, bibliographical knowledge, or simulation, equality constraints are added to the modeling. Moreover, bounds are imposed on the flows (biochemical networks) or linear combinations of flows are imposed to remain between bounds (urban and trophic). The set of these linear equations and inequations defines a convex bounded multidimensional polyhedron – a polytope, within which lie all realistic solutions to the problem.

Quite recently, the so-called sampling methods were developed to describe this kind of polytopes by calculating a representative sample of solutions through the Monte Carlo approach, and more efficiently the Monte Carlo Markov Chain approach (MCMC); see [23, 25], and [36]. Thus, LIM-MCMC methods involve mass balanced models that consider the uncertainty in the data and link them pertinently to the variability of the living; see [12, 26, 36]. In all MCMC methods, an ergodic time-reversible Markov chain is designed, with the desired asymptotic distribution, leading to a uniform sampling of the polytope. Further, two reflective MCMC algorithms particularly suited to sampling highly complex LIM polytopes, the Mirror Walk (MiW) in [36] and of the Billard Walk (BiW) in [29], were developed independently. The former is designed in an R package called {`limsolve`} –including the function `xsample()`–

especially intended for trophic modeling. The latter is implemented in [3] for sampling polytopes in order to estimate their volume, with an application to biochemical networks in [4]. Both include parameters, called jump lengths, to be fitted.

Recently, both [19] and [9] have undertaken rigorous comparisons of several sampling algorithms for biochemical networks, and concluded that the Coordinate Hit-and-Run with Rounding (CHRR) algorithm is the most efficient in terms of both computation time and multiple convergence diagnostics. Unfortunately, its only available implementation, the function `chrrExpSampler()` of the {`COBRA`} Matlab toolbox, works only for bounds on the flows, and not for more intricate inequality constraints, so cannot be applied to urban or trophic network models.

Regarding `xsample`(), [9] was only able to successfully apply it to the *Escherichia coli* network, out of ten considered genome scale systems, and hence concluded that an efficient reflective MCMC algorithm implementation allowing for the framework of [36] to be applied at a large scale was lacking. One of the main goals of the present paper is to fill this gap. In this aim, we will describe and compare sampling methods, theoretically and with respect to computation time and statistical diagnostics of convergence. This will lead us to develop an updated version of the R package {`limsolve`} and function `xsample()`, called {`samplelim`} and `rlim()`, that uses the framework of the R package {`volesti`} designed in [3].

The paper is organized as follows. Section 2 is devoted to the construction of the polytopes of solutions to metabolic networks constraints, whichever be their type, with examples of real data trophic, urban and biochemical networks. Section 3 details different MCMC methods for sampling polytopes, from the classical Hit and Run and its variants, to the most recent reflective Hamiltonian MCMC methods, together with available implementations. The updated implementation {`samplelim`} is proposed in Section 3.4. For similar computation times, sampling performances of the functions of the R packages {`samplelim`}, {`limsolve`}, {`volesti`}, plus `chrrExpSampler()`, are thoroughly compared in Section 4, on the real data examples of Section 2. The methodology inspired by both [9] and [19], that also involves a new numeric index that we call range coverage, is presented in Section 4.1. In particular, Section 4.2 is devoted to show that {`samplelim`} is much faster than {`limsolve`}, while retaining its specific desirable features, which makes it a very competitive MCMC package for sampling LIM polytopes, especially for trophic networks.

2 From the constraints to the polytope of solutions

As highlighted above, from a mathematical point of view, a metabolic network is a valued oriented graph. After its theoretical construction from the physical or biological constraints, we will introduce realistic models for real data of the three different types of metabolic networks.

Let us call the graph (V, E, f). Its vertices $i \in V$ are the compartments, $E \subseteq V \times V$ is the set of oriented edges with $n = |E|$, and $f = (f_{ij})_{ij \in E}$ is the vector of flow values associated to edges. Precisely, $ij = (i, j) \in E$ denotes the edge from compartment i to compartment j and f_{ij} is the amounts of matter or energy transiting from i to j, the so-called flow –or flux. In many situations, all masses B_i of the compartments can

be measured with accuracy, while the flows f_{ij} are much more difficult to evaluate. Therefore, we will adopt here the standard viewpoint that masses are given and flows are unknown.

The vector of flows $f = (f_{ij})_{ij\in E}$, has non negative components satisfying Kirchoff law on all compartments, yielding the MBE

$$\sum_{j\in N^+(i)} f_{ij} - \sum_{j\in N^-(i)} f_{ji} = 0, \quad i \in V,$$

where the successors and predecessors of compartment i are $N^+(i) = \{j \in V : ij \in E\}$ and $N^-(i) = \{j \in V : ji \in E\}$. Acquired knowledge on the network, coming from field measurements, laboratory experiments or any other source, may yield additional linear equations constraints on flows. All of them can be aggregated into

$$Af = b, \tag{1}$$

where A is an $m \times n$ matrix and b an m-dimensional vector, with m the number of linear equations. Note that we are only interested in the under-determined problem, that is $m < n$: the knowledge on flows is not exhaustive, the solution is not unique.

Note that for all quantities to be well-defined, constraints for all the flows to be positive have to be added, say, $f_{ij} \geq \varepsilon_{ij}$, for all $i \in E$, where the size of ε_{ij} is to be fitted to the range of f_{ij}. Other constraints imposed by acquired knowledge take the form of bounds on linear combinations of the flows. In mathematical words, this sums up to

$$Gf \geq h, \tag{2}$$

where G is a $k \times n$ matrix and h a k-dimensional vector, with k the number of such inequations. Apart from the MBE involving several flows in each equation, most constraints in trophic and urban networks involve only a small number of flows, and matrices A and G are mainly composed of 0. This is all the more true in the biochemical case where the inequality constraints are only bounds on the flows, so that G is of dimension $2n \times n$, with only one 1 per row corresponding to a lower bound and one -1 to an upper bound.

The set

$$\mathcal{S} = \{f = (f_{ij})_{ij\in E} \in \mathbb{R}^n : \ f_{ij} \geq \varepsilon_{ij}, \ Af = b, \ Gf \geq h\}, \tag{3}$$

of flows satisfying both constraints (1) and (2) is the intersection of hyperplanes and half-spaces so is a convex polyhedron. In addition, (1) and (2) always induce bounds on the flows, say

$$m_{ij} \leq f_{ij} \leq M_{ij}, \quad ij \in E, \tag{4}$$

where $R_{ij} = M_{ij} - m_{ij}$ is called the range of the flow f_{ij}. Therefore $\mathcal{S}$ is a bounded polyhedron – called a polytope.

The polytope $\mathcal{S}$ is a subset of $\mathbb{R}^n$. Still, its dimension is $n - \rho$, where ρ is the rank of A. In other words, $\mathcal{S} = f^* + \widetilde{A}\widetilde{\mathcal{S}}$, where $f^* \in \mathcal{S}$ is any admissible flow, $\widetilde{A}$ is the

$n \times (n - \rho)$- matrix whose columns form a basis of the kernel of A, and

$$\widetilde{\mathcal{S}} = \{\widetilde{f} \in \mathbb{R}^{n-\rho} : \widetilde{G}\widetilde{f} \geq \widetilde{h}\}, \tag{5}$$

is the so-called reduced polytope, where $\widetilde{G} = G\widetilde{A}$ and $\widetilde{h} = Gf^* - h$. Thus, up to an affine transformation, the polytope $\mathcal{S}$ with empty interior in $\mathbb{R}^n$ is reduced to a full dimensional polytope $\widetilde{\mathcal{S}}$ in $\mathbb{R}^{n-\rho}$. This proves to be a corner stone of most sampling methods.

We will conduct the comparison of packages on three different networks: a trophic network in the English Channel and its aggregated version, an urban network for nitrogen exchanges in Vienna and a biochemical network for *Escherichia coli*.

Example 1. A trophic network of a marine ecosystem at Courseulles-sur-mer, France, in the English Channel, with 19 compartments and 144 flows from [27].

The ranges of the flows go from $O(10^{-5})$ to $O(10^2)$. The dimension of A is 25×144, corresponding to the 19 MBE plus 6 equations, each involving only one flow –so the values of these 6 flows are actually known. The dimension of G is 361×144. Here 144 rows of this matrix ensure positivity –each consists in only 0 but for one 1, while the other rows involve more than one flow and contain coefficients different from 1 or -1. Once reduced, the polytope is of dimension 119 and its ranges goes from $O(10^{-2})$ to $O(10^2)$.

Example 2. The trophic network of Example 1 aggregated according to a realistic ecological point of view into a network of 6 compartments and 28 flows; see [2], where all network constraints are detailed.

The ranges of the flows go from $O(10^0)$ to $O(10^2)$. The dimension of A is 6×28, corresponding to 6 MBE, one for each compartment. The dimension of G is 72×28, of which 28 rows simply ensure positivity of the flows. The 44 other rows correspond to more complex constraints each involving more than one flow. Once reduced, the polytope is of dimension 22 with ranges going from $O(10^0)$ to $O(10^2)$.

Example 3. An urban model of nitrogen exchanges in Vienna megapolis with 13 compartments and 69 flows; see [14] and [20].

In[14], a LIM approach is considered to take into account the uncertainty on the flows measurements. In this network, the ranges of the flows go from $O(10^2)$ to $O(10^8)$, an artificial upper bound. The dimension of A is 16×69 with 13 MBE and 3 other equations, each involving several flows. The dimension of G is 302×69. Indeed, 69 inequations ensure that the flows are bounded by 10^8, and 69 ensure that they are positive –note that 138 actually appear because of duplicates due to the way the model is declared. Then 91 inequations give more precise bounds on single flows, and 2 involve several flows. The reduced polytope is of dimension 53 with the ranges going from $O(10^6)$ to $O(10^9)$.

Example 4. The core *Escherichia coli* metabolic model with 72 compartments (metabolites) and 95 flows; see [28]. The biochemical reactions are based on biochemical, genetic and genomic information, such as exact reaction stoichiometry, reaction reversibility, or relationships between genes and proteins.

The ranges of the flows go from $O(10^0)$ to $O(10^3)$, an artificial upper bound. The dimension of A is 72×95 with 72 MBE. All rows of G, of dimension 190×95, count

only one non null coefficient, corresponding to bounds on the flows: 95 rows contain one -1 for upper bounds, while the other 95 contain one 1 for lower bounds. The rank of the matrix A is 71, once equations hidden in inequations are added to the constraints. These equations come from 8 flows with ranges 0. The reduced polytope is of dimension 24 with all ranges of order $O(10^2)$.

3 MCMC for sampling polytopes

Despite the simplicity of its implicit definition in (3), the polytope of admissible flows is a complex, high dimensional, geometric object. Hence, its analytical description is an unreachable goal. A commonly used alternative consists in exhibiting sets of points taken at random in the polytope, obtained through the celebrated MCMC methods.

This section aims at presenting MCMC algorithms, together with some of their implementations, that are of common use in the context of metabolic network studies. An introductory discussion is proposed in Section 3.1 on the main points of attention to take care of when using MCMC methods. Then we focus on the Hit and Run algorithm and some of its variants in Section 3.2, and on two reflective Hamiltonian MCMC algorithms, the Mirror walk (MiW) and the Billard Walk (BiW), in Section 3.3. Section 3.4 will provide a unifying overview of three of these algorithms together with their most recent available implementations, the R packages {`limsolve`} and {`volesti`} and the Matlab toolbox {`COBRA`}, leading to the main innovation of this paper, an updated implementation of MiW and BiW, that we call {`samplelim`}. Its interface is in the R Statistical Software [31], but computation is handled in the C++ programming language as in {`volesti`}, which significantly reduces computation time.

Note that all MCMC methods discussed here generate points inside a set with a non-empty interior. Hence, below, the "polytope" is, unless explicitly stated, the "reduced polytope" $\widetilde{\mathcal{S}}$ in (5) and not the original one of (3). For the sake of simplicity, it will still be denoted by $\mathcal{S}$, with dimension n (instead of $n - \rho$).

3.1 MCMC uniform sampling of polytopes

First, basic Monte Carlo approaches have been considered in the ecological literature; see [22, 25]. Unfortunately, they fail to generate efficiently uniform samples in high dimensional polytopes. Indeed, the ratio between the volume of $\mathcal{S}$ and the volume of the smallest hyper-rectangle $\prod_{ij\in E}[m_{ij}, M_{ij}]$ that contains $\mathcal{S}$ generally decreases exponentially fast with the number of dimensions, making the classical rejection method inefficient.

As an alternative, numerous MCMC methods have been introduced for generating samples drawn uniformly into a polytope. In all of them, an ergodic time-reversible Markov chain is designed, with the desired asymptotic distribution –here, the uniform distribution over the polytope. Thus a set of N_0 candidate points in the polytope is obtained by simulating the N_0 first realizations –draws– of the chain, called the pilot sample.

Such sets of points have two characteristics that are to be properly taken into account in order to get adequate results. Obviously, the first coordinates of the Markov chain generally do not follow its asymptotic distribution, so the first draws have to

be discarded. The number M of such draws, called the burn-in period, intuitively represents the time to wait before convergence is quite achieved. Also, the successive coordinates of a Markov chain are correlated, so a certain number of draws have to be discarded after each drawn point before adding a next point to the returned sample. This number T is called the thinning parameter. Thus, the actual number of retained points from N_0 iterations of an MCMC algorithm is

$$N_1 \simeq (N_0 - M)/T. \tag{6}$$

In other words, from any pilot sample

$$f^{(1:N_0)} = (f^{(1)}, \ldots, f^{(N_0)}), \tag{7}$$

with elements for each flow $f_{ij}^{(1:N_0)} = (f_{ij}^{(1)}, \ldots, f_{ij}^{(N_0)})$, we obtain the actual burn-in free and thinned N_1-sample in the polytope

$$(f^{(M+1)}, f^{(M+T+1)}, \ldots, f^{(M+(N_1-1)T+1)}). \tag{8}$$

All MCMC methods require a starting point for the chain. This first point is generally obtained by using a numerical solver. For instance, {`limsolve`} uses the LSEI (Least Squares with Equalities and Inequalities) algorithm of [17] to find a particular solution, whereas {`volesti`} computes the Chebyshev ball of the polytope and uses its center as the starting point.

3.2 Hit and Run and its variants

The symmetric mixing algorithm nowadays known as the Hit and Run algorithm (HR) was introduced by [33]. The basic principle is to randomly choose both a direction to move forward (or backward) from a point of the polytope and a segment length to generate another one. The induced Markov chain formed by the iteration of this process is proven to converge to the uniform distribution in the polytope.

Precisely, in the HR procedure, from an initial point $f^{(1)}$, points $f^{(2)}$, $\ldots$, $f^{(N)}$, are iteratively built by repeating the same successive actions, of which Algorithm 1 presents the pseudo-code:

1. Choose a random direction d_i in the n dimensional space.
2. Determine the two intersection points $I^{(i)-}$ and $I^{(i)+}$ between the line passing through $f^{(i-1)}$ and directed by d_i, and the borders of the polytope.
3. Sample uniformly a point in the segment $[I^{(i)-}, I^{(i)+}]$.

Generating a random direction at Step 1 means simulating a realization of the uniform distribution on the unit sphere of $\mathbb{R}^n$. The main point here is that the distribution of the direction has to be symmetric (i.e., directions d and $-d$ have the same probability), so that the transition kernel of the underlying Markov chain is symmetric.

When the polytope is very anisotropic, HR needs many iterations to achieve a uniform distribution of points. Indeed the convergence time or mixing time is of the order $O(n^2 R_{\max}^2/R_{\min}^2)$, where $R_{\max}/R_{\min}$, called the sandwiching ratio of the body,

is the largest range of flows over the smallest. Examples 1 to 4 show that this ratio may be quite huge.

Sampling directions utterly randomly leads to going into thin directions, thus inducing a long time to sample the large ones. Indeed, the algorithm spends a lot of time bouncing on closed borders of the polytope. This is a strong drawback of the method, somewhat addressed by variants.

The CHR, also called Gibbs Sampler, proposed in [34], is similar to HR, except for Step 1 to be replaced by:

1. Choose uniformly a random direction along one of the n dimensions.

The overhead per step for CHR with respect to HR is reduced from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$, since the update follows only coordinate directions. On the contrary, this leads to extremely correlated samples and induces a high thinning parameter.

Introduced in [16], CHRR begins by a pre-processing step of rounding of the polytope followed by a CHR exploration of the rounded polytope, before mapping samples back to the original polytope. The rounding step consists in computing the maximum volume ellipsoid inscribed in the polytope and applying the transformation that maps this ellipsoid to the unit ball. Note that a remarkable property of CHRR is its guaranteed distributional convergence to the uniform; see [24] for the mixing time.

More recent versions of HR are the artificially centered hit-and-run (ACHR), and its improved version optimized general parallel (OPTGP); see [19]. Still, CHRR is found to perform the best for sampling LIM polytopes; see [9] and [19] and the references therein.

3.3 Reflective MCMC Algorithms

A well-documented pitfall of HR-like algorithms is the important auto-correlation of the samples; see [29] and the reference therein. This is particularly true for highly anisotropic LIM polytopes, with narrow angles inducing regions in which HR algorithms get stuck.

Reflective Hamiltonian MCMC algorithms rely on a reflection mechanism on the boundaries in order to avoid these traps and thus reduce auto-correlation; see [6] and the references therein. We will focus on two of them, the MiW of [36] in ecology, and the BiW in [29], independently designed but relying on a similar principle. The BiW

Algorithm 1 Hit and Run algorithm pseudo-code

Require: $f^{(1)}$, an initial point in $\mathcal{S}$.
Ensure: $f^{(1:N)}$, a sequence of N points in $\mathcal{S}$.
  for $i \in 2:N$ **do**
    $d_i \leftarrow$ uniformly sampled in the unit sphere of $\mathbb{R}^n$.
    $\lambda^{(i)+} \leftarrow \min\{\lambda > 0 : f^{(i-1)} + \lambda d_i \in \mathrm{Fr}(\mathcal{S})\}$, $I^{(i)+} = f^{(i-1)} + \lambda^{(i)+} d_i$.
    $\lambda^{(i)-} \leftarrow \min\{\lambda > 0 : f^{(i-1)} - \lambda d_i \in \mathrm{Fr}(\mathcal{S})\}$, $I^{(i)-} = f^{(i-1)} - \lambda^{(i)-} d_i$.

    $f^{(i)} \leftarrow$ uniformly sampled in $[I^{(i)+}, I^{(i)-}]$.
  end for

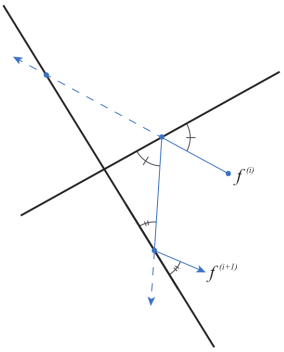


Fig. 1: Reflection of the path in the MiW and BiW algorithms

algorithm has been implemented in [4] for sampling polytopes in order to estimate their volumes.

As in HR, the general principle of MiW and BiW consists in drawing a direction and a path length. However, if the trajectory reaches a border of the polytope before the random path length is achieved, then it is reflected on this border, moving forward this reflected direction for the remaining distance. This is repeated until the trajectory finally reaches a point in the polytope; see Figure 1. Then, new direction and path length are randomly chosen, and so on. One major difference between HR and reflective algorithms is that the latter needs a parameter to be fitted, called the jump length of the algorithm.

Algorithm 2 Reflective Hamiltonian algorithms MiW and BiW pseudo-code

Require: $f^{(1)}$, an initial point in $\mathcal{S}$.
Ensure: $f^{(1:N)}$, a sequence of N points in $\mathcal{S}$
  for $i \in 2:N$ **do**
    $d_i \leftarrow$ sampled in unit sphere of $\mathbb{R}^n$.
    $L_i \leftarrow$ a random length for the trajectory.
    $d \leftarrow d_i$ (initial direction to follow)
    $L \leftarrow L_i$ (total length of the trajectory inside the polytope)
    $f \leftarrow f^{(i-1)+dL}$
    while $f \notin \mathcal{S}$ **do**
      $\lambda^* \leftarrow \min\{\lambda > 0 : f + d\lambda \in \mathrm{Fr}(\mathcal{S})\}$
      $z \leftarrow$ the normal vector to the hyperplan that reflects the trajectory
      $f^* \leftarrow f + d\lambda^*$ (intersection point in the hyperplan)
      $d \leftarrow (d - 2d^T zz)$ (update direction to the reflected one)
      $L \leftarrow (L - \lambda^*)$ (update the length of trajectory)
      $f \leftarrow f^* + dL$ (update candidate)
    end while
    $f^{(i)} \leftarrow f$
  end for

The difference between MiW and BiW lies in the determination of both the direction d and the length L, also different from HR; see Algorithm 2 for details.

In the MiW of [36], a vector $\mathbf{v}$ is drawn from a centered non correlated Gaussian vector $\mathbf{V} = (V_1, \ldots, V_n)$ with $\mathrm{Var}\, V_i = \sigma_i^2$. The vector $\boldsymbol{\sigma^2} = (\sigma_1^2, \ldots, \sigma_n^2)$ is called the jump length. Then d and L are obtained as the direction and norm of $\mathbf{v}$. The distribution of the sampled points is stated in [36] to converge to the uniform distribution on the polytope.

It is worth noting for sound comparison with the BiW, that when $\sigma_i = \sigma$ for all i, then the squared norm $||V||^2$ is $\sigma^2\chi^2(n)$ distributed, and hence its root $L = ||V||$ has a Nakagami distribution with expectation and variance

$$\mathbb{E}L = \frac{\Gamma((n+1)/2)}{\Gamma(n/2)}\sqrt{2}\sigma, \qquad \mathbb{V}\mathrm{ar}L = \sigma^2\left[n - 2\frac{\Gamma((n+1)/2)^2}{\Gamma(n/2)^2}\right].$$

Thanks to the law of large numbers, $L \sim \sqrt{n}\sigma$ for large n, and $\mathbb{V}\mathrm{ar}L$ converges to zero.

In the BiW of [29], the direction is drawn from the uniform distribution on the unit sphere of $\mathbb{R}^n$ and the path length is

$$L = \tau \log(1/U), \tag{9}$$

where U is drawn from a uniform distribution on $[0, 1]$, and the parameter $\tau > 0$ is called the jump length too. Note that L has an exponential distribution with parameter $1/\tau$, with expectation τ and variance τ^2, whatever be the dimension n of the polytope. The distribution of the sampled points is proven in [29] to converge to the uniform distribution on the polytope.

In both procedures, a major issue may occur depending on the choice of the jump length. Too small, a very large number of points would be needed to correctly sample the whole polytope. Too large, it would result into an important number of reflections and thus unnecessarily slow the process.

3.4 Implementation variants

According to the thorough studies [9] and [19], the CHRR is the best of HR variants for sampling LIM polytopes. So we have chosen to compare only CHRR to the reflective MiW and BiW. Available implementations of these algorithms, in various languages, generally addressed to a specific community, differ in their performances or outputs.

The implementation of CHRR from the MatLab {`COBRA`} toolbox [16] is, up to our knowledge, the only available one for sampling LIM polytopes. The function `chrrExpSampler()` takes a COBRA model structure as input and outputs the desired number of points. The number of iterations and the thinning parameter can be chosen. Unfortunately, this only works for upper and lower bounds on the flows, that is for biochemical models to which the implementation has been fitted. It does not work for inequations combining flows, which prevents its use for the more complex trophic or urban networks.

In contrast, [35] proposed, together with the MiW algorithm, an R package called {`limsolve`}, directed to sample LIM polytopes for trophic ecosystems. The function

`xsample()` takes one point of the polytope as input, and yields a set of randomly sampled points in the polytope as output. Among other parameters, the number of iterations and the burn-in period can be chosen. Three different random walks are implemented: HR, CHR and MiW. The latter, that was new, is also the most commonly used. An associated jump length vector can also be chosen, with default value the vector with coordinates one tenth of each range of the reduced polytope, easily computed through the function `xranges()`. This R package also provides annex functions that are of special interest for researchers in ecology. In particular, the function `setup()` takes as input a declaration file describing the LIM constraints and outputs the matrix representation. The declaration files are written in a more natural language than matrices and vectors, which makes the package easy to use for practitioners, furthermore thanks to recent improvements in [11]. Unfortunately, {`limsolve`} is extremely slow, mainly because it is entirely coded in R. Note also that, according to our study, the function `xranges`() sometimes yields absurdly large ranges, that make the default jump length vector unreliable and lead to even longer computation times.

For BiW, we use the R package {`volesti`} of [3], whose prime purpose is to sample general polytopes in order to approximate their volume. Still it can be used for LIM polytopes for all types of metabolic models; see [4]. The language R is here only an easy-to-use interface, while the computation part is coded in C++. Since C++ is a low level programming language –a language whose functions are closer to the processor's instructions, this leads to way more efficient computation. The function `sample_points()` takes as input the inequations defining the reduced polytope and the number of points to be sampled. Although polytopes may be defined through several representations, we will stick to the definition (5), and hence to the function `Hpolytope()`. While CHR, HR, and BiW can all be chosen, the latter is the default when targeting a uniform distribution. The starting point of the chain, the thinning parameter –there called walk length, the burn-in period, and the jump length τ in (9) are to be chosen. By default, the thinning parameter is set to 1, the burn-in period to 0, and the initial point $f^{(0)}$ is the center of the Chebyshev ball –the largest ball that fits inside the polytope– with radius r, obtained through the function `inner_ball()`. The default value for τ is

$$\tau = 4\sqrt{n}r. \tag{10}$$

Since it is of the order of the smallest range, this may lead to a poor exploration of the largest ranges, especially when the sandwiching ratio is large.

When analyzing the source code of the CRAN version of {`volesti`}[1], we uncovered two sticking points. First, the path length is said to be sampled as $L = \tau U$, instead of equation (9) as in [29]; this might be due to a typographical error since no justification is given for the change. Second, if the bound on the number of reflections typical of the BiW is reached, the point generated at the previous iteration is simply added to the returned sample before computing a new direction and length, leading to duplicates in the returned sample, thus clearly not uniform.

More recently, [4] proposed a multiphase Monte Carlo sampling algorithm addressed to the metabolic biochemical community. Its backbone is an accelerated variant of the BiW, performing both point and direction updates more efficiently by

[1] Version: 1.1.2-7, Date: 2023-09-18

Package	Function	Walk	Language	Community	Input	Ref.
{limsolve}	xsample()	MiW	R	trophic systems	polytope	[35]
{COBRA}	chrrExpSampler()	CHRR	MatLab	biochemical systems	polytope	[16]
{volesti}	sample_points()	BiW	C++ with R interface	generic	reduced polytope	[3]
{samplelim}	rlim()	MiW BiW	C++ with R interface	generic+t	polytope	

Table 1: The four considered implementations of MCMC algorithms

storing computations from the previous iteration, and including a preprocessing step involving the normal vectors of the facets. A multiphase part is also added, that is a sequence of sampling phases, each leading to a rounding of the original polytope. The efficiency of BiW is thus improved from one phase to the next, and the convergence to the uniform distribution accelerates. Unfortunately, this new algorithm is not yet a part of the CRAN version of {volesti} and therefore we have not been able to consider it for comparison.

We have taken into account all the qualities of the above implementations in order to produce a more efficient one. Indeed the new package {samplelim} combines the performance of {volesti} and the convenient features of {limsolve}. It keeps their shared ability to handle all types of LIM polytopes, which the MatLab implementation chrrExpSampler() of CHRR fails to do. Technically, {samplelim} is built from a fork of {volesti} version 1.1.2-3, of which all functions non necessary for sampling LIM polytopes have been removed. On the one hand, {samplelim} uses the same C++ structure as {volesti} in order to perform efficient computations of the MiW inside the polytope. Moreover, it allows one to choose between BiW and MiW. On the other hand, it contains all the annex functions available in {limsolve} in order to facilitate its practical use. Also {samplelim} uses the {Rglpk} package to solve the linear systems leading to the computation of the theorical ranges of each flow; this solves the issue of xranges() of {limsolve} using {lpSolve} that sometimes returns absurd values. In the BiW version of {samplelim}, we have also rectified one of the two sticking points noted above in the CRAN version of {volesti}: the path length is sampled according to equation (9), with a logarithm as prescribed in [29].

For easy comparison, Table 1 summarizes for each of the four packages above:

- the random walk used for exploring the polytope;
- the programming language in which the package is developed;
- the addressed community, where Generic stands for all types of metabolic networks, and generic+t indicates functions specific to the trophic systems community on top of the generic framework;
- the polytope given as input to the sampling function, either the general polytope defined in (3) or the reduced one in (5);
- the paper associated to the package.

4 Comparison of implementations on real data

This section aims at comparing through simulation some implementations of the MCMC sampling algorithms described in Section 3. Precisely, we focus here on the comparison between the updated implementation of MiW and BiW provided in the R packages {`limsolve`} with the function `xsample()`, {`samplelim`} with `rlim()`, and {`volesti`} with `sample_points()`, and with the CHRR implementation of the library {`COBRA`} of MatLab with the function `chrrExpSampler()`.

Computation time and sampling performances are the two essential aspects of sampling functions to take into account for a sound comparison. Clearly, they do not evolve independently. Generically speaking, higher computation time yields better sampling performances, but their relationships take various forms, depending, in particular, on the polytope shape and on the parameters of the sampling functions –here the jump lengths of MiW and BiW. Exploring exhaustively these relationships for the three sampling functions would require a huge, prohibitive, number of simulations.

Instead, in Section 4.2, we first compare `rlim()` with `xsample()`. This comparison aims at showing that both these MiW implementations lead to the same quality of samples, but that `rlim()` is much faster. Since the very long computation time of {`limsolve`} hinders its use on large real data, the comparison is performed only on the aggregated trophic model of Example 2. The aim of Section 4.3 is to compare more thoroughly the implementations of MiW in {`samplelim`}, BiW in {`volesti`}, and CHRR in {`COBRA`}. Precisely, `rlim()` and `sample_points()` will be compared to `chrrExpSampler()` for sampling the polytope associated to the biochemical network of Example 4. Inequlity contrainsts of Examples 1 and 3 involve combinations of several flows, that are not supported by `chrrExpSampler()`. Hence, only `rlim()` and `sample_points()` will be applied in both examples, and their performances compared.

In Sections 4.2 and 4.3, the sampling performances will be investigated through a methodology inspired by [9] and [19] of convergence and uniformity statistical diagnostics, presented in Section 4.1: the Raftery-Lewis and Geweke diagnostics and the Effective Sample Size, and a new numeric index that we call Range Coverage.

When comparing computation time performances, the CPU-time is used. All samples have been generated using a computer with an Intel Core i5 processor (i5-8600K, 3.60GHz× 6) and 15.5Go RAM. The R session used version 3.10.3 of both BLAS and LAPACK.

4.1 Sampling performance diagnostics

The Raftery and Lewis (RL) and Geweke diagnostics, and the Effective Sample Size (ESS) are all diagnostics of convergence. They are applied to the sequences of draws, and performed separately for each flow, inducing some insight on the uniformity of the distribution. They are completed by the new index Range Coverage (RC) based on the ranges of the flows. Additional diagnostics can be found in [9] and [19], such as the Hellinger distance or the Interval Based Scale Reduction Factor. Several visual diagnostics are also used in [11], with comments on the ambiguity of their interpretation. The diagnostics that we selected will prove sufficient for discriminating the performances of the different implementations for the mere purpose of comparison.

4.1.1 The Raftery and Lewis diagnostics

The RL introduced in [32] computes estimates of the thinning parameter, the burn-in period, and even the number of draws to be performed in an MCMC process. Its basis is that a correct sample of the objective distribution should give precise estimates of its quantiles.

For any pilot N_0-sample $f_{ij}^{(1:N_0)}$ of a given flow f_{ij} in (7) and $T \in \mathbb{N}^*$, the sequence S_T defined by

$$S_{T,l} = \mathbb{1}_{]-\infty,\widehat{q}]}\left(f_{ij}^{(1+Tl)}\right), \quad l \geq 0,$$

is a sequence of Bernoulli variables indicating whether each T-spaced sampled flow $f_{ij}^{(1+Tl)}$ is smaller than some quantile $\widehat{q}$ or not. The RL diagnostics processes as follows:

1. Compute the empirical estimator $\widehat{q}$ of the quantile q of f_{ij} of order $\alpha \in (0,1)$, where α is to be set by the user.
2. Let T be the smallest integer such that the amount of auto-correlation in S_T is small enough to be overlooked; precisely, such that the penalized likelihood ratio between the order 2 and order 1 Markov models for $(S_{T,l})$ is negative.
3. Set $M = TM'$, where M' is the smallest integer such that the M'-th power $P^{M'}$ of the (empirical) transition matrix P of S_T is close enough to its limit, whose both rows are $(\alpha, 1-\alpha)$. The closeness criteria is $\|e_i P^{M'} - (\alpha, 1-\alpha)\| < \epsilon$, for $i = 1, 2$, where $e_1 = (1,0)$, $e_2 = (0,1)$, $\|\cdot\|$ is the euclidean norm on $\mathbb{R}^2$, and $\varepsilon > 0$ is a parameter to be set by the user.
4. Set $N_0^{\mathrm{RL}} = M + TN'$, where N' is computed as follows. For any given N', the central limit theorem for Markov chains applied to the vector $(S_{T,l})_{l=0,\ldots,N'-1}$ yields that the empirical mean $\frac{1}{N'}\sum_{l=0}^{N'-1} \mathbb{1}_{]-\infty,\widehat{q}]}\left(f_{ij}^{(1+Tl)}\right)$ belongs to an interval $[\alpha - r, \alpha + r]$ with probability $p_{N'}$ that increases to 1 as N' goes to infinity, where the value of r is to be set by the user. The value to be retained for N' is the smallest integer such that $p_{N'}$ is greater than some prescribed value p close to 1.

The integers T and M returned by the RL diagnostics are estimates of the thinning parameter and burn-in period to be applied to the pilot sample. The integer N_0^{RL} is an estimate of the size of a pilot sample to draw for ensuring a good precision in estimating a prescribed quantile of the flows.

From a computational point of view, the parameters T, M, and N_0^{RL} are derived for all flows at once thanks to the function `raftery.diag()` of the R package {`coda`}. This requires five arguments: the pilot sample of points, the order α of the quantile to be estimated at first step, the tolerance level ε, the tolerance r required on the order of the quantile and the probability confidence level p. Note that the same function determines an estimate $N_0^{\min}$ of what would be the size of an idealized pilot sample, i.e., without correlation. It is determined as in Step 4 above, by assuming that $(S_{T,l})$ is an independent sequence instead of a Markov chain of order 1. We will not use this feature below, since it is not pertinent for comparison purposes.

Finally, the dependence factor

$$I = \frac{N_0^{\mathrm{RL}}}{N_0^{\min}} \tag{11}$$

assesses the extent to which auto-correlation inflates the required sample size; values larger than 5 indicate a strong auto-correlation.

In Sections 4.2 and 4.3 below, for the RL diagnostics, we have chosen the (default) parameters $\alpha = 0.025$, $\varepsilon = 0.001$, $r = 0.005$, $p = 0.95$. Note that the RL diagnostics will here only be used to compare the quality of the different pilot samples through the values of M, T, N_0^{RL} and I in (11) and not to perform further sampling of size N_1.

4.1.2 The Geweke diagnostics

The Geweke diagnostics introduced in [13] is a classical statistical test of comparison of means, applied to prescribed proportions of the first and last points generated by an MCMC algorithm, say $0 < p_1 < 1$ and $0 < p_2 < 1$, with $p_1 + p_2 < 1$. If the difference of means is too large, the null hypothesis of convergence is rejected.

For any pilot N_0-sample $f_{ij}^{(1:N_0)}$ of a given flow f_{ij} generated by an MCMC procedure,

$$Z = \frac{\overline{f}_{ij}^{(2)} - \overline{f}_{ij}^{(1)}}{\sqrt{\mathrm{se}_{ij}^{(1)} + \mathrm{se}_{ij}^{(2)}}}$$

is the standardized difference between the two empirical means, say $\overline{f}_{ij}^{(k)} = \frac{1}{np_k} \sum_{l=1}^{Np_k} f_{ij}^{(l)}$ for $k = 1, 2$, of the first and last parts of the sample, where $\mathrm{se}_{ij}^{(1)}$ and $\mathrm{se}_{ij}^{(2)}$ are the standard errors. Under the null hypothesis, Z is asymptotically normal. Thus, the convergence of the algorithm is rejected if $|Z|$ exceeds a prescribed quantile of the normal distribution, usually 1.28 corresponding to a significance level of 0.2. In practice, several independent pilot samples are drawn and the number of times the null hypothesis that the draws are stationary is rejected is computed. The number of rejections over the number of pilot samples is theoretically close to 0.2, corresponding to the significance level of the test.

In Sections 4.2 and 4.3 below, parameters for the Geweke diagnostics have been set to $p_1 = 0.1$ and $p_2 = 0.5$, with quantile 1.28. A number of rejections of 2 out of 10 at most is expected for good sampling.

4.1.3 The Effective Sample Size

The effective sample size (ESS) of a correlated sample obtained through any procedure is the theoretical number of independent draws that would yield the same variance; for the MCMC procedure, see, e.g., [10]. In mathematical words, for any N-sample $f_{ij}^{(1:N)}$ of the flows,

$$\mathrm{ESS}(f_{ij}^{(1:N)}) = \frac{N}{1 + 2\sum_{k=1}^{N-1} \rho_k(f_{ij}^{(1:N)})},$$

where $\rho_k(f_{ij}^{(1:N)}) = \mathrm{corr}(f_{ij}^{(1:(N-k))}, f_{ij}^{((1+k):N)})$ is the auto-correlation of order k of $f_{ij}^{(1:N)}$.

n	`rlim()`	`xsample()`
50	0.217	0.271
100	0.219	0.367
500	0.256	1.117
1 000	0.297	2.079
5 000	0.615	9.818
10 000	1.011	19.396
50 000	4.163	96.101

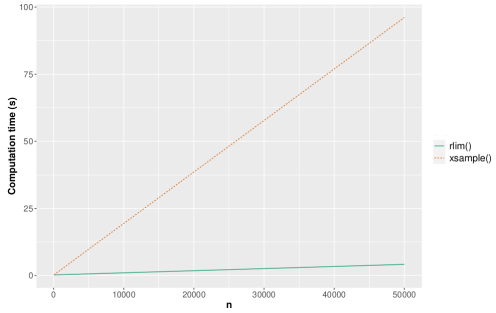


Fig. 2: Computation time of `xsample()` (dotted line) and `rlim()` (continuous line) in Example 2 (time in seconds).

This score reflects the amount of auto-correlation of the MCMC draws: the closer the ESS is to N, the less the MCMC draws are auto-correlated. Up to our knowledge, no numeric bound is available in the literature to assess the quality of the score.

4.1.4 The Range Coverage

The bounds m_{ij} and M_{ij} on the flows in (4) can be accurately computed from the matrices A et G by using a linear programming solver. This yields theoretical ranges $R_{ij} = M_{ij} - m_{ij}$ for all the flows.

Here we propose to compute empirical estimates of these ranges from a sample $f_{ij}^{(1:N)}$, and to compare them to the theoretical ones. Precisely, let us define the range coverages as

$$\mathrm{RC}_{ij} = \frac{\widehat{M}_{ij} - \widehat{m}_{ij}}{M_{ij} - m_{ij}}, \tag{12}$$

where $\widehat{m}_{ij} = \min_{l\in 1:N} f_{ij}^{(l)}$ and $\widehat{M}_{ij} = \max_{l\in 1:N} f_{ij}^{(l)}$, for $ij \in E$.

This produces a new index RC for assessing sampling performance. Quite obviously, the closer these rates are to 1, the wider is the percentage of area explored in the polytope.

4.2 Comparison of the functions `rlim()` and `xsample()`

We will show here that `rlim()` leads to the same quality of samples as `xsample()`, but much faster. Due to the very long computation time of `xsample()`, this comparison can only be conducted on relatively small models, here the aggregated trophic model of Example 2. The comparison is divided in two distinct parts, one to compare computation time and one to compare the quality of samples.

The first part consists in comparing the computation time[2] needed to obtain pilot samples of size N_0 varying from 50 to 50 000, when using the default jump length, that is the same for both implementations. For each sample size, 10 replicates are computed. Figure 2 presents the mean computation time, expressed in seconds, associated to

[2]By computation time, we mean CPU-time, i.e., the amount of time the CPU cores of the computer specifically spend on the computations.

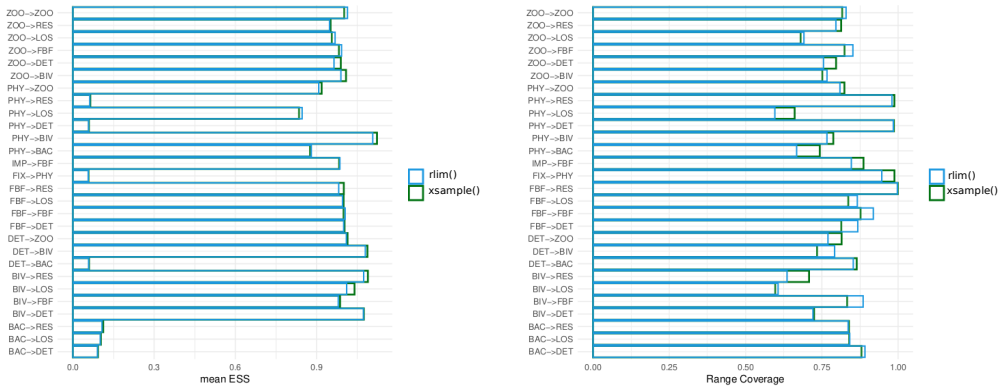
Flow	Algorithm	RL diagnostic M	T	N_0^{RL}	$N_0^{\min}$	I	#G rej.
1	`rlim()`	9.3	2.7	11 900	3 750	3.170	3
	`xsample()`	10.2	3.0	13 900	3 750	3.710	3
2	`rlim()`	1.9	1.0	3810	3750	1.020	0
	`xsample()`	1.9	1.0	3 760	3 750	1.000	1
3	`rlim()`	1.9	1.0	3 690	3 750	0.984	2
	`xsample()`	1.9	1.0	3 680	3 750	0.981	3
4	`rlim()`	2.0	1.0	3 750	3 750	1.000	3
	`xsample()`	1.9	1.0	3 720	3 750	0.992	3
5	`rlim()`	10.1	2.9	13 200	3 750	3.520	3
	`xsample()`	11.3	3.1	14 600	3 750	3.890	3
6	`rlim()`	6.9	2.3	10200	3750	2.720	2
	`xsample()`	7.8	2.6	11 400	3 750	3.030	4
7	`rlim()`	10.3	2.8	12 700	3 750	3.380	3
	`xsample()`	10.7	2.9	13 400	3 750	3.570	3
8	`rlim()`	2.0	1.0	3 650	3 750	0.975	2
	`xsample()`	2.0	1.0	3 680	3 750	0.982	3
9	`rlim()`	2.0	1.0	3 730	3 750	0.996	5
	`xsample()`	1.9	1.0	3 730	3 750	0.995	3
10	`rlim()`	1.9	1.0	3 790	3 750	1.010	3
	`xsample()`	1.7	1.0	3 790	3 750	1.010	2
11	`rlim()`	1.9	1.0	3 780	3 750	1.010	4
	`xsample()`	1.9	1.0	3 790	3 750	1.010	1
12	`rlim()`	6.3	2.1	8 860	3 750	2.370	2
	`xsample()`	6.9	2.3	9 740	3 750	2.600	4
13	`rlim()`	7.6	2.4	10 200	3 750	2.730	2
	`xsample()`	7.4	2.5	10 700	3 750	2.870	4
14	`rlim()`	8.0	2.6	11 500	3 750	3.080	4
	`xsample()`	9.0	2.8	12 500	3 750	3.320	2
15	`rlim()`	1.9	1.0	3 760	3 750	1.000	2
	`xsample()`	1.9	1.0	3 740	3 750	0.997	1
16	`rlim()`	1.9	1.0	3 740	3 750	0.998	3
	`xsample()`	2.0	1.0	3 750	3 750	1.000	3
17	`rlim()`	1.9	1.0	3790	3750	1.010	1
	`xsample()`	1.7	1.0	3 740	3 750	0.998	0
18	`rlim()`	1.9	1.0	3 730	3 750	0.994	2
	`xsample()`	2.0	1.0	3 730	3 750	0.996	0
19	`rlim()`	1.9	1.0	3 730	3 750	0.996	1
	`xsample()`	1.9	1.0	3 730	3 750	0.998	2
20	`rlim()`	1.9	1.0	3 750	3 750	1.000	3
	`xsample()`	1.9	1.0	3 730	3 750	0.994	3
21	`rlim()`	2.0	1.0	3 680	3 750	0.982	2
	`xsample()`	1.9	1.0	3 680	3 750	0.982	5
22	`rlim()`	1.9	1.0	3 700	3 750	0.987	2
	`xsample()`	1.9	1.0	3 700	3 750	0.988	3
23	`rlim()`	1.7	1.0	3 780	3 750	1.010	1
	`xsample()`	1.8	1.0	3 720	3 750	0.992	2
24	`rlim()`	1.9	1.0	3 770	3 750	1.000	2
	`xsample()`	2.0	1.0	3 780	3 750	1.010	0
25	`rlim()`	1.6	1.0	3 770	3 750	1.000	1
	`xsample()`	2.0	1.0	3 760	3 750	1.000	2
26	`rlim()`	1.8	1.0	3 780	3 750	1.010	3
	`xsample()`	1.9	1.0	3 750	3 750	1.000	2
27	`rlim()`	1.9	1.0	3 770	3 750	1.010	3
	`xsample()`	2.0	1.0	3 720	3 750	0.994	0
28	`rlim()`	2.0	1.0	3 720	3 750	0.992	3
	`xsample()`	1.8	1.0	3 730	3 750	0.995	2

Table 2: Aggregated results of the RL diagnostics and Geweke number of rejections of `xsample()` and `rlim()` for 10 replicates of 20 000 points in the polytope of Example 2.

the 10 replicates for each sample size. Clearly, `rlim()` outperforms `xsample()` by a factor close to 10 for sample sizes greater than 1 000. This is essentially explained by the low-level optimized implementation of `rlim()` in C++ compared to the high-level implementation of `xsample()` in direct R programming. The performance gap between C or C++ and R is a widely discussed topic on developers blogs, especially in the MCMC community; see, e.g. [21, 37], where examples are detailed for which the outperformance factor goes from 50 to 300. Hybrid approaches using seamless C++ integration tools for R, as used in {`samplelim`}, offer substantial performance improvements while preserving the easy-to-use interface of R.

Additional simulation, not included here, suggests that the speed difference between `rlim()` and `xsample()` increases with the size of the jump length and also with the complexity, dimension, and shape, of the polytope. They confirm comments in [9], where the authors failed to use `xsample()` on most real-size metabolic biochemical systems, due to unreasonable computation time.

The second part of the comparison between `xsample()` and `rlim()` concerns sampling performances, based the diagnostics and indexes presented above in Section 4.1.



(a) Mean ESS ratios of the 28 flows

(b) Mean RC of the 28 flows

Fig. 3: Means of the 10 replicates of `rlim()` in blue and `xsample()` in green. The categories of compartments (on the left) for Example 2 are described in [2]

First 10 replicates of a pilot sample of size $N_0 = 20\ 000$ are simulated with each function. For the RL diagnostics, M, T, N_0^{RL} and $N_0^{\min}$ are aggregated by taking the maximum value of the 10 samples, while the RL autocorrelation factor I is computed as their mean. The results of the Geweke diagnostic are the number of pilot samples among the 10 that failed the Geweke test ($|Z| > 1.28$). These scores are presented in Table 2, while the ESS and RC given in Figure 3 are the means of these indexes on the 10 samples.

These diagnostic outputs attest that both `xsample()` and `rlim()` yield MCMC samples of the polytope of Example 2 with similarly good convergence and auto-correlation properties. Small values for both the thinning parameter T and index I of RL diagnostics and the high values of ESS ratios for a large majority of flows indicate moderate auto-correlations. Small values of the burn-in period M, and the number of Geweke rejections close to the expected 2, that is 2.29 for `xsample()` and 2.39 for `rlim()`, indicate that convergence to the limit distribution is a statistically sound assumption even without discarding the first points of the pilot sample. The mean range coverages, with no RC lower than half the theorical range, indicate that an important part of the polytope has been explored. Further, these diagnostics highlight the very close behavior of `xsample()` and `rlim()`, since both lead to very similar performance results on all flows. The small residual differences may simply originate from the difference of the used pseudo-random number generating processes.

4.3 Comparison of the functions `rlim()`, `sample_points()` and `chrrExpSampler()`

This section aims to compare the sampling performances of `rlim()` (MiW in {`samplelim`}) with the functions `sample_points()` (BiW in {`volesti`}) and `chrrExpSampler()` (CHRR in {`COBRA`}). We will consider comparison on the basis of similar computation times: the parameters of the algorithms for the studied examples

will be tuned in order for the simulation of a fixed number of points of the polytope to take the same time for the three functions. Then the quality of the samples will be compared. Since the CHRR implementation only allows for lower and upper bounds on the flows, its comparison will be possible only on the biochemical model of Example 4. First, the two others will be compared on the trophic model of Example 1 and the urban model of Example 3.

Both MiW and BiW depend on jump lengths, that have an important impact on the computation time and the quality of the sample. In `sample_points()`, the default value of the jump length τ is given by (10), while for `rlim()` the default jump length $\boldsymbol{\sigma}$ is one tenth of the vector of the reduced polytope ranges. Comparing results obtained for a vector parameter with respect to a scalar one turns out to be difficult to handle. Fortunately, for all $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n)$ we have been able to determine numerically a scalar value σ such that $(\sigma, \ldots, \sigma)$ leads to a similar computation time for `rlim()`, with similar sampling performances; for the sake of shortness, these comparison results are not given here. This replacement questions the choice of a vector jump length in MiW with respect to a scalar one in BiW. Yet, since it is an open question to find a closed-form expression for σ as a function of $\boldsymbol{\sigma}$, the easy-to-compute vector value is to be favored by practitioners. For the purpose of comparison, $\boldsymbol{\sigma}$ will below be replaced by the corresponding computed σ. In other words, we will consider dual values τ and σ leading to similar computation times for `rlim()` and `sample_points()`, meaning that only the quality of the output samples will remain to be compared.

Further, MiW and BiW algorithms then differ mainly in the choice of the distribution of the path length L: Nakagami for MiW, and exponential for BiW; see Section 3.3. The path length value directly impacts the number of reflections of the trajectories, while the computation time is roughly a linear function of the number of reflections. Thus, we expected that, for a similar computation time, the path length mean would be essentially similar, and hence the samples produced by `rlim()` and `sample_points()` would present similar qualities. Still, the sampling performances of the latter appear to be highly penalized by the included bound on the maximal number of reflections. Practically, for large values of τ, the probability that a trajectory exceeds the maximal number of reflections increases and may get close to 1; for these trajectories, the returned sample point is then the previous point. This often happens in the simulated sample, resulting in very poor sampling performances with respect to all statistical indexes.

The following three-step procedure will be applied for the comparison to highlight above comments:

Preparation step. For each pertinent example and sampling function, we simulate 10 replicates of samples of moderate size 1 000, for 30 different values of τ and σ and record the corresponding computation times. This allows us to empirically identify values of τ and σ with similar computing times, that we will call dual values. Under the reasonable assumption that the computation time grows linearly with the number of points to sample, sampling larger sized samples for dual values will take similar times.

Selection step. We compute three values σ dual to the following values of τ:

- the default value (10) for the reduced polytope;
- the value with the highest global sampling performances in the preparation step;

Algorithm functions	Jump lengths	Comp. times (s)	M (max)	N_0^{RL} (max)	T (mean)	I (mean)	# Geweke rej. (mean)	ESS (mean)	Range Cov. (mean)
`sample_points()`	$\tau = 7.8 \times 10^{-4}$ (default)	1.46	13 803	1 115 920	14	78	7.98	139	0.06
`rlim()`	$\sigma = 5 \times 10^{-5}$	1.74	1 045	939 525	15	80	8.06	194	0.06
`sample_points()`	$\tau = 0.7$	190	929	1 060 313	15	45	6.41	1 140	0.41
`rlim()`	$\sigma = 0.03$	180	866	641 974	17	47	5.90	1 588	0.45
`sample_points()`	$\tau = 100$	1 120	15 178	3 396 166	1	433	7.10	4	0.08
`rlim()`	$\sigma = 0.2$	1 190	1 329	567 356	14	27	5.64	4 151	0.58
`rlim()`	$\sigma = 1.5$	9 166	811	609 165	12	8	3.79	9 961	0.72

Table 3: Comparison of `sample_points()` and `rlim()` on the trophic network of Example 1

- a value large enough to get a high proportion of trajectories whose number of reflections exceeds the upper bound. This value is selected empirically during the preparation step.

Simulation step. For each of the couple dual values, we simulate by using each sampling functions 10 samples of size 20 000. For each replicate, the convergence and uniformity diagnostics of Section 4.1 are performed. The results are aggregated first by a mean of the 10 replicates, then either by mean or a maximum of the flows, according to the index. This results into a single score value for each index and each sampling function.

Table 3 and 4 and the four first rows of Table 5 show the aggregated diagnostics results for `rlim()` and `sample_points()` for Examples 1, 3 and 4. The three first rows correspond to the three choices of dual values. The fourth one corresponds to large values of σ leading to better diagnostic results than the three dual values. The last row of Table 5 shows the comparison with `chrrExpSampler()` for Example 4, for which the jump lengths τ and σ of `rlim()` and `sample_points()` are chosen to lead to similar computation times for the three functions.

4.3.1 Comparison of `rlim()` and `sample_points()` on the trophic model of Example 1

This model is the more complex considered in this paper, with linear inequations involving several flows.

The very small default value τ of `sample_points()`, close to 7.8×10^{-4}, leads to a poor sampling of the polytope, see Table 3. The default value σ of `rlim()` leads to very satisfactory samples but takes way too long to be computed, about 20 hours for 20 000 solutions.

Above a jump length treshold, precisely $\tau = 1$ here according to the preparation step, the samples returned by `sample_points()` show a drop in quality, a behavior that `rlim()` does not share.

Table 3 shows the default value $\tau = 7.8 \times 10^{-4}$, the optimal value according to the statistical indexes $\tau = 0.7$, and a value above the critical threshold, precisely $\tau = 100$ for an easily noticeable effect. The dual jump lengths for `rlim()` chosen to yield the same computing times are respectively $\sigma = 5 \times 10^{-5}$, $\sigma = 0.03$ and $\sigma = 0.2$.

The sampling performances of `sample_points()` for $\tau = 7.8 \times 10^{-4}$ and `rlim()` for $\sigma = 5 \times 10^{-5}$ are not significantly different. Only the burn-in period M and the pilot sample size N_0^{RL} recommanded by the RL diagnostics really differ, both to the benefit of `rlim()`. The same is true for $\tau = 0.7$ and $\sigma = 0.03$. In all cases, the results of the

Algorithm functions	Jump lengths	Comp. times (s)	M (max)	N_0^{RL} (max)	T (mean)	I (mean)	# Geweke rej. (mean)	ESS (mean)	Range Cov. (mean)
`sample_points()`	$\tau = 2\ 600$ (default)	0.37	558	751 242	12	75	6.96	509	0.30
`rlim()`	$\sigma = 200$	0.42	755	1 014 509	12	83	6.68	697	0.30
`sample_points()`	$\tau = 10^6$	78	549	479 050	15	16	7.12	2 825	0.81
`rlim()`	$\sigma = 6 \times 10^4$	70	495	579 383	12	16	4.06	7 035	0.80
`sample_points()`	$\tau = 10^7$	220	777	1 228 831	2	132	7.87	4	0.10
`rlim()`	$\sigma = 2 \times 10^5$	228	472	433 045	15	14	3.54	10 590	0.85
`rlim()`	$\sigma \approx 3 \times 10^6$ (default)	3 703	25	31 601	6	2	2.20	16 182	0.98

Table 4: Comparison of `sample_points()` and `rlim()` on the urban network of Example 3

tests are way under what is required for good samples, with in particular, a number of Geweke rejections of 6.

For $\tau = 100$, the sampling performances of `sample_points()` drop significantly, with much lower quality than `rlim()` with the dual $\sigma = 0.2$. The recommended thinning parameter T returned by the RL diagnostic of 1 for all flows is hard to interpret; it may result from a very large number of duplicates in the sample. Note that `rlim()` with the dual $\sigma = 0.2$ leads to way better sampling performances.

At the cost of a longer computation time, a greater jump length for `rlim()` always yields better sampling performances according to all indexes. See Table 3, where $\sigma = 1.5$ leads to the best results, but at the cost of a computation time of about 2.5 hours.

4.3.2 Comparison of `rlim()` and `sample_points()` on the urban model of Example 3

Again, we will compare the sampling performances for jump lengths leading to similar computation times on this network of middle complexity. For `sample_points()`, the default $\tau = 2\ 600$, the optimal value of the preparation step according to all indexes $\tau = 10^6$, and a larger value leading to duplicates in the returned samples, say $\tau = 10^7$, lead to choose for `rlim()` respectively $\sigma = 200$, $\sigma = 6 \times 10^4$ and $\sigma = 2 \times 10^5$. The comparison is summarized in Table 4, where the default jump length $\sigma \approx 3 \times 10^6$ of `rlim()` is also presented.

For the default jump length $\tau = 2\ 600$ and the dual $\sigma = 200$, close values are obtained. Both samplings are of very poor quality with a high autocorrelation, $I \approx 80$, a number of Geweke rejections of 7, low ESS and RC.

For $\tau = 10^6$, the autocorrelation drops to $I = 16$, and both ESS and RC are higher, with especially RC≈ 0.8. The results of the Geweke test are no better than with the default τ. On the other hand, for the dual $\sigma = 6 \times 10^4$, `rlim()` leads to a better mean ESS of 7 035 compared to 2 825 for `sample_points()`, and a better number of Geweke rejections of 4 instead of 7.12, the value of the other indexes being similar.

For $\tau = 10^7$ and $\sigma = 2 \times 10^5$, the value of all indexes are clearly better for `rlim()`. Again, note the hard to interpret low value of the thinning parameter T returned by the RL diagnostic on samples obtained with `sample_points()`. According to all indexes, the sampling performances of `sample_points()` are really bad: absurdly high $I = 132$, number of Geweke rejections close to 8, and very low ESS and RC. In contrast, `rlim()` ouputs samples of way better quality even if still not what is to be required.

Finally, for `rlim()` with the default jump length σ , the sampling performances are really good, and the computation time not unreasonable, about one hour for a sample

Algorithm functions	Jump lengths	Comp. times (s)	M (max)	N_0^{RL} (max)	T (mean)	I (mean)	# Geweke rej. (mean)	ESS (mean)	Range Cov. (mean)
`sample_points()`	$\tau = 57.8$ (default)	1.38	265	289 862	11	6.59	3.11	1909	0.53
`rlim()`	$\sigma = 6$	1.43	255	267 457	12	7.4	3.19	1821	0.53
`sample_points()`	$\tau = 500$	7.08	29	35079	6	1.3	1.97	12 457	0.55
`rlim()`	$\sigma \approx 50$ (default)	7.68	24	28 172	5	1.2	1.57	15 281	0.56
`sample_points()`	$\tau = 50000$	53.1	41 430	3 054 460	1	321.1	7.51	3	
`rlim()`	$\sigma = 400$	53.3	3	4675	1	1.0	2.17	17 829	0.56
`rlim()`	$\sigma = 1000$	132	2.00	3870	1	1.0	2.23	18 302	0.56
`sample_points()`	$\tau = 1000$	13.6	15	19 655	4	1.2	1.60	14 614	0.56
`rlim()`	$\sigma = 100$	13.8	16	19 941	4	1.1	1.77	17 202	0.56
`chrrExpSampler()`		13.5	1101	1 257 021	8	80.8	5.08	103	0.17

Table 5: Comparison of `sample_points()`, `rlim()` and `chrrExpSampler()` on the biochemical network of Example 4

of size 20 000. The RL index I is under the threshold of 5, the number of Geweke rejections close to the expected 2, and both ESS and RC values are satisfying.

4.3.3 Comparison of `rlim()`, `sample_points()`, `chrrExpSampler()` on the biochemical model of Example 4

No parameter alike a jump length is to be fitted for the CHRR random walk of `chrrExpSampler()`. Thus, in order to compare the sampling performance of the three implementations, we have chosen the jump lengths $\tau = 1\,000$ for `sample_points()` and $\sigma = 100$ for `rlim()` leading to computation times similar to `chrrExpSampler()`. Both `rlim()` and `sample_points()` clearly outperform `chrrExpSampler()`. The samples returned by the latter are much more auto-correlated with $I = 80.8$ versus $I \approx 1$, show the highest number of Geweke rejections and very low ESS and RC. This may be due to the CHR exploration method where two following points in the chain only differ by one coordinate.

Aside comparison with `chrrExpSampler()`, we have also compared `rlim()` and `sample_points()`. Fitted for the same computation time, the two functions return samples of the same order of quality according to all indexes, as shown on rows 1 and 2 of Table 5, until the jump length of `sample_points()` is higher than a threshold and leads to duplicates in the returned sample. Flow values out of their theoretical ranges even happen, and hence the RC for `sample_points()` with $\tau = 50\,000$ does not appear on row 3 of Table 5; we fail to figure out a reason for this phenomena. Finally, as shown on row 4, `rlim()` with a large $\sigma = 1\,000$ leads to a computation time close to three times longer than $\sigma = 400$, with only a negligible improvement in the quality of the returned samples : the jump length $\sigma = 400$ seems to be large enough to get the best possible results on this model.

5 Conclusion

In this paper, we designed a new R package called {`samplelim`}, and compared it with other existing computing packages for sampling polytopes in linear inverse modeling.

Its sampling function `rlim()` was compared with the functions `xsample()` of the R package {`limsolve`}, `sample_points()` of the R package {`volesti`}, and `chrrExpSampler()` of the Matlab toolbox {`COBRA`}. The comparison was made both

in terms of computation time and sample quality. To sum up the comparison results, `rlim()`:

- leads to the same quality of samples as `xsample()` but much faster;
- is more reliable than `sample_points()` which has a threshold on the jump length leading to a drop in the quality of samples;
- is more efficient than `chrrExpSampler()` in terms of computation time, quality of the returned sample and, last but not least, of complexity of the concerned models.

Besides the high qualities of its sampling function, the new package {`samplelim`} keeps the easy to use features of {`limsolve`}, as well as its annex functions that are important to practitioners.

Most of the time, when information is gathered on the simple elements of a trophic, biochemical or urban metabolic networks, the elements to be considered are very numerous. Then, when building the model network, this number has to be reduced, by aggregating some elements within the same compartment or edge of the network. This is mainly due to the very long computation time associated to the needed number of iterations for stabilizing the results. With the new R package {`samplelim`}, the very high gain of time and quality will allow the development of these models at a level of aggregation that will be closer to the study scale of the biological processes. Furthermore, coupling networks together may become possible for studying them in a spatial framework.

References

[1] Bonarius HP, Schmid G, Tramper J (1997) Flux analysis of underdetermined metabolic networks: the quest for the missing constraints. *Trends in Biotechnology* 15(8):308-314. https://doi.org/10.1016/S0167-7799(97)01067-6

[2] Caputo JG, Girardin V, Knippel A, Nguyen H, Niquil N, Noguès Q (2021) Analysis of trophic networks: an optimisation approach, *Journal of Mathematical Biology* 83:53. https://doi.org/10.1007/s00285-021-01682-3

[3] Chalkis A, Fisikopoulos V (2021) {`volesti`}: Volume Approximation and Sampling for Convex Polytopes in R. *The R Journal* 13:642-660. https://doi.org/10.32614/RJ-2021-077

[4] Chalkis A, Fisikopoulos V, Tsigaridas E, Zafeiropoulos H (2021) Geometric algorithms for sampling the flux space of metabolic networks. *The 37th International Symposium on Computational Geometry*. Buffalo, USA. https://doi.org/10.4230/LIPIcs.SoCG.2021.21

[5] Chen S, Fath BD, Chen B (2010) Information indices from ecological network analysis for urban metabolic system. *Procedia Environmental Sciences* 2:720-724. https://doi.org/10.1016/j.proenv.2010.10.082

[6] Chevallier A, Pion S, Cazals F (2018) Hamiltonian Monte Carlo with boundary reflections, and application to polytope volume calculations, *Research Report RR-9222*. INRIA Sophia Antipolis, France. https://hal.science/hal-01919855

[7] Covert MW, Schilling CH, Famili I, Edwards JS, Goryanin II, Selkov E, Palsson BO (2001) Metabolic modeling of microbial strains in silico. *Trends in biochemical sciences* 26(3):179-186. https://doi.org/10.1016/S0968-0004(00)01754-0

[8] Duarte NC, Becker SA, Jamshidi N, Thiele I, Mo ML, Vo TD, Srivas R, Palsson BO (2007) Global reconstruction of the human metabolic network based on genomic and bibliomic data. *Proceedings of the National Academy of Sciences* 104(6):1777-1782. https://doi.org/10.1073/pnas.0610772104

[9] Fallahi S, Skaug HJ, Alendal G (2020) A comparison of Monte Carlo sampling methods for metabolic network models. *PLoS One* 15(7). https://doi.org/10.1371/journal.pone.0235393

[10] Gelman A, Carlin JB, Stern HS, Dunson DB, Vehtari A, Rubin DB (2013) *Bayesian Data Analysis*. CRC Texts in Statistical Science, 3rd Edition Chapman & Hall. https://doi.org/10.1201/b16018

[11] Gerber G, Brooker B, Scharler UM (2023) Automated workflow for incorporation and evaluation of data uncertainty in ecological networks with autoLIMR. *Ecological Informatics* 102375. https://doi.org/10.1016/j.ecoinf.2023.102375

[12] Grami B, Rasconi S, Niquil N, Marlène J, Saint-Béat B, Sime-Ngando T (2011) Functional Effects of Parasites on Food Web Properties during the Spring Diatom Bloom in Lake Pavin: A Linear Inverse Modeling Analysis. *PLoS One* 6. https://doi.org/10.1371/journal.pone.0023273

[13] Geweke J (1991) Evaluating the accuracy of sampling-based approaches to calculating posterior moments. In: Bernado JM, Berger JO, Dawid AP, Smith AF (ed) *Bayesian Statistics 4*. Clarendon Press, Oxford, UK, pp 169-194. https://doi.org/10.1093/oso/9780198522669.003.0010

[14] Guéret S, Winiwarter W (2022). *Deliverable D2/3 of the UNCNET project: Using probability approaches to inform, revise, and improve contributions on the respective nitrogen flows. Funded under the JPI Urban Europe / China pilot call.* https://www.uncnet.org/wp-content/uploads/sites/17/2022/02/UNCNET_D2.3.pdf

[15] Hannon B (1973) The structure of ecosystems. *Journal of theoretical biology* 41(3):535-546. https://doi.org/10.1016/0022-5193(73)90060-X

[16] Haraldsdóttir HS, Cousins B, Thiele I, Fleming RM, Vempala S (2017) CHRR: coordinate hit-and-run with rounding for uniform sampling of constraint-based models. *Bioinformatics* 33(11):1741-1743. https://doi.org/10.1093/bioinformatics/btx052

[17] Haskell KH, Hanson RJ (1981) An algorithm for linear least squares problems with equality and nonnegativity constraints. *Mathematical Programming* 21:98-118. https://doi.org/10.1007/BF01584232

[18] Havránek M (2009) *ConAccount 2008: Urban Metabolism, Measuring the Ecological City*. Charles University Environment Center. Prague, Czechia.

[19] Herrmann HA, Dyson BC, Vass L, Johnson GN, Schwartz JM (2019) Flux sampling is a powerful tool to study metabolism under changing environmental conditions. *NPJ systems biology and applications* 5(1):32. https://doi.org/10.1038/s41540-019-0109-0

[20] Kaltenegger K, Bai Z, Dragosits U, Fan X, Greinert A, Guéret S, Suchowska-Kisielewicz M, Winiwarter W, Zhang L, Zhou F (2023) Urban nitrogen budgets: Evaluating and comparing the path of nitrogen through cities for improved management. *Science of the Total Environment* 904:166827. https://doi.org/10.1016/j.scitotenv.2023.166827

[21] Kinlay J (2018) A comparison of Programming Languages. *Quantitative Research and Trading*. http://jonathankinlay.com/2018/10/comparison-programming-languages/. Accessed 24 January 2024.

[22] Kones JK, Soetaert K, van Oevelen D, Owino JO, Mavuti K (2006) Gaining insight into food webs reconstructed by the inverse method. *Journal of Marine Systems* 60:153–166. https://doi.org/10.1016/j.jmarsys.2005.12.002

[23] Kones JK, Soetaert K, van Oevelen D, Owino JO (2009) Are network indices robust indicators of food web functioning? A Monte Carlo approach. *Ecological Modelling* 220:370–382. https://doi.org/10.1016/j.ecolmodel.2008.10.012

[24] Laddha A, Vempala SS (2023) Convergence of Gibbs sampling: coordinate Hit-and-Run mixes fast. *Discrete & Computational Geometry* 70:406-425. https://doi.org/10.1007/s00454-023-00497-x

[25] Leguerrier D (2005) Construction et étude d'un modèle de réseau trophique de la vasière de Brouage (bassin de Marennes Oléron, France). PhD Thesis, Université de La Rochelle, France. https://archimer.ifremer.fr/doc/00000/2260

[26] Niquil N, Saint-Béat B, Johnsin GA, Soetaert K, van Oevelen D, Bacher C, Vézina AF (2011) *Inverse Modeling, in Modern Ecology and Application to Coastal Ecosystems*. Elsevier, Amsterdam. https://doi.org/10.1016/B978-0-12-374711-2.00906-2

[27] Noguès Q, Raoux A, Araignous E, Hattab T, Leroy B, Ben Rais Lasram F, Le Loc'h F, Dauvin J, Niquil N (2020) Cumulative effects of marine renewable energy and climate change on ecosystem properties: Sensitivity of ecological network analysis. *Ecological Indicators* 121:107128. https://doi.org/10.1016/j.

ecolind.2020.107128

[28] Orth JD, Fleming RM, Palsson BO (2010) Reconstruction and use of microbial metabolic networks: the core Escherichia coli metabolic model as an educational guide. *EcoSal plus* 4(1):10-1128.

[29] Polyak BT, Gryazina EN (2014) Billiard walk - a new sampling algorithm for control and optimization. *IFAC Proceedings Volumes* 47(3):6123-6128. https://doi.org/10.3182/20140824-6-ZA-1003.02312

[30] Price ND, Reed JL, Palsson BO (2004) Genome-scale models of microbial cells: evaluating the consequences of constraints. *Nature Reviews Microbiology* 2(11):886-897. https://doi.org/10.1038/nrmicro1023

[31] R Core Team (2024) R: A Language and Environment for Statistical Computing. *R Foundation for Statistical Computing*. Vienna, Austria. https://www.R-project.org/

[32] Raftery AE, Lewis SM (1995) The number of iterations, convergence diagnostics and generic Metropolis algorithms. *Practical Markov Chain Monte Carlo* 7(98):763-773.

[33] Smith RL (1984) Efficient Monte-Carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research* 32:1296–1308. https://doi.org/10.1287/opre.32.6.1296

[34] Turchin V (1971) On the computation of multidimensional integrals by the Monte-Carlo method. *Theory of Probability & Its Applications* 16(4):720–724. https://doi.org/10.1137/111608

[35] Van den Meersche K, Soetaert K, Van Oevelen D (2009) xsample(): An R Function for Sampling Linear Inverse Problems. *Journal of Statistical Software* 30:1-15. https://doi.org/10.18637/jss.v030.c01

[36] van Oevelen D, Van den Meersche K, Meysman FJR, Soetaert K, Middelburg JJ, Vézina AF (2010) Quantifying Food Web Flows Using Linear Inverse Models. *Ecosystems* 13:32-45. https://doi.org/10.1007/s10021-009-9297-6

[37] Wilkinson D (2011) Gibbs sampler in various languages. *Darren Wilkinson's blog*. https://darrenjw.wordpress.com/2011/07/16/gibbs-sampler-in-various-languages-revisited/. Accessed 24 January 2024.

[38] Zhang Y (2013) Urban metabolism: A review of research methodologies. *Environmental pollution* 178:463-473. https://doi.org/10.1016/j.envpol.2013.03.052