Quantitative Biology
DOI 10.1007/s40484-014-0036-4

RESEARCH ARTICLE

Construction, visualization, and analysis of biological network models in Dynetica

Derek Eidum[1,†], Kanishk Asthana[1,†], Samir Unni[1], Michael Deng[1] and Lingchong You[1,2,3,*]

[1] Department of Biomedical Engineering, Duke University, Durham, NC 27708, USA
[2] Center for Systems Biology, Duke University, Durham, NC 27708, USA
[3] Center for Genomic and Computational Biology, Duke University, Durham, NC 27708, USA
* Correspondence: you@duke.edu

Received October 30, 2014; Revised January 1, 2015; Accepted January 4, 2015

Mathematical modeling has become an increasingly important aspect of biological research. Computer simulations help to improve our understanding of complex systems by testing the validity of proposed mechanisms and generating experimentally testable hypotheses. However, significant overhead is generated by the creation, debugging, and perturbation of these computational models and their parameters, especially for researchers who are unfamiliar with programming or numerical methods. Dynetica 2.0 is a user-friendly dynamic network simulator designed to expedite this process. Models are created and visualized in an easy-to-use graphical interface, which displays all of the species and reactions involved in a graph layout. System inputs and outputs, indicators, and intermediate expressions may be incorporated into the model via the versatile "expression variable" entity. Models can also be modular, allowing for the quick construction of complex systems from simpler components. Dynetica 2.0 supports a number of deterministic and stochastic algorithms for performing time-course simulations. Additionally, Dynetica 2.0 provides built-in tools for performing sensitivity or dose response analysis for a number of different metrics. Its parameter searching tools can optimize specific objectives of the time course or dose response of the system. Systems can be translated from Dynetica 2.0 into MATLAB code or the Systems Biology Markup Language (SBML) format for further analysis or publication. Finally, since it is written in Java, Dynetica 2.0 is platform independent, allowing for easy sharing and collaboration between researchers.

Keywords: mathematical modeling; systems biology; synthetic biology; quantitative biology; gene circuits

INTRODUCTION

Over the past several decades, mathematical modeling has become an important tool in biological research. Due to the rapid expansion of biological knowledge, kinetic modeling has become a realistic goal, particularly for experimentally well-characterized systems.

Kinetic models have been essential for the progress of the fields of systems and synthetic biology, where scientists have successfully applied these models to explain many biological phenomena. For example, researchers recently constructed a whole cell kinetic model of the bacterium *Mycoplasma genitalium*, which was then used to predict phenotypic features of the bacterium [1]; kinetic modeling and synthetic circuits were recently used to successfully examine the role of stochastic pulses in regulating the stress response in bacteria [2], quantitatively understand quorum sensing in bacteria [3] and create genetic oscillators [4].

A kinetic model essentially represents a mathematical integration of existing data and mechanisms on a particular system, and is useful in a number of ways. A kinetic model can be used to test the consistency in the experimental data or mechanisms [5], provide mechanistic explanations for counter-intuitive observations [6], to facilitate the formulation of experimentally testable hypotheses [7] or to provide insight into emergent properties, such as robustness [3,8,9], which may be

[†] These authors contributed equally to this work

otherwise difficult to grasp intuitively. Kinetic models are often simulated using various numerical algorithms. Because this process is computationally intensive, model simulations using those algorithms are usually done using modeling software.

Many of the latest annotated kinetic models submitted to the BioModels Database [10] use either COPASI, MATLAB, or both for modeling or simulation needs. Although these pieces of software are powerful for model creation and simulation, they have a high learning threshold. This high learning threshold makes it difficult for beginning modelers with little or no programming experience to start modeling gene regulatory networks. To alleviate this problem and facilitate modeling, many GUI programs have been created, such as OpenAlea [11], CellDesigner [12], VCell [13] and E-Cell [14]. However these modeling programs have some limitations. For example, OpenAlea does not cater to general modeling needs and is specialized for solving plant modeling problems. CellDesigner allows GUI modeling of biochemical networks, but it requires an external mathematical package (COPASI) for simulating networks or fitting parameters. Both E-Cell and VCell allow simulation, GUI modeling, and provide very high functionality, but they have a high learning threshold, making them hard to intuitively use for beginners. Many of the other programs are generally available only as plugins and require some other main package or software for either simulating or modeling a network. There are also proprietary tools available such as Monolix [15], which allow both modeling and simulation capabilities in a single user friendly package, but these are not openly accessible, hampering the facilitation of teaching and research.

Therefore, there is a need for an open, user-friendly, and GUI-based software package with a low learning threshold for both simulating and modeling synthetic and natural gene regulatory networks. Dynetica 2.0 caters to this need. To increase the user friendliness and utility of Dynetica 2.0 over that of version 1.0 [16], we have incorporated many new features, such as an improved GUI, expression variables, modular organization, support for import and export of models in the form of MATLAB code and Systems Biology Markup Language (SBML) models, improved sensitivity analysis, parameter searching, and many bug fixes.

To allow greater adoption and usage of Dynetica 2.0 by the synthetic and systems biology community, we have released the source code of Dynetica 2.0 at under the GPL Version 2 license.

RESULTS

A general overview of the design paradigm of Dynetica is shown in Figure 1. Models in Dynetica 2.0 are represented by the substances and reactions that make up the system, as well as the parameters and expressions that govern them. Models are drawn in a graph layout and can be compartmentalized into individual modules. These modules can be copied, saved, and imported independently, thus facilitating the construction of large systems from basic subcomponents.

Simulations may be run using any of the following algorithms: Runge-Kutta (Fourth Order, Fixed Time Step), Runge-Kutta-Fehlberg (Fourth Order, Variable Time Step), Gillespie algorithms (both First Reaction and Direct Method implementations), Gibson algorithm, and the Euler-Maruyama method for stochastic differential equations.

Additionally, Dynetica 2.0 provides a number of useful tools for analyzing the behavior of models. Sensitivity analysis allows the user to examine how the time courses or characteristic metrics of a system change with perturbations in the system's parameters or initial conditions. Repeated stochastic simulations allow the user to visualize the distribution of a system's substance values across many trials. Parameter searching perturbs the system's parameter values in order to match the system's output to some user-specified behaviors.

Finally, Dynetica 2.0 supports the import and export of models in SBML format, and provides tools for exporting models to either deterministic or stochastic MATLAB scripts, allowing the user to perform any analyses not covered by Dynetica.

In comparison to the previous version, the areas of improvement in Dynetica 2.0 can be summarized in four segments: interface enhancements, sensitivity analysis support, genetic parameter searching, and import/export functions.

Interface

The three primary interface enhancements in Dynetica 2.0 are expression variable support, modular system construction, and import/export support.

Expression Variables

An expression variable is an entity in Dynetica 2.0 that can be represented by a mathematical or Boolean function of substance concentrations, parameters, and time. Expression variables can be used in reaction kinetics equations just like substance concentrations or parameter values. Because of this flexibility, they can be used as time-dependent system inputs, intermediate expressions, and output variables or indicators.

Modules

When building large or complex systems in Dynetica, it

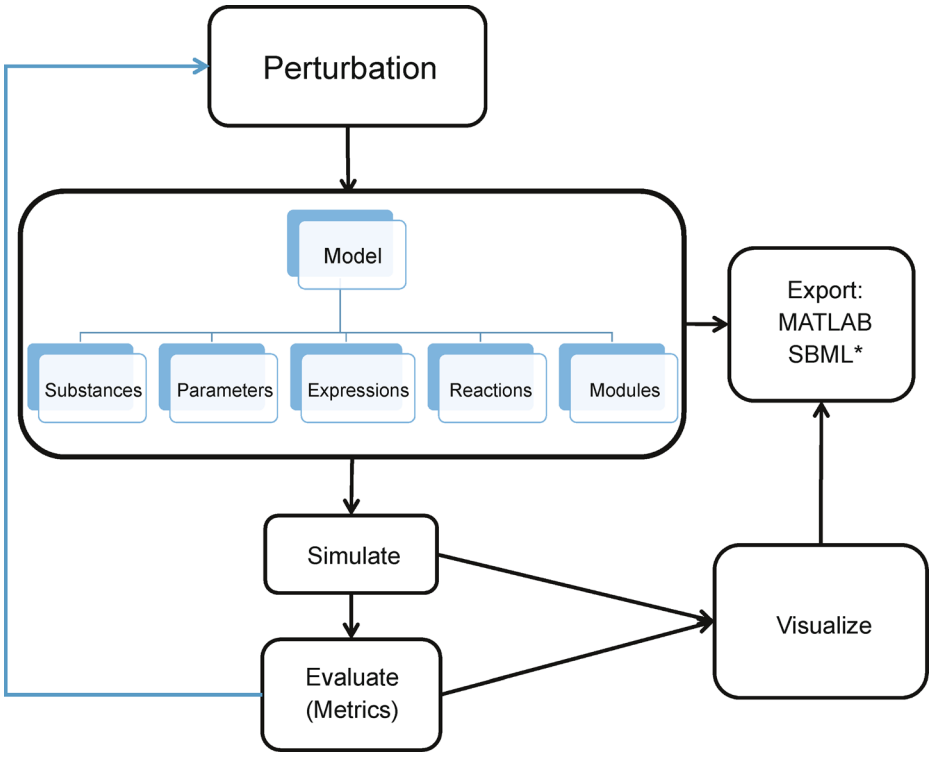

Figure 1. The design paradigm of Dynetica 2.0. The user specifies the module components: the substances, reactions, parameters, and expression variables. This module representation is independent of the algorithm used to simulate it. Dynetica 2.0 provides additional analysis tools which measure the behavior of the system in the form of metrics, which are calculated independent of how the simulation is performed. Depending on the type of analysis performed, these metrics may lead to or be measured as the result of some perturbation in the system parameters or initial conditions. Additionally, the model representation may be imported from or exported to the SBML format, and can generate simple MATLAB simulation scripts.

often becomes difficult to keep track of the various components and how they interact. One way to mitigate this problem is by taking advantage of the modular nature of many of these large networks. Although the total number of components may be large, most components only interact with a small subset of all of the components. The large networks tend to form small sub-networks, which then interact with one another via a handful of interconnects.

Dynetica 2.0 allows users to take advantage of this phenomenon by creating such sub-networks, called "modules." Figure 2, displaying a model for programmed altruistic death, exemplifies how Dynetica modules can be harnessed to enhance conceptual simplicity. The substances comprising the cheater and cooperator modules, on the bottom right and bottom left corners of the network graph, respectively, have each been isolated into their own module. Another advantage of such functionally defined modules is that they can be easily reused across or within models, speeding up new model construction. Further details regarding the use of modular systems in Dynetica can be found in section 1 of the Supplemental Materials.

Import and export

A major improvement in Dynetica 2.0 over the previous version is its ability to export models in SBML or MATLAB. The earlier version of Dynetica only allowed models to be stored in the native DYN format used by Dynetica. This made it difficult for models created in Dynetica to be further extended using more advanced modeling software, as they would have to be recreated in the new modeling platform. This new feature allows models created in Dynetica to be imported by other software packages that recognize the SBML format or MATLAB code.

In addition, Dynetica 2.0 allows the user to merge existing models with each other or add one model to another as a module. These features greatly facilitates creation of new models because it allows existing models to be reused or extended. Moreover, these new features allow the user to quickly create and test synthetic circuits by combining circuit elements from a library. Another advantage of using this new feature is that the initial MATLAB or SBML code for a model can be generated much more quickly and robustly, since the user will not

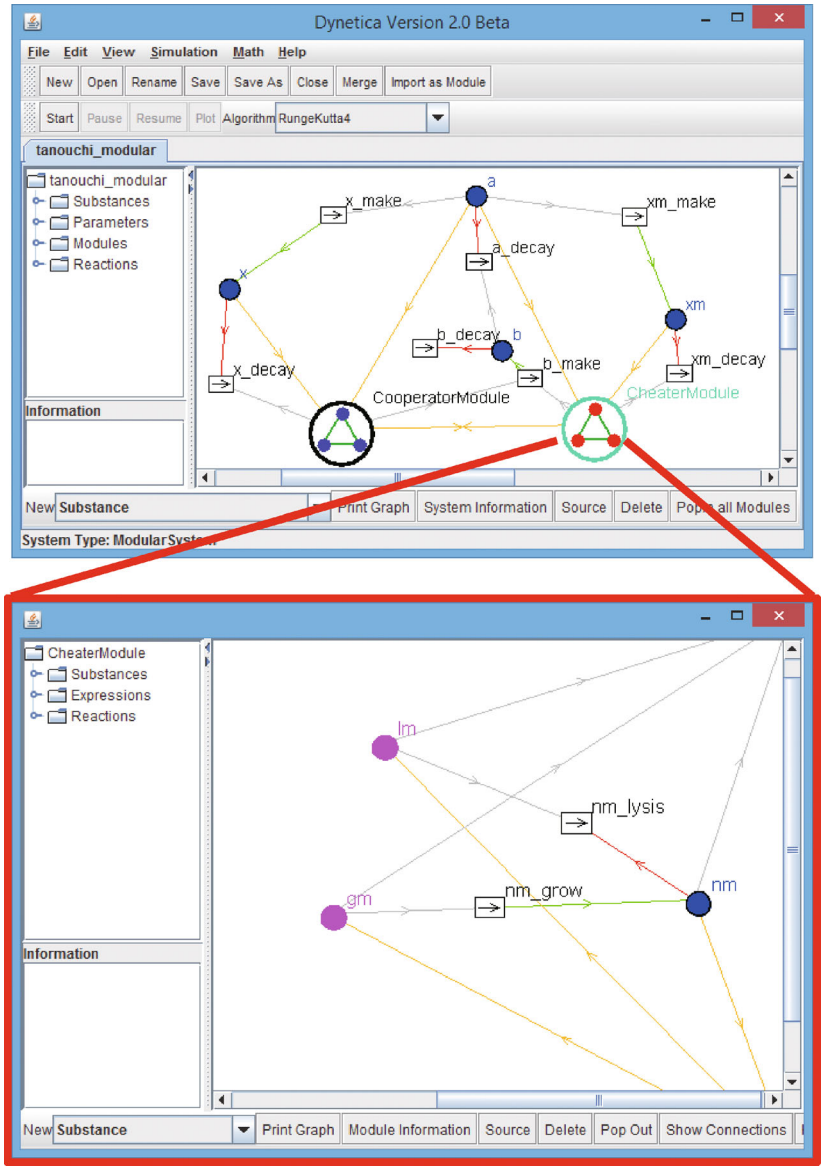

Figure 2. The module feature of Dynetica's UI. The upper window represents the entire cheater-cooperator system, while the lower window shows just the cheater module. Each triangle with a colored dot at each vertex and contained in a black circle corresponds to one user-defined module. The lines extending from those circles in the top window correspond to the lines in the bottom window that extend to its edges. The arrows on the lines indicate the relationship between the connected components. The component the arrow faces towards uses the component the arrow faces away from in its definition. When there are multiple lines connecting one module to another, they are all collapsed into one line in the whole system view (for example, the bottom line in the top window). In such a case, there may be arrows pointing in both directions, indicating that individual components from the two connected module both use components from the other in their respective definitions.

have to rewrite the entire code when the model is modified or extended. Moreover, parameter values and constants are also preserved during the merger.

Sensitivity Analysis

Metrics

Metrics are the key units used to analyze the behavior of a system in Dynetica 2.0. A metric is a value that describes some characteristic of a single time course. These are used in both sensitivity analysis and parameter searching.

The metrics implemented in Dynetica 2.0 include:

- Final, min, and max value;
- Range;
- Maximum rate;
- Time to reach a user-specified fraction of steady state value;

- Area under curve;
- Peak frequency in the power spectrum;
- Correlation coefficient when compared to another substance or expression variable in the same system.

Simulation types

Dynetica 2.0 provides the user the ability to explore how the behavior of the system varies with changes in either parameter values or initial substance concentrations, through its sensitivity analysis tool. The user selects a parameter or substance from the model to act as the independent variable, and defines any number of metrics to evaluate. The simulation is run multiple times over a range of values for the independent variable. To visualize the change in behavior as the independent variable is perturbed, the value of each metric can be plotted as a function of the parameter or initial concentration value, or alternatively, the time courses for each trial can be plotted on the same axes.

Deterministic simulations

One type of sensitivity analysis supported by Dynetica uses deterministic simulations. An example analysis, using the ePop system [17], is shown in Figure 3A. Since the simulations are deterministic, there only needs to be one for each combination of parameter values.

The time courses produced by a deterministic sensitivity analysis can be programmatically analyzed to produce a more succinct plot. In Figure 3B, the cell count (n) time courses shown in Figure 3A are analyzed for their peak frequencies, which is plotted as a function of the parameter w.

Stochastic simulations

The sensitivity analysis tools may be used with stochastic algorithms as well. However, the results may prove less meaningful because the change in system behavior may be due to random noise rather than the deviations in the independent variable. To examine how stochastic systems respond to changes in parameters or initial concentrations, Dynetica 2.0 provides a tool for performing repeated stochastic simulations for each value of the independent variable, as demonstrated with the Rb-E2F system analysis displayed in Figure 3C. Both the time courses and the distribution of values can be viewed for user-specified time intervals, as shown in Figure 3D.

Dynetica 2.0 also provides the ability to perform sensitivity analyses using stochastic algorithms and generate distributions of values, such as the Rb-E2F system analysis seen in Figure 3C.

Summary of Sensitivity Analysis Types

In summary, the following types of sensitivity analysis are supported in Dynetica 2.0:

- Deterministic
  - o Basic (time courses);
  - o Processed according to one of the metrics in the section of metrics.
- Stochastic (repeated to obtain a distribution)

Parameter Search

Dynetica 2.0 provides tools for searching the parameter space of a system in order to optimize user-defined objectives. This is useful for when the user knows the desired behavior of a system and wants to find a set of parameters for which the system closely matches this behavior. The search may be performed over all or any subset of the system's parameters.

The user defines the desired behavior of the system by specifying a number of target objectives, which the algorithm will attempt to match. The user selects a metric and specifies a goal for that metric's value, which can be to minimize, maximize, or target its value to a specific number. Target objectives can also be given different weights, with higher weighted objectives having a greater influence on the search than lower ones.

The parameter search is done using a genetic algorithm, a computational technique that imitates natural selection to heuristically find an optimal solution given a set of constraints. The initial population is made up of individual parameter vectors. In order to calculate the evolutionary fitness of each individual parameter vector, its parameters are applied to the system and the simulation is run. The vector then receives a fitness score between 0 and 100 based on how well the simulation matches the user-defined metrics. At each generation, high scoring parameter vectors survive and reproduce, while low scoring vectors die off, maintaining a constant population size. The details of the implementation of this algorithm can be found in the supplementary materials.

To run a parameter search, the user must specify which parameters will be perturbed. All others will remain fixed at their user-defined values. The user then must specify the target objectives for the algorithm to optimize, each of which consists of a metric, a goal for that metric, and a relative weight. The user must also specify the simulation duration, population size, and maximum number of generations for which the algorithm will run. The user can additionally define a stopping threshold, which is a fraction specifying what score is needed to cause the search to complete. For example, the default value of 0.05 indicates that a fitness score of 95 or higher (out of 100) is sufficient to stop the search. Setting this value to 0 means

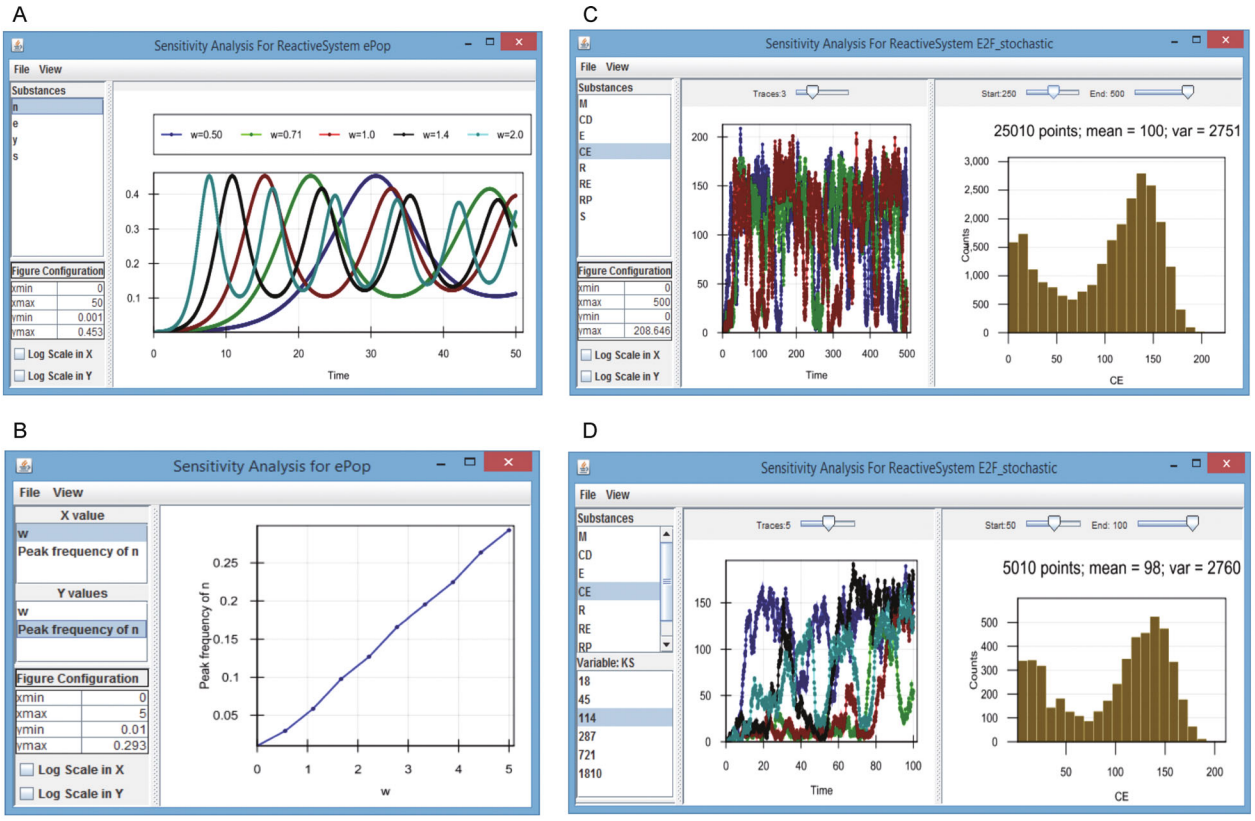

Figure 3. Conducting basic sensitivity analysis or stochastic simulations in Dynetica. (A) A sensitivity analysis conducted on the ePop circuit [17] using deterministic simulations. The parameter '*w*' is varied from 0.5 to 2.0, resulting in a change in the length of the transient response and the oscillation frequency at steady state for the substance '*n*' (the cell count). Frequency is not the only metric that Dynetica's sensitivity analysis tools support. The other difference between the time courses can be measured as well, using the "time to steady state" metric. (B) The results of a more sophisticated sensitivity analysis, also using the ePop circuit. Rather than simply showing the time courses for the various substances comprising the system under simulation, the frequencies of oscillation are shown. This is particularly useful for a system such as that in Figure 3B, where one of the few differences between the time courses is their oscillation frequency. (C) A stochastic simulation of the Rb-E2F system [18]. The histogram on the right shows the distribution of Cyclin E concentrations across all simulations within the specified time window (250 to 500 ms). (D) The results of a stochastic sensitivity analysis on the Rb-E2F system. The stochastic differential equation (SDE) algorithm was used, with the intrinsic noise scale set to 0.25 and K_S varied from 18 to 1810, on a log scale. The displayed bimodal distribution is for $K_S = 721$. The distribution of R_E values was collected using only the second halves of the time courses, once they had reached steady state. The options panel on the far left allows the user to select a substance or expression variable to plot, which value of the independent variable to look at, and the figure window options. The center panel shows the resulting time courses. The right panel shows the distribution of values across the repeated simulations, as well as the mean and variance. The slider bars at the top allow the user to specify the time window of interest.

the search will not stop unless the target objectives are matched exactly or the maximum number of generations is reached.

By default, each parameter vector is created by random normal perturbations around the user's defined value. This allows the algorithm to use the user-given values as an initial guess. However, the user may also chose to create the initial population by selecting uniformly random numbers between the parameter's specified minimum and maximum values. This may produce better results by exploring new areas of the parameter space. Note, however, that if this option is selected, it is important to specify realistic constraints for the parameter's minimum and maximum values. Figure 4 shows a typical example of using the parameter search tool to identify a new parameter set for a specific target function.

Dynetica 2.0 also features a tool for searching the parameter space of a system when the desired behavior is not a feature of a single simulation, but of the dose response or sensitivity analysis curves. In addition to the inputs above, the user must specify a parameter or initial substance concentration to use as the independent

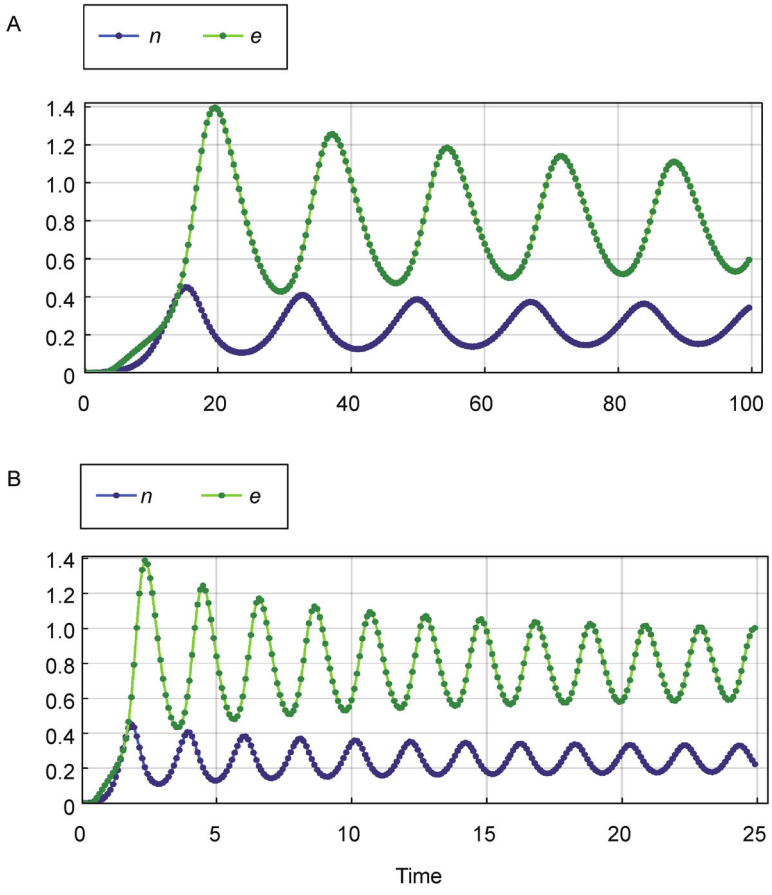

Figure 4. The resulting time course plots before (A) and after (B) running a parameter search on the ePop circuit [17] for the parameters *w* and α. Three objective functions were used: match the peak frequency of *n* to 0.5 Hz (with a weight of 0.6), match the maximum value of *n* to 1 (with a weight of 0.2) and match the maximum value of *e* to 2 (with a weight of 0.2). The high priority frequency objective was matched almost exactly; however, this comes at the expense of the lower priority objectives.

variable, as well as a range of values to scan over. The same genetic search algorithm is used; however, rather than using a single metric value, the fitness score is calculated as a function of the shape of curves when the specified metrics are plotted as against the independent variable.

For each metric selected, there are six target functions that can be used to calculate the fitness score. The range of the sensitivity plot can be minimized or maximized, resulting in the metric being minimally or maximally responsive to perturbations in the independent variable, respectively. The user may also select to maximize or minimize the difference between the first two local extrema, or the first local extremum and the steady state value. The user may also select to maximize the difference from the initial value to the first local extremum. Lastly, the user may choose for the response curve to be maximally linear, in which case a linear regression will be case the score will be based on the coefficient of determination. Figure 5 illustrates the use of the search algorithm to identify a parameter set to generate a strong biphasic response in a synthetic gene circuit.

Further details on Dynetica 2.0's Parameter Search feature can be found in section 2 of the Supplemental Materials.

DISCUSSION

Significant interface and experience improvements have been made to Dynetica, in the form of the expression variable/module features, as well as import/export functionality. These enhancements ease the process of constructing and evaluating models in Dynetica 2.0, as well as the collaboration process when working with researchers who utilize other pieces of dynamic network simulation software.

Dynetica 2.0 has added support for much more sophisticated analyses of stochastic simulations. Stochas-

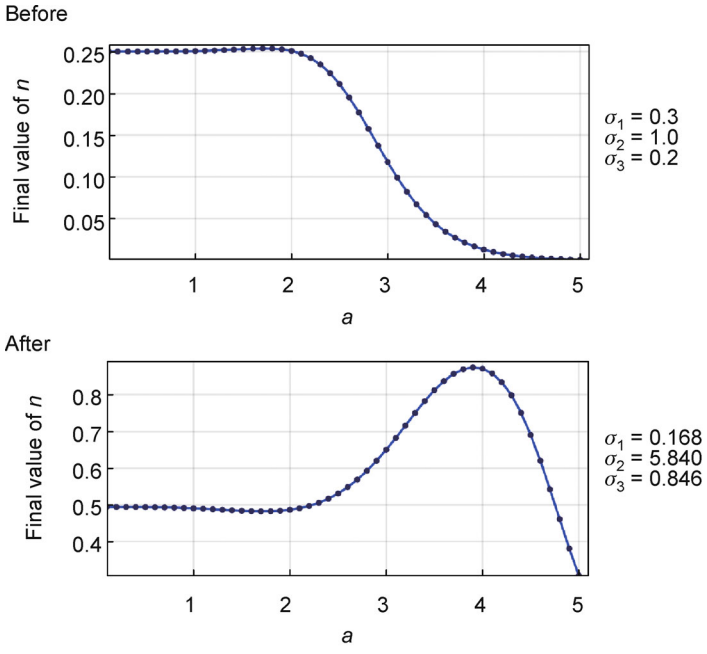

Figure 5. The dose response of the final population size of a system before and after performing a dose response parameter search. The figure demonstrates an example of the Eagle effect, a seemingly paradoxical phenomena in which increasing antibiotic dosage within a certain range causes bacterial populations to recover faster [19]. This can be shown by an increase in the final population size when plotted against antibiotic concentration, as evidenced by the increase in *n* for small values of the parameter *a* [20]. In the original model, this increase is very small. After performing a parameter search to maximize the difference between the initial value and first local extremum, the increase is much stronger, suggesting that these parameter values lead to a stronger Eagle effect.

tic simulations can be repeated with a fixed parameter set to generate outcome distributions, or with a varying parameter set to accomplish sensitivity analyses. For small systems, stochastic analyses can provide realistic insight into laboratory phenomena. For those systems that are too large for stochastic algorithms to be practical, sensitivity analyses can also be conducted using deterministic algorithms. In addition, a genetic search algorithm can be used to determine a set of parameter values that would produce the desired value of a specified metric.

Some of the limitations of Dynetica 2.0 are its inability to directly model spatial gradients and multiple compartments. However, many basic modeling tasks do not require the added complexity of multiple compartments, computationally intensive simulations, and spatial modeling. Despite these limitations, we believe that Dynetica 2.0 serves its purpose by providing a robust and easy to use modeling and simulation environment for non-programmers.

ACCESSING THE SOFTWARE

Dynetica is freely available to the research community.

(i) General users can access the binary code at: http://www.genome.duke.edu/labs/YouLab/software/dynetica/index.php.

(ii) Developers can access the source code at: https://github.com/youlab/dynetica.

SUPPLEMENTARY MATERIALS

The supplementary materials can be found with this article at DOI 10.1007/s40484-014-0036-4.

ACKNOWLEDGMENTS

The work is partially funded by the National Institutes of Health, the National Science Foundation, the Army Research Office, and the David and Lucile Packard Foundation. DE acknowledges the support for a Pratt Fellowship.

COMPLIANCE WITH ETHICS GUIDELINES

The authors Derek Eidum, Kanishk Asthana, Samir Unni, Michael Deng and Lingchong You declare that they have no conflict of interests.

This article does not contain any studies with human or animal subjects performed by any of the authors.

REFERENCES

1. Karr, J. R., Sanghvi, J. C., Macklin, D. N., Gutschow, M. V., Jacobs, J.

M., Bolival, B. Jr., Assad-Garcia, N., Glass, J. I. and Covert, M. W. (2012) A whole-cell computational model predicts phenotype from genotype. Cell, 150, 389–401

2. Locke, J. C. W., Young, J. W., Fontes, M., Hernández Jiménez, M. J. and Elowitz, M. B. (2011) Stochastic pulse regulation in bacterial stress response. Science, 334, 366–369
3. Pai, A., Tanouchi, Y. and You, L. (2012) Optimality and robustness in quorum sensing (QS)-mediated regulation of a costly public good enzyme. Proc. Natl. Acad. Sci. USA, 109, 19810–19815
4. Danino, T., Mondragón-Palomino, O., Tsimring, L. and Hasty, J. (2010) A synchronized quorum of genetic clocks. Nature, 463, 326–330
5. Ferrezuelo, F., Colomina, N., Palmisano, A., Garí, E., Gallego, C., Csikász-Nagy, A. and Aldea, M. (2012) The critical size is set at a single-cell level by growth rate to attain homeostasis and adaptation. Nat. Commun., 3, 1012
6. Tan, C., Smith, R. P., Srimani, J. K., Riccione, K. A., Prasada, S., Kuehn, M. and You, L. (2012) The inoculum effect and band-pass bacterial response to periodic antibiotic treatment. Mol. Syst. Biol., 8, 617
7. Yao, G., Tan, C., West, M., Nevins, J. R. and You, L. (2011) Origin of bistability underlying mammalian cell cycle entry. Mol. Syst. Biol., 7, 485
8. Wong, J. V., Li, B. and You, L. (2012) Tension and robustness in multitasking cellular networks. PLoS Comput. Biol., 8, e1002491
9. Wang, J., Li, C. and Wang, E. (2010) Potential and flux landscapes quantify the stability and robustness of budding yeast cell cycle network. Proc. Natl. Acad. Sci. USA, 107, 8195–8200
10. Li, C., Donizelli, M., Rodriguez, N., Dharuri, H., Endler, L., Chelliah, V., Li, L., He, E., Henry, A., Stefan, M. I., et al. (2010) BioModels Database: An enhanced, curated and annotated resource for published quantitative kinetic models. BMC Syst. Biol., 4, 92
11. Pradal, C., Dufour-Kowalski, S., Boudon, F., Fournier, C. and Godin, C. (2008) OpenAlea: a visual programming and component-based software platform for plant modelling. Plant Biol., 35, 751–760.
12. Funahashi, A., Morohashi, M., Kitano, H. and Tanimura, N. (2003) CellDesigner: a process diagram editor for gene-regulatory and biochemical networks. Biosilico, 1, 159–162
13. Moraru, I. I., Schaff, J. C., Slepchenko, B. M., Blinov, M. L., Morgan, F., Lakshminarayana, A., Gao, F., Li, Y. and Loew, L. M. (2008) Virtual Cell modelling and simulation software environment. IET Syst. Biol., 2, 352–362
14. Takahashi, K., Ishikawa, N., Sadamoto, Y., Sasamoto, H., Ohta, S., Shiozawa, A., Miyoshi, F., Naito, Y., Nakayama, Y. and Tomita, M. (2003) E-Cell 2: multi-platform E-Cell simulation system. Bioinformatics, 19, 1727–1729
15. Team, M. (2012) The Monolix software, Version 4.1. 2. Analysis of mixed effects models. LIXOFT and INRIA, http://www.lixoft.com/, March
16. You, L., Hoonlor, A. and Yin, J. (2003) Modeling biological systems using Dynetica—a simulator of dynamic networks. Bioinformatics, 19, 435–436
17. Marguet, P., Tanouchi, Y., Spitz, E., Smith, C. and You, L. (2010) Oscillations by minimal bacterial suicide circuits reveal hidden facets of host-circuit physiology. PLoS One, 5, e11909
18. Wong, J. V., Dong, P., Nevins, J. R., Mathey-Prevot, B. and You, L. (2011) Network calisthenics: control of E2F dynamics in cell cycle entry. Cell Cycle, 10, 3086–3094
19. Eagle, H. and Musselman, A. D. (1948) The rate of bactericidal action of penicillin in vitro as a function of its concentration, and its paradoxically reduced activity at high concentrations against certain organisms. J. Exp. Med., 88, 99–131
20. Tanouchi, Y., Pai, A., Buchler, N. E. and You, L. (2012) Programming stress-induced altruistic death in engineered bacteria. Mol. Syst. Biol., 8, 626

