IEEE Life Sciences Letters

Received 30 November 2015; revised 18 July 2016; accepted 1 October 2016.
Date of publication 6 February 2017; date of current version 7 March 2017.

Digital Object Identifier 10.1109/LLS.2017.2652448

GillesPy: A Python Package for Stochastic Model Building and Simulation

JOHN H. ABEL[1,2], BRIAN DRAWERT[3], ANDREAS HELLANDER[4], AND LINDA R. PETZOLD[3]

[1]Department of Systems Biology, Harvard Medical School, Boston, MA 02115 USA
[2]Department of Chemical Engineering, University of California, Santa Barbara, CA 93106 USA
[3]Department of Computer Science, University of California, Santa Barbara, CA 93106 USA
[4]Department of Information Technology, Division of Scientific Computing, Uppsala University, SE-751 85 Uppsala, Sweden

This work was supported in part by NIH under Grants R01GM096873-01 and R01EB014877, in part by DOE under Grant DE-SC0008975, and in part by the Institute for Collaborative Biotechnologies, U.S. Army Research Office, under Grant W911NF-09-0001.
(John H. Abel and Brian Drawert contributed equally to this work.)

ABSTRACT GillesPy is an open-source Python package for model construction and simulation of stochastic biochemical systems. GillesPy consists of a Python framework for model building and an interface to the StochKit2 suite of efficient simulation algorithms based on the Gillespie stochastic simulation algorithms. To enable intuitive model construction and seamless integration into the scientific Python stack, we present an easy-to-understand action-oriented programming interface. Here, we describe the components of this package and provide a detailed example relevant to the computational biology community.

INDEX TERMS Biological systems, open-source software, stochastic systems, systems biology.

I. INTRODUCTION

STOCHASTICITY has recently been recognized as an essential feature of cellular processes. Extrinsic noise may be caused by fluctuations in the physical environment or properties of the individual cell (e.g., cell age or size), and may be captured in dynamic models through time-varying noise in model parameters. Intrinsic noise is caused by the low copy numbers of genes, transcripts, and proteins, and spatial inhomogeneity within the cell. Intrinsic noise in particular has gained attention, due to its essential role in cellular processes such as genetic toggle switches, noise-driven oscillation, cell polarization, and cell population dynamics [1]–[5].

Deterministic ordinary differential equation (ODE) models of biochemical processes are useful and accurate in the high-concentration limit, but fail to accurately capture stochastic cellular dynamics, as they assume spatial homogeneity and continuous biomolecule concentration. To address the issue of quantized concentrations, we can replace deterministic ODEs with a continuous-time discrete-space Markov process, with the probability density of the system governed by the chemical master equation (CME). The CME is expensive to solve directly due to the curse of dimensionality, and more often, Gillespie's stochastic simulation algorithm (SSA) method is used to generate a trajectory that is a statistically correct sample of the probability density. An ensemble of trajectories can be generated via a Monte Carlo method to form a basis for statistical analysis, a process that can be computationally intensive for large systems or large ensembles.

The original SSA has been extended to include methods for more efficient exact and approximate simulations, including the optimized-direct method, composition-rejection method, and τ-leaping [6]–[9]. For a recent review, see [10]. Many of these improved methods have been distributed in the popular StochKit2 software package [11]. StochKit2 provides an efficient C-implementation of algorithms for discrete stochastic simulation with a command-line interface. StochKit2 models must be created in StochML format, and simulation trajectories are returned as CSV files.

With its wide variety of numerical libraries and statistical packages, Python has become one of the most commonly used and effective languages in computational biology. In order to optimize computational biology workflow and simplicity in working with stochastic model building and simulation, we have created the *GillesPy* package. GillesPy combines a Python-based model construction toolkit with the computational efficiency of the StochKit2 C-based SSAs. GillesPy builds on StochKit2, and provides many enhancements to the model construction and simulation workflows. The model construction toolkit allows simple setup and parameterization of the CME. For stochastic simulation, StochKit2 automatically inspects the model to be simulated and selects the most efficient SSA formulation

based on the model size (direct method for small and composition-rejection method for large; see [11] for a complete description). For deterministic simulation, GillesPy uses the StochKitODE solver from the StochSS software suite, which uses the CVODES solvers from the Sundials software package [12]. The GillesPy package encapsulates the entire process in Python, for seamless integration with other computational packages or statistical analysis. GillesPy also supports the import of SBML models.

In this letter, we describe the features and use of GillesPy, and provide a relevant example for efficient simulation and numerical stability analysis of a genetic toggle switch.

II. MECHANISTIC MODELING OF BIOLOGICAL SYSTEMS

GillesPy is designed to simulate dynamic mechanistic models of biochemical networks. In mechanistic models, the dynamic behavior of the system is built up from individual interactions between biochemical species. Typically, this is a finite number of species and reactions interacting probabilistically in a well-stirred domain. This is in contrast to empirical models, which focus on mathematical functions based on external characteristics such as dose-response curves. In these models, the external dynamic behavior of the system is captured by a set of mathematical equations, in a "top-down" approach, and as a result has reduced predictive power [13]. Mechanistic models, unlike empirical models, can be used to predict the system's future behavior, or its behavior under perturbation. Mechanistic models differ in that they may be used to probe hypotheses about the underlying reaction pathways rather than simply the in–out behavior of a system. Often, complicated reaction pathways may be simplified through the use of Michaelis–Menten or Hill-type kinetic equations. GillesPy does allow for Michaelis–Menten and Hill propensity functions, as these types of functions are quite common. However, we caution that the developer of a model must always be careful to check the validity of the assumptions of these model reduction functions [14]–[16].

III. GillesPy DESIGN

GillesPy is designed to follow the "pythonic" object-oriented principles, and thus, biochemical models in GillesPy are constructed in an object-oriented fashion. To construct a new model, you define a new class that extends a base model *gillespy.Model*. The model constructor defines the parameters, species, and reactions of the biochemical system by associating the like named objects (*Parameter*, *Species*, and *Reaction*) from the *gillespy* Python package. Once defined, users interact with their model by instantiating instances objects of the model. These objects may be used to generate simulation trajectories of the biochemical systems through their *.run()* method. This command calls the StochKit2 C-solvers to simulate the provided model. StochKit2 selects the computationally optimal algorithm for simulating each model.

After a simulation is completed, the resulting trajectories are returned as Numpy arrays into the Python interface, where the data are available for processing by the large library of scientific Python tools. Model fitting, statistical analysis, or visualization is not directly handled within GillesPy, as there are numerous mature software packages for these purposes commonly used in the scientific Python community (e.g., DEAP [17] for model fitting through evolutionary algorithms, pandas [18] for statistical analysis, or Matplotlib [19] for visualization). The following section demonstrates the building and simulation of a simple and biologically relevant example.

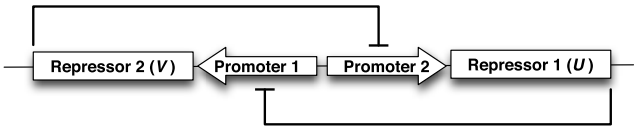

FIGURE 1. Schematic showing the genetic switch model from [2].

IV. EXAMPLE: A BISTABLE GENETIC SWITCH

Bistable stochastic genetic switches have been shown to play important roles in cellular differentiation [20]. As the system has two equilibria, deterministic simulations fail to accurately capture the random switching between states. Fig. 1 is a diagram of the genetic toggle switch, showing how each of the two promoters expresses a gene that is the inhibitor for the opposite promoter. Here, we demonstrate using GillesPy to simulate a bistable switch from [2]. The deterministic equations comprising this switch are

$$\frac{dU}{dt} = \frac{a_1}{1+V^{\beta}} - U \tag{1}$$

$$\frac{dV}{dt} = \frac{a_2}{1+U^{\gamma}} - V \tag{2}$$

where U and V are corepressor concentrations. Here, the parameters a_1 and a_2 are synthesis rates of U and V, respectively. Parameters β and γ represent the cooperativity of each repressor. We create a stochastic model from these equations by converting them to four stochastic reaction channels: synthesis and degradation of U and V, respectively

$$\emptyset \xrightarrow{\frac{\alpha_1}{1+[V]^{\beta}}} U, \quad U \xrightarrow{\mu[U]} \emptyset \tag{3}$$

$$\emptyset \xrightarrow{\frac{\alpha_2}{1+[U]^{\gamma}}} V, \quad V \xrightarrow{\mu[V]} \emptyset. \tag{4}$$

We note that this simple model does not explicitly differentiate between transcription and translation of U and V.

Constructing this model in Python begins with creating a model object by inheriting from GillesPy's model class

```
class BistableToggleSwitch(gillespy.Model).
```

We create and add parameters within this object by

```
a1 = gillespy.Parameter('a1', expression=4)
self.add_parameter([a1, ...])
```

and equivalently add species and reactions. We then simulate this model by invoking

```
model = BistableToggleSwitch()
results = model.run()\!.
```

A single simulation showing U and V populations using both stochastic and deterministic solvers over the course of hundreds is shown in Fig. 2.

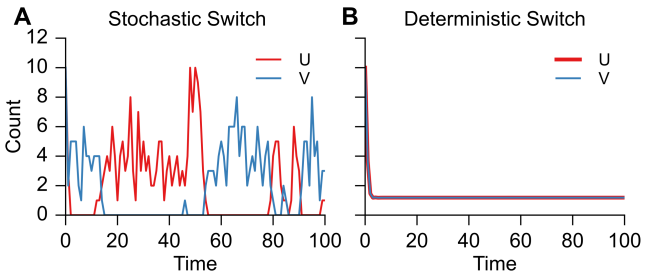

FIGURE 2. Stochastic and deterministic simulation of the genetic toggle switch with GillesPy. Bistability is evident in the stochastic model of this switch (with $\beta = \gamma = 2.0$).

Here, the difference between stochastic and deterministic results is visually evident. For identical initial populations of U and V, the deterministic system evolves to a metastable state with equal U and V. For nonidentical initial conditions, the system evolves to a stable state where only the species with a higher initial population are produced (not shown). Meanwhile, the stochastic simulation of this system shows dynamic switching between U-dominated and V-dominated states, as seen in [2], regardless of the choice of initial condition. Spontaneous switching between stable states is never observed under deterministic conditions. The full code for this model and simulation is available at http://github.com/GillesPy/gillespy/ as GeneticToggleSwitch.ipynb.

Exploring model dynamics via a parameter sweep is a common task in computational biology. To perform a simple parameter sweep, we allowed our model to accept parameter arrays and assign these values to parameters upon initialization of the model object. Thus, GillesPy was used to algorithmically generate, simulate, and analyze model dynamics without forcing the user to manually create or parameterize different models. Instead, the user can simply use a loop (or a parallel loop) to parameterize a dynamically generated model, perform simulations, analyze simulation trajectories, and return summary statistics.

As noted in [2], the mono- or bistability of the switch system is dependent on the values of cooperativity parameters (β and γ), and promoter strengths (αs). Equations 1 and 2 yield stable solutions where $(dU/dt) = (dV/dt) = 0$. For low cooperativity, this results in a single monostable steady state. Increased cooperativity results in an increased nonlinearity and sigmoidal switch-like behavior of the promoter [21]. The sigmoidal shape of promoter kinetics results in two stable states, and one metastable state for the system. For this example, we investigated how the cooperativity parameter affects bistability by repeatedly generating and simulating the stochastic model with parameters $\beta = \gamma \in [0, 4.0]$. Fig. 3 demonstrates the results of this process for varying β and γ in increments of 0.1. For our example, $\alpha_1 = \alpha_2 = 10.0$, and a bimodal distribution of states first appeared at approximately $\beta = \gamma > 1.3$, indicating that this is where the transition to bistability occurs. This critical bistability threshold would increase with a lower α, as lower (or unbalanced) promoter activity results in less switch-like behavior.

V. PARALLEL PROCESSING WITH MOLNs

GillesPy has been integrated with, and is distributed with MOLNs, a cloud computing platform for computational systems biology that focuses on reproducibility and scalability [22]. MOLNs allow computational scientists to easily create compute clusters using cloud computing resources. It further provides methods for automatically parallelizing workflows for computing large ensembles of trajectories or for conducting global parameter sweeps. This is combined with a facility for efficient postprocessing of the resulting stochastic trajectories, designed to maximize data locality by distributing the code to worker nodes where the data are generated. This facility is built to utilize the simple-to-use programming paradigm of MapReduce [23]. The user interface for this system is an interactive Web-based environment, the IPython Notebook [24]. These computable documents combine code, equations, narrative text, visualizations, images, as well as other media. Integration with MOLNs provides

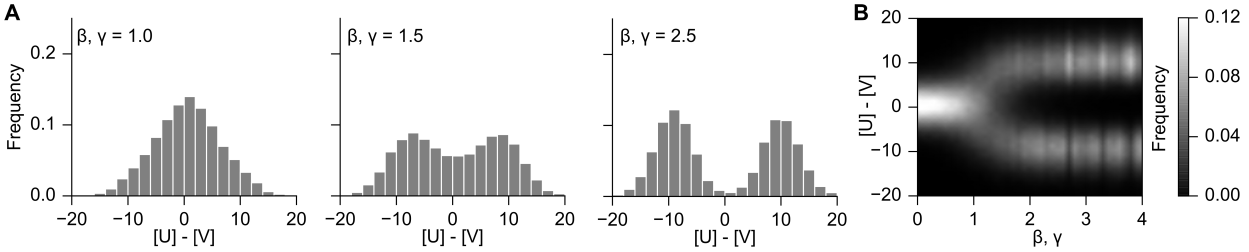

FIGURE 3. Investigating bistability through β and γ. (a) Histograms of state ($U - V$) for three selections of cooperativity parameters β and γ. (b) Heatmap of state probability for a range of β and γ in increments of 0.1. A bimodal state distribution first appears when $\beta = \gamma > 1.3$. Simulations were performed for 25 000 s and sampled in increments of 1 s for each parameterization. Parallelization through IPython enables this computationally intensive simulation to be performed on an 8-CPU desktop machine in approximately 5 min.

GillesPy users with a simple and powerful interface to scale their computational workflows using public or private cloud computing infrastructures.

VI. CONCLUSION

GillesPy is an open source package for stochastic model building and simulation, and a Python interface to the StochKit2 solvers. GillesPy runs on Linux/Unix or Mac OS X. It is freely available under GPL version 3. Installation instructions and downloads are available at http://github.com/GillesPy/gillespy. We welcome both bug reports and requests for assistance on our Github page.

REFERENCES

[1] M. B. Elowitz, A. J. Levine, E. D. Siggia, and P. S. Swain, "Stochastic gene expression in a single cell," *Sci. Signal.*, vol. 297, no. 5584, pp. 1183–1186, 2002.
[2] T. S. Gardner, C. R. Cantor, and J. J. Collins, "Construction of a genetic toggle switch in *Escherichia coli*," *Nature*, vol. 403, no. 6767, pp. 339–342, 2000.
[3] C. H. Ko *et al.*, "Emergence of noise-induced oscillations in the central circadian pacemaker," *PLoS Biol.*, vol. 8, no. 10, p. e1000513, 2010.
[4] M. J. Lawson, B. Drawert, M. Khammash, L. Petzold, and T.-M. Yi, "Spatial stochastic dynamics enable robust cell polarization," *PLoS Comput. Biol.*, vol. 9, no. 7, p. e1003139, 2013.
[5] P. C. St. John, S. R. Taylor, J. H. Abel, and F. J. Doyle, III, "Amplitude metrics for cellular circadian bioluminescence reporters," *Biophys. J.*, vol. 107, no. 11, pp. 2712–2722, 2014.
[6] Y. Cao, H. Li, and L. Petzold, "Efficient formulation of the stochastic simulation algorithm for chemically reacting systems," *J. Chem. Phys.*, vol. 121, no. 9, p. 4059, 2004.
[7] Y. Cao, D. T. Gillespie, and L. R. Petzold, "Adaptive explicit-implicit tau-leaping method with automatic tau selection," *J. Chem. Phys.*, vol. 126, no. 22, pp. 1–27, 2007.
[8] D. T. Gillespie, "Approximate accelerated stochastic simulation of chemically reacting systems," *J. Chem. Phys.*, vol. 115, no. 4, pp. 1716–1733, 2001.
[9] A. Slepoy, A. P. Thompson, and S. J. Plimpton, "A constant-time kinetic Monte Carlo algorithm for simulation of large biochemical reaction networks," *J. Chem. Phys.*, vol. 128, no. 20, p. 05B618, 2008.
[10] D. T. Gillespie, A. Hellander, and L. R. Petzold, "Perspective: Stochastic algorithms for chemical kinetics," *J. Chem. Phys.*, vol. 138, no. 17, p. 170901, 2013.
[11] K. R. Sanft *et al.*, "StochKit2: Software for discrete stochastic simulation of biochemical systems with events," *Bioinformatics*, vol. 27, no. 17, pp. 2457–2458, 2011.
[12] R. Serban and A. C. Hindmarsh, "CVODES: The sensitivity-enabled ODE solver in SUNDIALS," in *Proc. ASME Int. Design Eng. Tech. Conf.*, 2005, pp. 257–269.
[13] A. K. Thakur, "Model: Mechanistic vs empirical," in *New Trends in Pharmacokinetics*. New York, NY, USA: Springer, 1991, pp. 41–51.
[14] M. J. Lawson, L. Petzold, and A. Hellander, "Accuracy of the Michaelis–Menten approximation when analysing effects of molecular noise," *J. Roy. Soc. Interface*, vol. 12, no. 106, p. 20150054, 2015.
[15] R. Grima, "Noise-induced breakdown of the Michaelis-Menten equation in steady-state conditions," *Phys. Rev. Lett.*, vol. 102, no. 21, p. 218103, 2009.
[16] P. Thomas, A. V. Straube, and R. Grima, "Communication: Limitations of the stochastic quasi-steady-state approximation in open biochemical reaction networks," *J. Chem. Phys.*, vol. 135, no. 18, p. 181103, 2011.
[17] F.-A. Fortin, F.-M. De Rainville, M.-A. G. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary algorithms made easy," *J. Mach. Learn. Res.*, vol. 13, pp. 2171–2175, Jul. 2012.
[18] W. McKinney, "Data structures for statistical computing in python," in *Proc. 9th Python Sci. Conf.*, vol. 445, pp. 51–56, Jun. 2010.
[19] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 99–104, 2007.
[20] J. E. Ferrell, Jr., "Self-perpetuating states in signal transduction: Positive feedback, double-negative feedback and bistability," *Current Opinion Cell Biol.*, vol. 14, no. 2, pp. 140–148, 2002.
[21] U. Alon, *An Introduction to Systems Biology: Design Principles of Biological Circuits*. Boca Raton, FL, USA: CRC Press, 2006.
[22] B. Drawert, M. Trogdon, S. Toor, L. Petzold, and A. Hellander, "MOLNs: A cloud platform for interactive, reproducible, and scalable spatial stochastic computational experiments in systems biology using PyURDME," *SIAM J. Sci. Comput.*, vol. 38, no. 3, pp. C179–C202, 2016.
[23] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.
[24] F. Pérez and B. E. Granger, "IPython: A system for interactive scientific computing," *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 21–29, 2007.