**Introducing BioStoch: A Python Library for Simulating Biological Systems**

BioStoch is a Python library encompassing various deterministic and stochastic methods employed in systems biology for simulating biological systems. Among the deterministic methods offered are the Forward Euler method and the Runge-Kutta method (RK4). Additionally, BioStoch provides three stochastic methods, namely the Stochastic Simulation Algorithm (also known as the Gillespie Algorithm), the Tau-Leaping Algorithm, and the Chemical Langevin Equation method.

**Introduction**

In the realm of systems biology, understanding the intricate connections between individual components at various levels, ranging from molecule-molecule interactions to cell-cell communication, is paramount. Exploring how these components influence each other forms a central aspect of research in this field. Alongside experimental endeavors aimed at elucidating these systems, mathematical descriptions and computational approaches offer valuable insights. Among these methods, computer simulations stand out as particularly powerful, often providing a cost-effective and time-efficient means of studying complex biological systems.

Simulation methods, while immensely useful, adopt a different perspective compared to experimental approaches. Rather than tracking individual entities over time, such as molecules within a system, they focus on the overall population dynamics. These simulations operate under the assumption of spatial homogeneity, positing uniform concentrations of molecules throughout the volume of interest. For instance, in the study of a reactor or a biological cell, it is assumed that the concentrations of various species remain consistent across the entire volume, without localized gradients.

Mathematical modeling and simulation serve as indispensable tools in systems biology for several reasons. Foremost among these is the ability to validate our assumptions about the system under investigation. Such theoretical models provide a reliable framework for testing different scenarios, especially in cases where experimental approaches may be impractical or prohibitively expensive. Moreover, they offer a means to explore and understand complex biological phenomena in a controlled and systematic manner.

**Terminology**

* **Model:** A model serves as an abstract representation of the system under investigation, typically formulated mathematically. In simple cases, these models can be solved analytically. However, in most instances, numerical methods are employed for solving the model, which serve as approximations to analytical methods. The accuracy of numerical solutions improves as smaller steps are used.
* **Simulation:** Simulation involves the numerical solution of a system, aiming to replicate the behavior of the real system over time. This approach is known as simulation. As the British statistician George Box wrote in 1976, "All models are wrong, some are useful," signifying that all models are abstractions of reality, simplified versions of the real systems they represent.
* **Deterministic simulation:** Deterministic models rely on precise inputs and produce consistent outputs for a given set of inputs. These models operate under the assumption that the future can be predicted with certainty based on the current state.
* **Stochastic simulation:** Stochastic simulation involves simulating a system with variables that can change stochastically (randomly) with individual probabilities. Realizations of these random variables are generated and incorporated into a model of the system.

**Simulation methods**

The choice of simulation method is determined by the aspects of the natural system in which we are interested, as well as our mathematical and computational resources.

* **Molecular dynamic simulations:** While they offer the highest level of accuracy, they are also the most computationally expensive models. These models can be categorized into two sub-groups:

1. Quantum methods: These methods evaluate the wave functions at the level of individual electrons and are necessary when quantum effects become significant.
2. Classical methods: They solve the classical equations of motion to deterministically simulate the motion of molecules.

* **Stochastic methods:** This category includes several classes:

1. Spatial discrete stochastic
2. Discrete stochastic:

* Master equation
* Stochastic Simulation Algorithm (SSA)
* Tau-leaping
* Chemical Langevin Equation

1. Continuous stochastic

* **Continuous deterministic:**

1. Random:

* Quantum molecular dynamics
* Stochastic simulations

1. Deterministic methods:

* Classical molecular dynamics
* Ordinary differential equations

Our focus here is on discrete stochastic methods, which begin with the chemical master equation and utilize various numerical methods to approximate it. These methods are commonly employed in systems biology to simulate molecular populations over relatively long time periods while still considering them as composed of discrete units.

Using these methods, we do not track each molecule or momentum, thus losing the spatial aspect. It must also be assumed that the system of interest is stochastic.

The effects of intrinsic noise have generally been ignored in biology, both conceptually and computationally (mathematically). These models utilize state variables as real numbers representing the concentrations of molecules and do not account for noise. They can be considered accurate when we are interested in the mean dynamics of a large number of molecules, large enough that we do not need to concern ourselves with individual molecules but can approximate them as populations.

Assuming the system below represents a biological system with two species that can convert to each other with rate constants k1​ and k2​:

Simulating such a system could, in principle, begin with the spatial information and velocity of each molecule in the system, followed by running the simulation and tracking collisions between molecules and the resulting interactions. However, this method is typically too complex and computationally expensive when dealing with a large number of molecules or when simulating the dynamics of the system over a long period of time.

Instead of this approach, one can ignore spatial information and simply keep track of the number of molecules of each species. In this case, the system is assumed to be homogeneous, meaning that the molecules of each species are uniformly spread throughout the system, and the system is in thermal equilibrium with a constant volume.

Since spatial information about the molecules in the system is not accessible, at each step or time of the simulation, we think in terms of the probability of a reaction taking place based on the current state of the system. This leads to the formulation of the "Chemical Master Equation (CME)," which consists of a set of ordinary differential equations (ODEs), with one ODE equation for each possible state of the system.

represents the probability distribution of the system being in state x at time t. Essentially, it describes the likelihood of finding the system in a particular state xx at a given time t.

The propensity function of reaction j at state xx. It quantifies the likelihood of reaction j occurring at state x per unit time.

The state-change vector associated with reaction j, describing the change in the number of molecules when reaction j occurs. This vector accounts for the stoichiometry of the reaction.

The probability distribution of the system in state x − vj​ at time t. It represents the probability of finding the system in state x − vj after the occurrence of reaction j.

The total number of different types of reactions in the system, capturing the diversity of processes affecting the system's evolution.

Stochastic Simulation Algorithm (SSA):

The direct solution of the Chemical Master Equation (CME) entails calculating the full set of Ordinary Differential Equations (ODEs) to obtain the probability distribution across all possible states for each time (t). However, this approach is computationally demanding, particularly for vast biological systems housing millions or billions of molecules. Instead of directly solving the CME, we can approximate it by sampling from the probability distribution at each time (t). In essence, this involves computing realizations of the state vector {t, x(t)} such that the likelihood of a particular realization being computed reflects the corresponding probability dictated by the CME. This method is known as the Stochastic Simulation Algorithm (SSA) or Gillespie's algorithm. Initially conceived by Joseph L. Doob and others around 1945, it was formally presented by Dan Gillespie in 1976 and gained popularity in 1977 when he efficiently and accurately simulated chemical or biochemical systems of reactions using limited computational resources.

To derive the SSA from the CME, we require the following quantities:

The probability that no reaction takes place in the time interval [t, t + .

The probability that the next reaction occurs in the interval [t + , t and is of type j.

We assume that what occurs over [t, t + τ) is independent of what happens over [t + τ, t + τ + dt), implying that each step in the simulation is independent and unaffected by the previous step. Consequently, we have:

p0()(1-)

Which gives after rearrangement:

With the limit as dτ approaches zero, we obtain:

Now, shifting our focus to , we have:

Here, represents the discrete distribution of the reaction indices j = 1, 2, …, m while describes the density function of an exponential random variable with parameter , characterizing the distribution of waiting time to the next event.

The SSA simulator in “biostoch” library can be used as follow:

Implementation of SSA using biostoch

Tau-leaping Algorithm:

The Stochastic Simulation Algorithm (SSA) ensures exactness in producing statistics from the Chemical Master Equation (CME). However, its practical utility diminishes when reactions occur frequently due to its dynamic update of the entire system after each reaction occurrence. A solution to this inefficiency is to update the system after a certain number of reactions occur within a fixed time interval, denoted as τ. While in some modified versions of the tau-leaping algorithm, τ can be dynamically calculated based on system characteristics, here we consider a fixed interval length.

The tau-leaping algorithm involves updating each reaction in the system based on the number of occurrences within the time interval τ, given x(t). The update equation is:

To ensure accuracy, the time interval τ should be sufficiently small, such that only a relatively small number of reactions take place within it. This restriction ensures that the propensity functions do not change drastically after each update. During the interval [t, t+), remains constant, allowing the number of type-j reactions to fire to be calculated using a Poisson distribution, denoted as In general, a Poisson random variable p with parameter λ > 0 takes possible values {0, 1, 2, …}.

The tau-leaping algorithm can be implemented using “biostoch” as follows:

Tau-leaping implementation using biostoch

Chemical Langevin Equation (CLE):

if the number of reactions kj in the tau-leaping approach is sufficiently large for each j, can be approximated by a random distribution with the same mean and variance. To achieve this approximation, each in the tau-leaping algorithm can be replaced with , where Zj​ are independent normal(0, 1) random variables. This transition alters the entries of the state vector XX from natural numbers (representing the number of molecules) to real numbers (denoting molecule amounts in arbitrary units or simply concentration). Consequently, the tau-leaping algorithm can be rewritten as follows, known as the CLE algorithm:

Here, Y(t) represents the state vector in the CLE algorithm, and denotes the noise term.

CLE implementation using biostoch

Reaction Rate Equations (RREs)

If the system is sufficiently large, the right side of the Chemical Langevin Equation (CLE) – the noise term – tends to become negligible compared to the left side. Consequently, one can disregard this term, reducing the equation to a set of ordinary differential equations (ODEs):

This simplification allows for the utilization of various deterministic methods to dynamically simulate the system. Utilizing Euler forward integration, we can derive the RRE model with respect to time, resulting in the following equation:

=

Here, Y(t) represents the deterministic state vector containing concentrations of each species in the system, while vj corresponds to reaction velocities, and aj​ denotes the reaction rate.

Euler implementation using biostoch

The Runge-Kutta equations (RK4)

The Runge-Kutta method, particularly the fourth-order variant (RK4), stands as a notable advancement over the Euler method in numerical integration, especially when dealing with systems characterized by nonlinearity or rapidly changing dynamics. Unlike the Euler method, which approximates the derivative at a single point to update the solution, RK4 calculates intermediate points within each step, leading to greater accuracy and stability.

Mathematically, RK4 employs a weighted average of function evaluations at different points within a time step, resulting in a more accurate estimation of the next state. Specifically, RK4 computes four intermediate slopes using function evaluations at the initial point and three additional points along the trajectory within the time step. These slopes are then weighted and combined to determine the final slope used to update the solution. This multi-step process inherently captures more information about the system's behavior within each time step, leading to improved accuracy compared to the simplistic single-step approach of Euler's method.

Mathematically, we can describe RK4 as follow:

* Calculate the first slope k1​ using the initial condition:
* Use k1​ to estimate the value of Y at the midpoint of the interval:
* Calculate the second slope k2 at the midpoint:
* Use k2​ to make a better estimate of Y at the midpoint:
* Calculate the third slope k3​ at the midpoint:
* Use k3 to make another estimate of Y at the endpoint of the interval:
* Finally, calculate the fourth slope k4​ at the endpoint:
* Combine the slopes to update the solution:

Runge-Kutta implementation using biostoch