# GillesPy: A Python Package for Stochastic Model Building and Simulation

**John H. Abel**[†],

Department of Systems Biology, Harvard Medical School, Boston, MA 02115 USA and the Department of Chemical Engineering, University of California, Santa Barbara, CA 93106 USA

**Brian Drawert**[†],

Department of Computer Science, University of California, Santa Barbara, CA 93106 USA

**Andreas Hellander**, and

Department of Information Technology, Division of Scientific Computing Uppsala University, Uppsala, Sweden SE-751 85

**Linda R. Petzold**

Department of Computer Science, University of California, Santa Barbara, CA 93106 USA

## 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 (SSA). 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; stochastic systems; systems biology; open-source software

## 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

[†]Authors contributed equally to this work.

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 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 which 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 $\tau$-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, 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 work, 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, 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*, *Reaction*) from the *gillespy* Python package. Once defined, users interact with their model by instantiating instances objects of the model. These objects may be use 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 is 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.
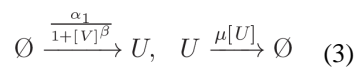
## IV. Example: A Bistable Genetic Switch

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. Figure 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:

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

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

where $U$ and $V$ are co-repressor concentrations. Here, the parameters $a_1$ and $a_2$ are synthesis rates of $U$ and $V$ respectively. Parameters $\beta$ and $\gamma$ 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.

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

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

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 100s is shown as Figure 2.

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 non-identical initial conditions, the system evolves to a stable state where only the species with a higher initial population is produced (not shown). Meanwhile, the stochastic simulation of this system shows dynamic switching between U-dominated and V-dominated states, as seen in Ref [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/asGeneticToggleSwitch.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 Ref. [2], the mono- or bistability of the switch system is dependent on the values of cooperativity parameters ($\beta$, $\gamma$), and promoter strengths ($a$s). Eqns. 1 and 2 yield stable solutions where $\frac{dU}{dt} = \frac{dV}{dt} = 0$. 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 $\beta = \gamma \in [0, 4.0]$. Figure 3 demonstrates the results of this process for varying $\beta$ and $\gamma$ in increments of 0.1. For our example, $a_1 = a_2 = 10.0$, 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 $a$, 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 allows 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 post-processing of the resulting stochastic trajectories, designed to maximize data locality by distributing the code to worker nodes where the data is 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 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.

## Acknowledgments

## References

1. Elowitz MB, Levine AJ, Siggia ED, Swain PS. Stochastic gene expression in a single cell. Sci Signal. 2002; 297(5584):1183.

2. Gardner TS, Cantor CR, Collins JJ. Construction of a genetic toggle switch in Escherichia coli. Nature. 2000; 403(6767):339–342. [PubMed: 10659857]

3. Ko CH, Yamada YR, Welsh DK, Buhr ED, Liu AC, Zhang EE, Ralph MR, Kay SA, Forger DB, Takahashi JS. Emergence of noise-induced oscillations in the central circadian pacemaker. PLoS Biol. 2010; 8(10):e1000513. [PubMed: 20967239]

4. Lawson MJ, Drawert B, Khammash M, Petzold L, Yi T-M. Spatial Stochastic Dynamics Enable Robust Cell Polarization. PLoS Comput Biol. 2013; 9(7):e1003139. [PubMed: 23935469]

5. St John PC, Taylor SR, Abel JH, Doyle FJ III. Amplitude Metrics for Cellular Circadian Bioluminescence Reporters. Biophys J. 2014; 107(11):2712–2722. [PubMed: 25468350]

6. Cao Y, Li H, Petzold L. Efficient formulation of the stochastic simulation algorithm for chemically reacting systems. J Chem Phys. 2004; 121(9):4059. [PubMed: 15332951]

7. Cao Y, Gillespie DT, Petzold LR. Adaptive explicit-implicit tau-leaping method with automatic tau selection. J Chem Phys. 2007; 126(2007):1–27.

8. Gillespie DT. Approximate accelerated stochastic simulation of chemically reacting systems. J Chem Phys. 2001; 115(4):1716–1733.

9. Slepoy A, Thompson AP, Plimpton SJ. A constant-time kinetic Monte Carlo algorithm for simulation of large biochemical reaction networks. J Chem Phys. 2008; 128(20)

10. Gillespie DT, Hellander A, Petzold LR. Perspective: Stochastic algorithms for chemical kinetics. J Chem Phys. 2013; 138(17):170901. [PubMed: 23656106]

11. Sanft KR, Wu S, Roh M, Fu J, Lim RK, Petzold LR. StochKit2: software for discrete stochastic simulation of biochemical systems with events. Bioinformatics. 2011; 27(17):2457–2458. [PubMed: 21727139]

12. Serban R, Hindmarsh AC. CVODES: the sensitivity-enabled ODE solver in SUNDIALS," in. Proc 2005 ASME Int Des Eng Tech Conf. 2005:257–269.

13. Thakur AK. Model: mechanistic vs empirical," in. New trends Pharmacokinet Springer. 1991:41–51.

14. Lawson MJ, Petzold L, Hellander A. Accuracy of the Michaelis-Menten approximation when analysing effects of molecular noise. J R Soc Interface. 2015; 12(106)

15. Grima R. Noise-induced breakdown of the Michaelis-Menten equation in steady-state conditions. Phys Rev Lett. 2009; 102(21):218103. [PubMed: 19519139]

16. Thomas P, Straube AV, Grima R. Communication: limitations of the stochastic quasi-steady-state approximation in open biochemical reaction networks. J Chem Phys. 2011; 135(18):181103. [PubMed: 22088045]

17. Fortin F, Rainville D. DEAP: Evolutionary algorithms made easy. J Mach Learn Algorithms. 2012; 13:2171–2175.

18. McKinney W. Data Structures for Statistical Computing in Python. Proc 9th Python Sci Conf. 2010; 1697900:51–56.

19. Hunter JD. Matplotlib: A 2D graphics environment. Comput Sci Eng. 2007; 9(3):99–104.

20. F JE Jr. Self-perpetuating states in signal transduction: positive feedback, double-negative feedback and bistability. Curr Opin Cell Biol. 2002; 14(2):140–148. [PubMed: 11891111]

21. Alon U. An introduction to systems biology: design principles of biological circuits CRC press. 2006

22. Drawert B, Trogdon M, Toor S, Petzold L, Hel-lander A. MOLNs: A Cloud Platform for Interactive, Reproducible, and Scalable Spatial Stochastic Computational Experiments in Systems Biology Using PyURDME. SIAM J Sci Comput. 2016; 38(3):C179–C202. [PubMed: 28190948]

23. Dean J, Ghemawat S. MapReduce Commun ACM. 2008; 51(1):107.

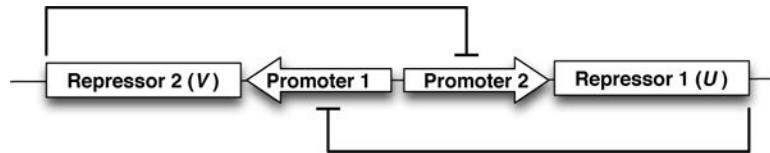24. Pérez F, Granger BE. IPython: A system for interactive scientific computing. Comput Sci Eng. 2007; 9(3):21–29.

**Fig. 1.**
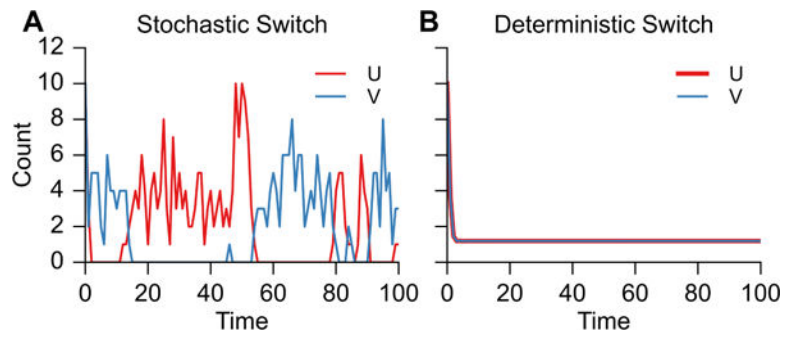A schematic showing the genetic switch model from Gardner et al. [2].

**Fig. 2.**
Stochastic and deterministic simulation of the genetic toggle switch with GillesPy. Bistability is evident in the stochastic model of this switch (with $\beta = \gamma = 2.0$).
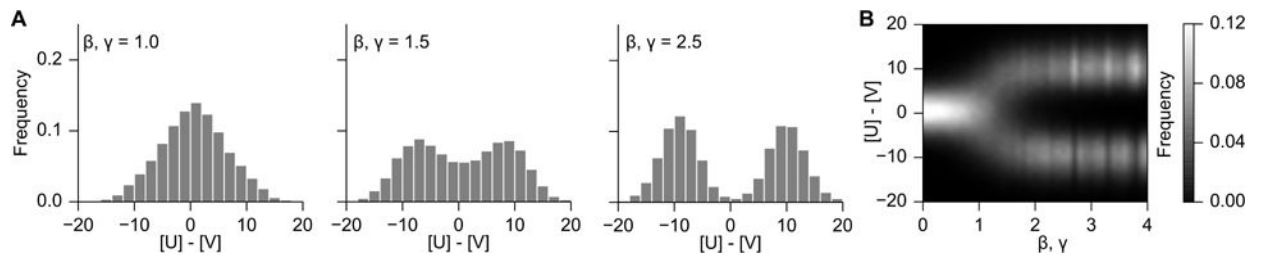
**Fig. 3.**

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