Journal of Bioinformatics and Computational Biology

DIZZY: STOCHASTIC SIMULATION OF LARGE-SCALE GENETIC REGULATORY NETWORKS

Stephen Ramsey, David Orrell, and Hamid Bolouri*

Institute for Systems Biology
1441 North 34th Street, Seattle, Washington 98103-8904, United States

We describe Dizzy, a software tool for stochastically and deterministically modeling the spatially homogeneous kinetics of integrated large-scale genetic, metabolic, and signaling networks. Notable features include a modular simulation framework, reusable modeling elements, complex kinetic rate laws, multi-step reaction processes, steady-state noise estimation, and spatial compartmentalization.

Keywords: kinetics; stochastic; simulation; biochemical network; software

1. Introduction

Genetic regulatory networks (GRNs) control cellular state, form, and function. They are responsible for executing embryonic developmental programs, changing cellular state in response to signaling events, and controlling metabolic processes based on environmental conditions. Specific examples include the early cell specification process within the sea urchin embryo,[1] the control of galactose uptake in yeast,[2–4] and the pathogen-triggered immune response within a macrophage[5].

GRNs typically involve feedback interactions among multiple genes. For example, Figure 1 shows interactions among 36 genes involved in the specification of three cell types in early sea urchin embryos[1]. The expanded section highlights the complex feedback interactions in a small part of the network. Such multiple feedbacks make *Gedanken* understanding the behavior of larger GRNs virtually impossible. The situation is frequently more complex in adult organisms, where feedback loops intertwine genetic, metabolic, and signaling networks closely. Signaling and metabolic events change the state of a GRN, which in turn modifies the structure of the "upstream" signaling/metabolic network. For example, in the yeast galactose utilization pathway, complex interactions among the regulatory genes *GAL3*, *GAL4*, and *GAL80* control the synthesis of a handful of enzymes that regulate galactose metabolism[4]. In turn, imported galactose activates the regulatory protein Gal3p, which inhibits the repressive effect of the Gal80p protein on the *GAL* regulon. This is a positive feedback loop between the regulatory and metabolic networks, as shown in Figure 2. The prevalence and complexity of feedback loops necessitates

*To whom correspondence should be addressed. E-mail: hbolouri@systemsbiology.org

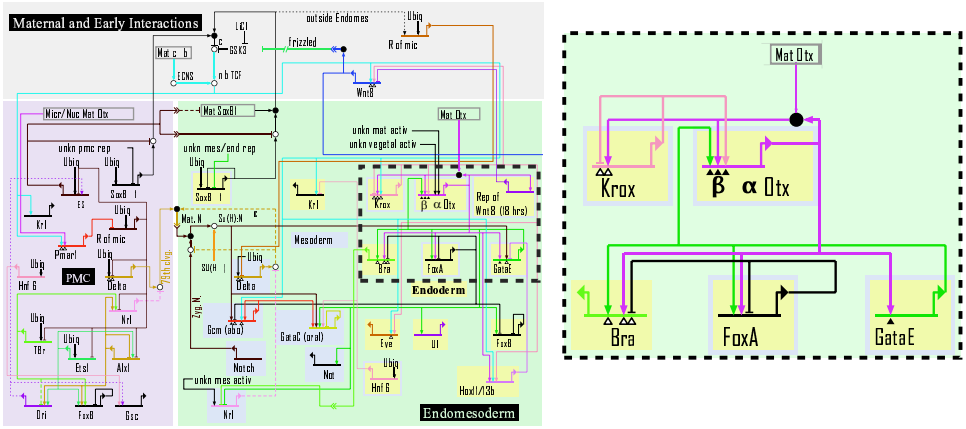

Fig. 1. Schematic diagram of 36 interacting genes underlying endomesoderm specification in early sea urchin embryos[1]. Each gene is represented by a horizontal line (denoting DNA) and a bent arrow (denoting the basal transcription apparatus). The portion of the horizontal line preceding the bent arrow represents the *cis*-regulatory region of the gene. Lines emanating from one gene and incident on another, denote transcriptional regulatory interactions. Some interactions involve signaling (indicated by a double chevron symbol) and/or protein interactions. The call-out region highlights some of the important feedback interactions in the endoderm. It is noted that additional information such as biochemical reaction rate constants and a mechanistic description of gene regulation are required in order to simulate the dynamics of the network depicted in the diagram[6].

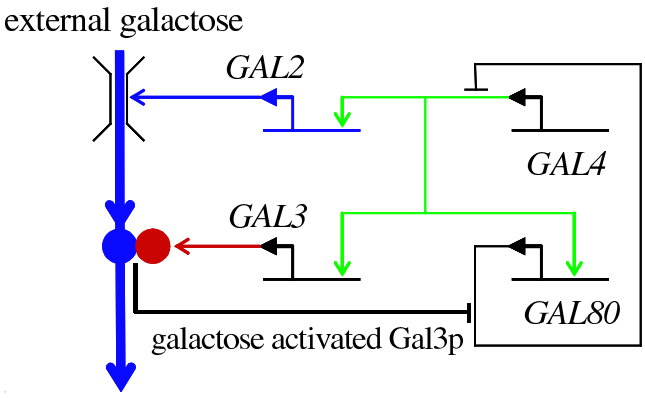

Fig. 2. Schematic diagram of the core genetic regulatory network of the galactose utilization pathway in yeast[4]. Only the galactose-import part of the metabolic pathway is shown (*GAL2* gene and thick blue arrows indicating galactose importation and processing). There is a feedback loop between the metabolic import of galactose, Gal3p activation, Gal80p inactivation, and Gal2p control of galactose importation. It is noted that additional information such as biochemical reaction rate constants and a mechanistic description of gene regulation are required in order to simulate the dynamics of the pathway depicted in the diagram[4].

computational modeling and simulation of large-scale networks that include genetic, metabolic, and signaling elements.

A complication in modeling large-scale GRNs is that the small numbers of molecules involved result in inherently high levels of stochastic noise in GRNs. Only two copies of each gene are transcribed in diploid organisms, and the number of mRNA molecules expressed by genes that code for transcription factors is typically small[7]. Stochastic noise in GRNs can result in complex patterns of cellular heterogeneity[8–10]. The effects and significance of such variability can be analyzed with stochastic simulations[11–17], but the computational cost of stochastic simulations often increases dramatically with model complexity.

The task of conducting stochastic, large-scale GRN simulations is further complicated because transcription and translation are highly complex processes whose kinetics we can only approximate. In Figure 3, we show some of the critical steps in

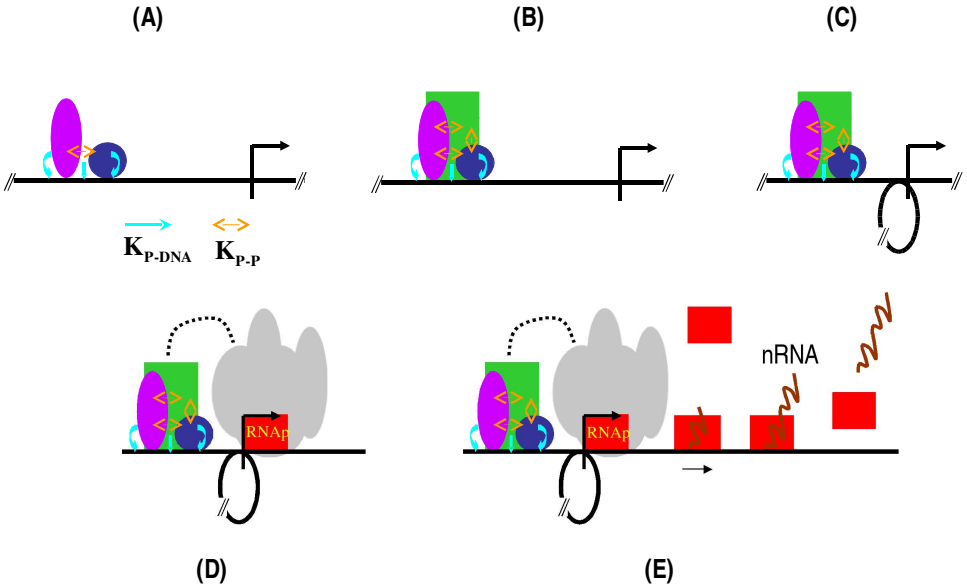

Fig. 3. Major steps in transcription: (A) One or more transcription factors diffuse along DNA and bind to specific sites. (B) Cooperative interactions recruit co-factors. (C) Physical DNA is re-structured (e.g., looping, chromatin remodeling). (D) Recruitment of the transcription complex and initiation of transcription. (E) Translocation on DNA, and initiation of further transcriptions.

eukaryotic transcription. The five steps depicted in this cartoon compress into single events, complex chemical interactions each involving tens of molecular species. The component interactions for most of these "events" are not fully understood and so cannot be modeled at the level of true chemical kinetics. Even if these details were fully known, the sheer number and complexity of interactions would make exact mechanistic simulation of all steps in gene transcription computationally pro-

hibitive for any more than a few genes. Thus, there is a pressing need to model transcription in terms of a series of approximate models. The approximate models abstract out the unknown detailed steps, but preserve the key known features of transcription and translation and faithfully capture intrinsic noise characteristics.

Consider the last step in Figure 3, the translocation of the transcription apparatus along DNA. For a typical mRNA molecule, this would involve approximately 10^3–10^4 individual, base-pair by base-pair DNA copying and translocation steps. A GRN modeler would often require only the times at which RNA molecules are released and not be interested in the intermediate steps between transcription initiation and transcript release. Yet, to understand the effect and role of transcriptional noise in network behavior, an accurate model of RNA production is needed.

Because of the large number of factors and co-factors that collectively regulate animal genes, modeling the formation of a transcription factor complex on DNA and the steps that lead to transcription initiation can involve thousands of elementary reaction steps. For a multi-gene network, the task of generating a model description would therefore be daunting even if all the individual steps were known.

Thus, analysis of the behavior of GRNs requires the modeling of

- Interactions of multiple (often large numbers of) genes.
- Metabolic / signaling networks that regulate and are regulated by the genes.
- Feedback interactions within and between the above networks, giving rise to complex nonlinear dynamics.
- Stochastic kinetics arising from the small numbers of molecules.
- Reduced models of transcription and translation that compress the hundreds of individual reaction steps involved in transcription and translation to a few steps, while retaining the above four critical features.

In the rest of this paper, we describe a modeling and simulation software program, Dizzy, which answers all the above needs and offers additional benefits.

2. Overview of the Dizzy software system

In this section, we give an overview of the major features of Dizzy, a software framework for modeling the dynamics of complex biochemical systems.

Modular simulation framework: Dizzy employs a modular design in which each simulator is a software unit that conforms to a simple, well-defined interface specification. The biochemical modeling semantics are separated from the description of how the dynamics is to be solved. This architecture facilitates an iterative model development cycle in which the model is analyzed using various simulation algorithms. The simulators implemented in Dizzy are described in Section 3.

Templates–reusable and hierarchical model elements: Dizzy's model definition language permits the definition of reusable, parameterized model elements called templates. This enables the construction of a prepackaged library of templates that can simplify the task constructing a complex model. The template feature is

further described in Section 4.

Complex kinetic rate laws: Dizzy enables the creation of reduced stochastic models containing reactions whose propensities may be expressions of arbitrary complexity, representing the average effect of underlying reaction steps that are in quasi-steady-state (QSS). This permits efficient approximate modeling of enzyme-catalyzed reactions and other processes for which the overall kinetic rate is more complicated than mass-action kinetics. This feature is further described in Section 5.

Multi-step and delayed reaction processes: Dizzy enables the simulation of complex multi-step processes such as elongation and translocation during transcription or translation, through two methods. One may define a "multi-step" reaction process, or a reaction process with an intrinsic, phenomenological time delay. The multi-step and delayed reaction features are described further in Section 6.

Estimation of steady-state stochastic noise: Dizzy provides a feature for estimating or calculating the steady-state stochastic fluctuations of the species in a biochemical model, requiring only the solution of the deterministic dynamics. The steady-state noise estimation feature is described further in Section 7.

Integrated, graphical, and portable software framework: Dizzy has several important software features including integration with external software tools, a graphical user interface (GUI), and a high level of portability. The software features of Dizzy are described in Section 8.

Many software tools are available for solving the deterministic and stochastic dynamics of complex biochemical networks[18–20]. A detailed overview of the most common algorithms for simulating biochemical kinetics is presented in Section 2 of the Supplementary Material. In Table 1, we compare some of the more widely-used simulation software tools against a specific list of simulation algorithms and features described above. To the best of our knowledge, Dizzy is the first software tool available that includes all of the features enumerated above. In addition, it includes novel implementations of the Gibson-Bruck and Gillespie Tau-Leap algorithms that are applicable to models with complex kinetic rate laws. At present, Dizzy is not able to explicitly model spatially inhomogeneous chemical species concentrations and transport phenomena such a diffusion. However, Dizzy permits partitioning of a model into distinct spatial compartments. Each compartment volume is treated as a spatially homogeneous, continuously well-stirred system.

3. Modular Simulation Framework

Dizzy has a modular design in which a *simulator* is a plug-in that conforms to a well-defined software interface. Each simulator is implemented as a self-contained unit that creates all of the internal data structures it needs to function. This allows for a variety of simulation techniques to be applied to a single model description, and for the clean separation of the simulation method from the model description. The model definition is focused on the biochemical semantics of defining chemical species and reactions. The technique and parameters for simulating the model are

Software Name	*Deterministic*	*Stochastic*	*Hybrid*	*Templates*	*Stoch QSS*	*Multi-step*
Dizzy	**ODE**	**GD, GB**	**TL**	**yes**	**yes**	**yes**
BioNetS[21]	ODE	GD	HR	yes		
BioPSI[22]		PRSS				
BSTLab[23]	ODE					
Cellerator[24]	ODE			yes		
BioCharon[25]	ODE/DE			yes		
Cellware[26]		GD	TL			
DBSolve[27]	ODE					
Dynetica[28]	ODE	GD		yes	yes	
E-Cell[29]	ODE	GD, GB	HR	yes		
ESS[30]		GD				
Genomic Object Net[31]	ODE/DE	HPN	HFPN	yes		limited
Gepasi[32, 33]	ODE			yes		
Jarnac[34]	ODE	(see SBW)		yes		
JSim[35]	ODE/DE					
JWS[36]	ODE					
KineticIT[17]			VB			
KINSOLVER[37]	ODE					
MCell[38]		GD				
Metabolizer[39]	ODE	GB				
MMT2[40]	ODE			yes		
NetBuilder[41]	DGAF					limited
SBW[34, 42–46]	ODE	GD, GB	TL			
SigTran[47]	ODE	GD, GB, FB			limited	
Simpathica[48]	ODE					
Stochastirator[49]		GD				
StochSim[50]		FB				
STOCKS1[51]		GD				
STOCKS2[14]		GB	TL		yes	
STODE[52]		GD			yes	
ULTRASAN[53]		SPN				
Virtual Cell[54]	ODE					

Table 1. This table compares simulation software programs. The simulation algorithms referenced in this table are described in Section 2 of the Supplementary Material. The column "Deterministic" lists the deterministic algorithms supported by the software. The abbreviations used are as follows: **ODE**, an ordinary differential equation solver; **ODE/DE**, a hybrid ODE solver that supports discrete events; **DGAF**, a directed graph of algebraic functions. The column "Stochastic" lists the stochastic simulation algorithms supported by the software. The abbreviations used are as follows: **GD**, Gillespie's Direct method[55]; **GB**, the Gibson-Bruck Next Reaction method[56]; **FB**, the Firth-Bray multi-state stochastic method[57]; **HPN**, a hybrid Petri net method[58]; **SPN**, a stochastic Petri net method[53]; **PRSS**, the Priami-Regev-Shapiro-Silverman π-calculus method[22, 59]. The column "Hybrid" lists the hybrid stochastic/deterministic or accelerated approximate stochastic simulation algorithms supported by the software. The abbreviations used are as follows: **TL**, the Gillespie Tau-Leap method[11, 13]; **HFPN**, a hybrid function Petri net method[31]; **HR**, the Haseltine-Rawlings method[12]; **VB**, the Vasudeva-Bhalla method[17]. The column "Stoch QSS" means that stochastic simulations with complex rate laws based on the quasi-steady-state (QSS) assumption (as described by Rao and Arkin[16]), are supported. It should be noted that BioNetS, Simpathica, ESS, and BioCharon are all available within the BioSPICE Dashboard software[60].

specified in the simulation controller, and do not require changes to the model.

Dizzy includes both stochastic and deterministic simulators. The stochastic simulators are discrete-event or multiple-event Monte Carlo algorithms; for information about the random number generator used, please refer to Section 3.2 of the Supplementary Material. The deterministic simulators model the dynamics as a set of ordinary differential equations (ODEs) which are solved numerically.

One benefit of this modular design is that one may use a deterministic ODE-based solver for optimization and parameter fitting, and switch to a stochastic simulation technique for exploring the stochastic dynamics, once the model parameters have been established. This modularity also simplifies the task of implementing a new simulator and integrating it into the system. In this section we describe the simulators available in our software system.

Dizzy includes an efficient implementation of a stochastic simulator based on Gillespie's Direct Method[55]. It uses the Monte Carlo technique to generate an approximate solution of the master equation[61] for chemical kinetics. In this method, simulation time is advanced in discrete steps, with precisely one reaction occurring at the end of each discrete time-step. Both the time steps and the reaction that occurs are random variables. The Direct Method requires recomputing all reaction probability densities after each iteration. The computational complexity of the method therefore increases linearly with the number of reactions. Furthermore, for sufficiently large simulation time, the total number of iterations can be prohibitively large for some systems. Nevertheless, for simple systems with small numbers of species and reactions, the Direct Method can be useful.

Dizzy also implements a stochastic simulator based on Gibson and Bruck's Next Reaction Method[56]. The computational cost of this Monte Carlo-type method scales logarithmically with the number M of reaction channels, in contrast with the Gillespie algorithm which scales linearly with M. We have implemented a tree traversal technique to analyze a rate expression for a chemical reaction that has a complex kinetic rate law, in order to ascertain the dependence of the rate expression upon the various chemical species in the model. This permits applying the Gibson-Bruck method to models that implement complex kinetic rate laws, such as those cases described in Section 5.

Two stochastic simulators based on Gillespie's Accelerated Approximate Method[11, 13] (here referred to as the "Tau-Leap" Method) have been implemented in Dizzy. The Tau-Leap Method is a stochastic process that approximately solves the chemical master equation, based on a controllable, dimensionless error parameter ϵ. A quantity known as the "maximum allowed leap time" τ is periodically computed based on ϵ, according to the Gillespie-Petzold formula[13]. If the time scale τ is less than a few times the inverse aggregate reaction probability density (where the exact threshold is configurable and only effects efficiency), a Gillespie Direct discrete event is carried out. In the case where τ exceeds the threshold time scale, a "leap" is performed. The number of times each reaction occurs during the time interval τ is generated using the Poisson distribution based on the reaction's proba-

bility density per unit time. The state of the system is updated to reflect the realized number of times each reaction occurs during the time interval τ. After each "leap" iteration, the maximum allowed leap time τ is recomputed.

Two versions of the Tau-Leap algorithm have been implemented, Tau-Leap Complex and Tau-Leap Simple. The Tau-Leap Simple algorithm is intended for simple models entirely composed of reaction channels with mass-action kinetics. The Tau-Leap Complex algorithm is a novel adaptation of Tau-Leap that is intended for use with models with complicated (e.g., enzymatic) rate expressions whose partial derivatives are very expensive to evaluate symbolically. In this method, the full symbolic Jacobian is stored and used at each iteration, in order to exploit the caching of evaluated, algebraically complex sub-expressions in the computation of the Gillespie-Petzold formula.

Using the Poisson distribution to model the number of times N_f a given reaction can occur during a time interval τ, has the disadvantage that the exponential tail allows for rare events in which the realized number of times a reaction occurs (generated from the distribution) is too great for the numbers of reactant species available; we call this "reactant exhaustion." This is not indicative of a failure of the algorithm *per se*, but of a need to decrease the time scale τ. Our implementation avoids this problem with a three-part adaptation. First, we calculate the expected change in all species concentrations assuming all reactions occur a number of times equal to the propensity times τ (this is just the mean value of the distribution). If any species is expected to be become negative as a result, the time scale τ is decreased in proportion to how early in the interval τ the species is expected to become exhausted. Second, for each reaction channel, we derive the maximum number of times a given reaction can occur before its reactants are exhausted; from this we derive the mean time scale t_{ex} for reactant exhaustion. If half this time scale is less than τ, then τ is decreased to $0.5t_{ex}$. Third, if despite the previous steps a resultant N_f causes reactant exhaustion, a new set of realized N_f values is generated; in practice such occurrences are extremely rare, so the risk of bias is minimal.

Dizzy also includes a deterministic simulator based on a fifth order Runge-Kutta ODE solver. Step size is adaptively controlled, based on a fourth order error estimation formula[62]. Both relative and absolute error tolerances may be independently specified, as well as the initial step size. Although Runge-Kutta is not state-of-the-art for high-accuracy integration, it is particularly useful for models in which a derivative function is discontinuous[62, 63].

Two additional deterministic simulators have been implemented based upon the odeToJava ODE solver package by Patterson and Spiteri[64, 65]. This package includes a Dormand-Prince fourth/fifth order solver[66] with adaptive step size control. It also contains a Runge-Kutta implicit-explicit ODE solver[67] that is useful for systems with a high degree of stiffness[63].

The performance of the deterministic and stochastic simulation algorithms described above has been benchmarked using a variant of the heat-shock response model for *Escherichia coli* proposed by Srivastava *et al.*[68] and adapted by Taka-

hashi *et al.* for benchmarking the performance of the E-Cell simulator[69]. This model includes a large separation of dynamical time scales, which is typical of complex biochemical networks. The text of the model is included in Section 4 of the Supplementary Material. The benchmark results for the heat-shock model are summarized in Table 2. The results show the efficiency of the Gibson-Bruck algorithm relative to

Algorithm	*Running time*
Gillespie Direct	514.6 ± 28.2 s
Gibson-Bruck	404.9 ± 10.7 s
Tau-Leap Simple	35.53 ± 0.43 s
Tau-Leap Complex	9.55 ± 0.15 s
Dormand-Prince 5/4 ODE	1.814 ± 0.020 s

Table 2. Benchmark results for solving the dynamics of the *E. coli* heat-shock model out to 100 seconds. The benchmark was carried out on a workstation with a single Intel Pentium 4 processor with a clock speed of 2.79 GHz and 1 GB of RAM. The operating system was Red Hat Linux release 8.0, kernel version 2.4.20. The Java Runtime Environment (JRE) used was the IBM Java Development Kit (JDK) version 1.4.1, standard edition. The heat-shock model was solved for 100 seconds of simulation time, using five algorithms: Dormand-Prince fourth/fifth ODE, Gillespie, Gibson-Bruck, Tau-Leap Complex, and Tau-Leap Simple. The ensemble size for the stochastic simulators was one. For each of the five simulators, the simulation was repeated ten times, in order to obtain an average. The ϵ error tolerance for the Tau-Leap algorithms was 0.01. Both the relative and absolute error tolerances for the ODE solver were 10^{-4}.

the Gillespie Direct algorithm, and the significant speed improvement of Tau-Leap algorithms over the Gibson-Bruck and Gillespie algorithms. It should be emphasized that no modifications[a] of the model definition file were necessary in order to switch between the various simulation algorithms shown above This is made possible because our model definition language is simulation algorithm-agnostic. Furthermore, the Tau-Leap method does not require an *ad hoc* partitioning of the model into stochastic and deterministic reaction channels. This is a potential advantage in analyzing a complex model for which the "fast" and "slow" degrees of freedom are not known *a priori*.

4. Templates: Reusable and Hierarchical Model Elements

The iterative and integrative process of model building in systems biology entails frequent re-use and adaptation of existing models of biochemical networks and systems. The increasing complexity of models leads to repetition of structurally similar network elements within a model. We employ a template semantic to enable the definition of reusable and hierarchical model elements. In this section we describe this template feature.

[a]Changing the simulation algorithm necessitates changing certain parameters related to simulation control and accuracy. These changes occur outside the scope of the model definition file.

A template is a parameterized macro that has a *type name*. The template may be referenced multiple times within a biochemical model. The type name gives some indication of the function and content of the template. For example, one might define a template with type name `yeastGene`. Each time a template is used within a model, it is given an *instance name*, such as `GAL7`. For a given template, all instance names must be unique.

A template typically contains one or more chemical species definitions and reactions that are *internal* to the template, which means that these species and reactions inhabit a namespace that is "local" and distinct from the "global" namespace of the model. This means that if template `yeastGene` contains a definition for species `A`, then the template instance `GAL7` will contain a species `GAL7::A`. This ensures that chemical species and reactions that are physically or conceptually localized to a particular template instance, do not interfere with their counterparts in another instance of the same template.

A template is a *parameterized* macro, which enables the template instance to interact with the rest of the model. A template parameter may simply be a numeric value such as the rate of a biochemical process or the coding region length of a gene. Each template instance may have a different value, that is passed to the template instance through the template reference semantic. Alternatively, a template parameter may represent a chemical species that is not localized to the template instance, for example, a transcription factor in the nucleus. Finally, a template parameter may represent a numeric quantity that is the "output" of the template. For instance, the template feature is used to define fractional saturation functions for transcriptional activation, and the value of the fractional saturation function is the "output" parameter of the template (see Section 5 for more information).

Templates are hierarchical in the sense that a template definition may contain a template instance. In such a case, the inner template namespace is nested within the containing template's namespace. A file inclusion mechanism allows the separation of template definitions from the model definition. This facilitates the development and re-use of a library of previously defined and curated model elements, such as genes, fractional saturation functions, and so forth.

A sample model definition file employing the template feature is shown in Section 5 of the Supplementary Material. The model describes the galactose utilization pathway in yeast.

5. Stochastic reactions with complex kinetic rate laws

Reactions described by complex kinetic rate laws present a particular challenge for stochastic modeling. Most formulations of stochastic chemical kinetics are based upon the assumption of elementary mass-action kinetics for all reaction channels[55]. This assumption often does not apply to complex rate laws, which may consist of many independent elementary interaction steps. These steps may be unknown, or their kinetic data may be unknown. This lack of sufficient information to formulate

the kinetics of the system completely in terms of elementary (mass-action) reaction channels is a stumbling block to performing stochastic simulations of complex biochemical networks. In a related problem, very large collections of fast-occurring reaction channels within a model may be prohibitively expensive to stochastically simulate. In both cases, it may be desirable to simulate a reduced master equation that approximates the time-averaged effect of the fast reaction channels. In this section we describe how our simulation environment enables the stochastic simulation of such a reduced master equation.

Rao and Arkin propose an approximate method[16] for stochastically simulating an enzyme-catalyzed processes governed by a complex rate law. In this method, the rate law is evaluated in terms of the number of substrate molecules, which will be a random (fluctuating) variable. The rate is then converted to the units of the reaction parameter, molecules per second. The Rao-Arkin method is applicable under the "quasi-steady-state" (QSS) condition, in which there exists a large separation between the (shorter) time scale of the reversible enzyme-substrate complex reaction events, and the (longer) time scales of interest in the system. Rao and Arkin suggest that under the QSS conditions, this approximation represents a systematic reduction of the chemical master equation based on time scales.

With the goal of enabling the stochastic simulation of models using complex rate expressions beyond mass-action kinetics, our model definition environment allows defining the rate of a chemical reaction as an arbitrary mathematical expression. It is understood that one is solving the stochastic dynamics of an approximate *reduced master equation,* and that the validity of the solution depends on the validity of the reduced master equation. This feature enables the solution of the dynamics of systems involving the coupled dynamics of a regulatory network and a metabolic pathway. In the regulatory network, intrinsic stochasticity is important. In the metabolic pathway, metabolite conversions are taking place at such a high rate that one may apply the QSS assumption for the detailed interactions underlying the pathway steps.

The initiation of transcription for genes in the *GAL* regulon in yeast illustrates the efficiency of using complex kinetic rate expressions in a stochastic simulation. In the *GAL* regulon, the Gal4p homodimer is a transcription factor, and the Gal80p homodimer can bind on top of Gal4p and repress transcription. For genes with multiple binding sites, Gal80p dimer binding is thought to be cooperative. The number of Gal4p transcription factor binding sites for *GAL* genes varies from one to five[70]. The number of distinct states (and possible state transitions) increases as three to the power of the number of binding sites. The *GAL2* gene has five binding sites; the number of states of the *cis*-regulatory region is 243. The number of unidirectional elementary reactions implementing transitions between these states is 1580. In a complex multi-gene model, exact modeling of the dynamics of transcription factor binding for all of the *cis*-regulatory regions is computationally prohibitive.

In order to approximately model the effect of transcription factor concentrations

on the stochastic probability density for initiation of transcription, one can define a time-averaged fractional saturation function for transcription initiation at steady-state. This function can be analytically calculated in terms of a sum of probabilities for configurations that lead to transcription[71]. We have derived a general formula for the fractional saturation for N binding sites and three states (empty, factor bound, and factor plus repressor bound) at each binding site, including the effect of cooperativity for transcription factor binding[4]. This formula permits efficient stochastic simulation of transcription initiation as a function of repressor and transcription factor concentrations, without the burden of simulating a multiplicity of elementary reactions for all possible state transitions. This constitutes an approximation to the full underlying dynamics of transcription factor binding. The accuracy of the approximation depends on the relative sizes of two time scales: the time scale over which the transcription factor concentration is varying, and the time scale on which transcription factor binding reactions are occurring. Our simulations indicate that if the former time scale is at least two times the latter, the fractional occupations of the micro-states of the *cis*-regulatory region should remain in quasi-steady-state with respect to the slowly-varying transcription factor concentrations, and the approximate method should have no more than 10% deviation from the transcription initiation rate given by the full stochastic dynamics.

Figure 4 shows the ensemble-averaged fractional saturation calculated using the exact stochastic method, as compared with our time-averaged steady-state fractional saturation function. The excellent agreement between the approximate and exact methods illustrates that in some cases, a large collection of fast-occurring elementary reactions may be replaced with an algebraic expression summarizing the steady-state, time-averaged behavior. In addition, the dependence of the fractional saturation level on the number of binding sites is consistent with expectations based on thermodynamics[4]. Using the template feature of the Dizzy modeling language described in Section 4, the fractional saturation function is made available as a reusable modeling element.

6. Multi-step and Delayed Reaction Processes

The dynamics of genetic regulatory networks involves delays due to translocation/elongation times for transcription and translation[6, 18, 72]. In this section, we describe our two methods for modeling such processes.

Following Gibson and Bruck's model of transcription as a cascade of structurally identical chemical reaction steps[56, 73], we have implemented a "multi-step" reaction channel as a modeling element in our system. Conceptually, a multi-step reaction is a chain of elementary, irreversible reaction steps transforming species A into species B. The reaction parameter for each elementary reaction step is k. We have implemented methods for modeling the kinetics of such multi-step processes in both the deterministic and stochastic simulation cases.

In the case of stochastic simulations, the method of Gibson and Bruck[56] is used,

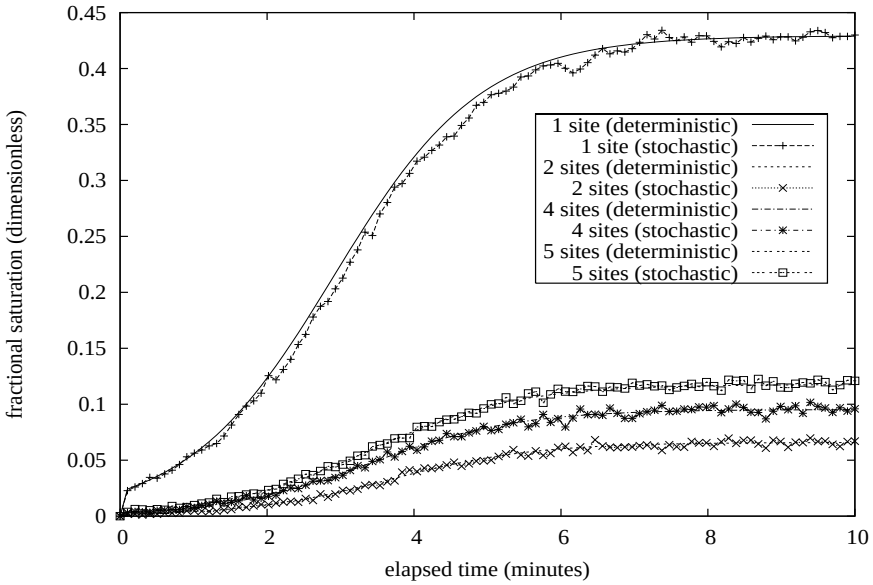

Fig. 4. Comparison between the exact stochastic model and the approximate ODE model of transcriptional activation, for genes with different numbers of binding sites in the *GAL* regulon in yeast. The stochastic simulation includes all possible state transitions between different binding site states for the *cis*-regulatory region; the Gibson-Bruck algorithm was used, with an ensemble size of 1000. The ODE solution shows the transcriptional activation in the case where a fractional saturation function is used to approximately model the average effect of the possible *cis*-regulatory interactions; the Dormand-Prince 5/4 algorithm with a relative tolerance of 10^{-4} was used.

in which the probability density distribution $P(\tau)$ for the multi-step reaction to complete after it has started is given by the gamma distribution. For the case of deterministic (ODE) kinetics, we implement a numerical method for solving the multi-step kinetics as an integro-differential equation. We compute the instantaneous rate of production of B as k multiplied by a convolution of the concentration A with $P(\tau)$. For details, please refer to Section 3.4 of the Supplementary Material.

As an approximation to a multi-step reaction with a large number of steps, Dizzy allows the definition of an irreversible reaction $A \longrightarrow B$ with an intrinsic time delay. For the case of a stochastic simulation of a reaction channel with a time delay, we employ a non-Markovian stochastic process in which a molecule of species A is consumed, and a molecule of species B is produced, with a fixed delay between the two events. For the case of a deterministic simulation of a reaction channel with a time delay, we employ a delay differential equation (DDE)[74]. For a detailed description of the implementation, please refer to Section 3.3 in the Supplementary Material.

The yeast galactose utilization model has been analyzed with the inclusion of translocation delay times for transcription and translation. Figure 5 shows the time-series data for mRNA levels for the model. The effect of the elongation time is clearly

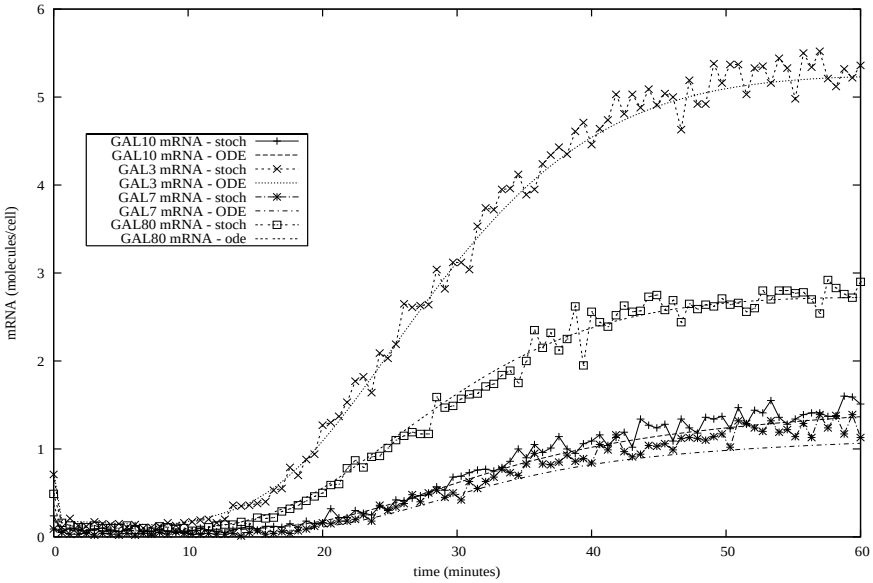

Fig. 5. Time-series plot of mRNA levels in the galactose utilization model, including elongation delays for transcription and translation, comparing the stochastic and ODE simulation methods. The ODE dynamics was solved using the Runge-Kutta 5/4 algorithm, with a relative tolerance of 10^{-4}. The stochastic dynamics was solved using the Gibson-Bruck algorithm, with an ensemble size of 100. The external galactose level is 0.5 mM, which corresponds to 10% induction.

visible, as compared to the same model with no translocation/elongation delays[4].

7. Estimation of steady-state stochastic noise

In model development, it is frequently necessary to conduct many simulations of a model, in order to optimize model parameters for best agreement with experimental data. The high computational cost of conducting stochastic simulations of complex models motivates the need for a fast technique to estimate the steady-state noise level in a biochemical network. Orrell *et al.*[75] have developed a fast method for estimating the steady-state stochastic noise in a kinetic model.

The system, consisting of M reaction channels and N species, is evolved deterministically using ODEs to steady-state, and the $N \times N$ Jacobian matrix $\mathbf{J}$ is computed and diagonalized. The eigenvalues of $\mathbf{J}$ are represented as a vector $\vec{E}$. The real parts of $\vec{E}$ are checked to ensure negativity at the steady-state. We construct a diagonal matrix $\mathbf{T}$ whose elements $T_{ii'}$ are given by the formula,

$$T_{ii'} = I_{ii'} \frac{1}{\sqrt{-\mathrm{Re}(E_i)}} \tag{1}$$

where $\mathbf{I}$ is the identity matrix, and $i, i' \in \{1, \ldots, N\}$. We then compute the matrix $\mathbf{P}$ of eigenvectors of $\mathbf{J}$. The stochastic variance σ_i^2 of the ith species at steady state

can then be computed using the formula[75]

$$\sigma_i^2 = \frac{1}{2}\sum_{j=1}^{M} a_j \left[(\mathbf{PTP}^{-1}\mathbf{v})_{ij}\right]^2 . \tag{2}$$

where the ij subscript denotes the ij element of the matrix $\mathbf{PTP^{-1}}$, a_j is the instantaneous rate of the jth reaction in events per unit time, and $\mathbf{v}$ is the $N \times M$ stoichiometric matrix for the system. Because this technique is based on a linearization of the model around a steady state, it is only applicable for estimating the fluctuations of the system when it is near steady state.

The aforementioned steady-state fluctuation estimation technique has been applied to a reduced version of the yeast galactose utilization model (see Section 5 of the Supplementary Material) governing the production of the protein Gal7p, when the external galactose level is 0.5 mM. Simulations were performed using both an ODE solver and the Gibson-Bruck simulator. The results are summarized in Figure 6. The results show that the technique gives a reasonable estimate for the

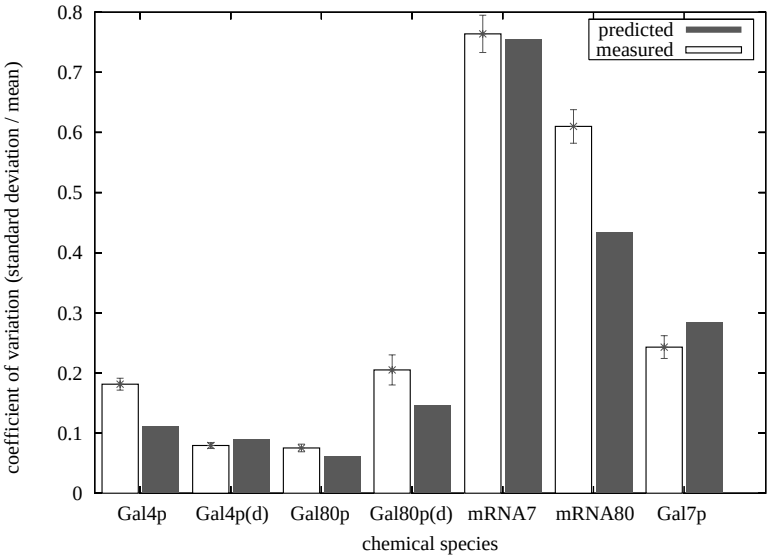

Fig. 6. Coefficient of variation of the steady-state fluctuations of the species involved in the production of Gal7p in the yeast galactose utilization pathway. The stochastic method of calculating the fluctuations ("measured"), and the ODE estimation technique ("predicted"), are compared. The error bars on the stochastic results are one-sigma uncertainties, due to the finite ensemble size of 100. The `(d)` symbol indicates a homodimer.

steady-state noise level, even for the case of a complex model. The degree of accuracy in the estimate will depend on how close the system is to steady-state, and on the extent of non-linearities in the model that may render the linearized approximation invalid. The technique quite accurately quantifies the change expected in the

noise level due to a change in the model[75]. The ODE-based estimation technique is extremely fast, in this case 10^5 times faster than the Monte Carlo (stochastic) method.

Because of the speed of the estimation technique and its accuracy for quantifying changes to noise based on changes to the model, the technique is well suited for use in model parameter optimization, as well as estimating whether a full stochastic simulation is needed to capture the steady-state behavior of chemical species of interest. In addition, the estimation technique can be used to quantify the various contributions to the noise for a given chemical species, from the different reaction channels in the model[76].

8. Integrated, graphical, and portable software framework

The notable software features of Dizzy are: portability across many computer architectures, integration with external software programs, and a graphical user interface (GUI). In this section, these software features are described.

Dizzy is implemented in the Java programming language, which enables Dizzy to execute on any computer platform for which a Java 2 Runtime Environment of version 1.4.1 or newer, is available. Dizzy has been optimized for efficient numerical computation in the Java Runtime Environment.

Dizzy is capable of simulating models expressed in the Systems Biology Markup Language (SBML)[77, 78] Level 1. Dizzy can also export a model into SBML. For optimal performance, a model should be written in the Dizzy model definition language, rather than imported from SBML. The SBML import feature enables Dizzy to simulate a model constructed with the BioTapestry software program[79].

Three of the simulation algorithms within Dizzy may be invoked through the Systems Biology Workbench (SBW)[42]; the Gibson-Bruck, Gillespie Direct, and Runge-Kutta ODE simulators. These simulators may be easily invoked from any SBW-enabled model development platform, such as CellDesigner[80] or JDesigner[81].

Dizzy provides a menu-driven graphical user interface. This user interface includes screens for simulation control, model editing, plotting simulation results, and browsing/searching the hypertext user manual. A screen capture of the Dizzy graphical user interface is shown in Figure 7. Visualization of a biochemical model in a graphical representation is enabled through a software bridge to the Cytoscape[84] software system. More information about Dizzy's software implementation, including a listing of library dependencies, can be found in Section 3.1 of the Supplementary Material.

Dizzy is free and open-source software, distributed under the GNU Lesser General Public License (LGPL)[85]. The Dizzy software is available for download at the web page `http://magnet.systemsbiology.net/dizzy`.

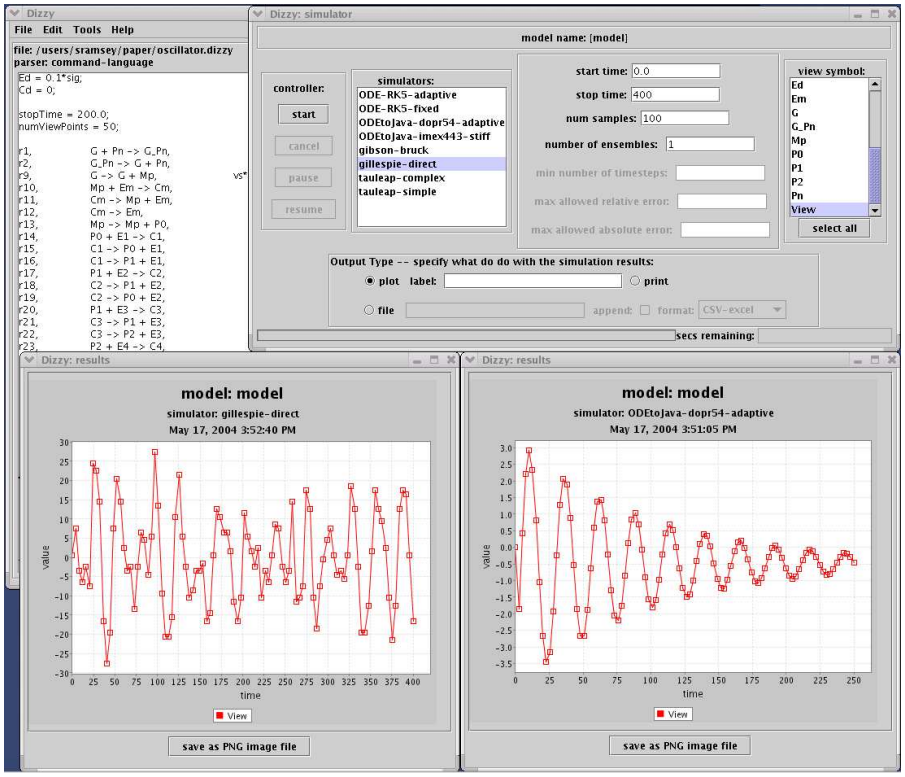

Fig. 7. A screen capture of the Dizzy program showing a simulation of a model of circadian rhythms in *Drosophila melanogaster*[82]. The results indicate that the model has a stable fixed point when simulated deterministically, but has unstable perpetual oscillations when simulated stochastically[83].

9. Conclusions

In this paper we have presented a comprehensive software tool for conducting stochastic and deterministic simulations of the dynamics of complex networks of biochemical reactions. The tool is particularly well suited for modeling the dynamics of integrated large-scale genetic, metabolic, and signaling networks.

Acknowledgments

This research was supported in part by grant #10830302 from the National Institute of Allergy and Infectious Diseases. Pedro de Atauri, Daehee Hwang, and William Longabaugh contributed helpful discussions. Raymond Spiteri kindly provided the ODEToJava numerical ODE solver library[65].

References

1. E. H. Davidson *et al.*, "A genomic regulatory network for development," *Science*, vol. 295, pp. 1669–1678, 2002.
2. J. Li *et al.*, "Green fluorescent protein in *Saccharomyces cerevisiae:* real-time studies of the *GAL1* promoter," *Biotech. Bioeng.*, vol. 70, no. 2, pp. 187–196, 2000.
3. S. R. Biggar and G. R. Crabtree, "Cell signaling can direct either binary or graded transcriptional responses," *EMBO J.*, vol. 20, no. 12, pp. 3167–3176, 2001.
4. P. de Atauri, D. Orrell, S. Ramsey, and H. Bolouri, "Evolution of 'design' principles in biochemical networks," *IEE Sys. Bio.*, vol. 1, pp. 28–40, 2004.
5. D. M. Underhill, A. Ozinsky, K. D. Smith, and A. Aderem, "Toll-like receptor-2 mediates mycobacteria-induced proinflammatory signaling in macrophages," *Proc. Nat. Acad. Sci. USA*, vol. 96, no. 25, pp. 14459–14463, 1999.
6. H. Bolouri and E. H. Davidson, "Transcriptional regulatory cascades in development: Initial rates, not steady state, determine network kinetics," *Proc. Nat. Acad. Sci. USA*, vol. 100, no. 16, pp. 9371–9376, 2003.
7. P. Guptasarma, "Does replication-induced transcription regulate synthesis of the myriad low copy number proteins of *E. coli?*," *BioEssays*, vol. 17, pp. 987–997, 1995.
8. M. B. Elowitz, A. J. Levine, E. D. Siggia, and P. S. Swain, "Stochastic gene expression in a single cell," *Science*, vol. 297, pp. 1183–1186, 2002.
9. E. M. Ozbudak *et al.*, "Regulation of noise in the expression of a single gene," *Nature Gen.*, vol. 31, pp. 69–73, 2002.
10. W. J. Blake, M. Kaern, C. R. Cantor, and J. J. Collins, "Noise in eukaryotic gene expression," *Nature*, vol. 422, pp. 633–637, 2003.
11. D. T. Gillespie, "Approximate accelerated stochastic simulation of chemically reacting systems," *J. Chem. Phys.*, vol. 115, no. 4, pp. 1716–1733, 2001.
12. E. L. Haseltine and J. B. Rawlings, "Approximate simulation of coupled fast and slow reactions for stochastic chemical kinetics," *J. Chem. Phys.*, vol. 117, pp. 6959–6969, 2002.
13. D. T. Gillespie and L. R. Petzold, "Improved leap-size selection for accelerated stochastic simulation," *J. Chem. Phys.*, vol. 119, no. 16, pp. 8229–8234, 2003.
14. J. Puchalka and A. M. Kierzek, "Bridging the gap between stochastic and deterministic regimes in the kinetic simulations of the biochemical reaction networks," *Biophys. J.*, vol. 86, pp. 1357–1372, 2004.
15. T. R. Kiehl, R. M. Mattheyses, and M. K. Simmons, "Hybrid simulation of cellular behavior," *Bioinf.*, vol. 20, no. 3, pp. 316–322, 2004.
16. C. V. Rao and A. P. Arkin, "Stochastic chemical kinetics and the quasi-steady-state assumption: Application to the Gillespie algorithm," *J. Chem. Phys.*, vol. 118, no. 11, pp. 4999–5010, 2003.
17. K. Vasudeva and U. S. Bhalla, "Adaptive stochastic-deterministic chemical kinetic simulations," *Bioinf.*, vol. 20, no. 1, pp. 78–84, 2004.
18. A. Gilman and A. P. Arkin, "Genetic 'Code': Representations and dynamical models of genetic components and networks," *Annu. Rev. Genom. Hum. Genet.*, vol. 3, pp. 341–369, 2002.
19. T. C. Meng, S. Somani, and P. Dhar, "Modelling and simulation of biochemical systems with stochasticity," *In Silico Bio.*, vol. 4, p. 0024, 2003.
20. L. You, "Toward computational systems biology," *Cell Biochem. Biophys.*, vol. 40, pp. 1–18, 2004.
21. D. Adalsteinsson, D. McMillen, and T. C. Elston, "Biochemical network stochastic simulator (BioNetS): software for stochastic modeling of biochemical networks," *BMC Bioinf.*, vol. 5, no. 1, p. 24, 2004.

22. C. Priami *et al.*, "Application of a stochastic name-passing calculus to representation and simulation of molecular processes," *Inf. Proc. Lett.*, vol. 80, pp. 25–31, 2001.
23. E. O. Voit, *Computational analysis of biochemical systems theory*. New York, NY, USA: Cambridge University Press, 2000.
24. B. E. Shapiro *et al.*, "Cellerator: Extending a computer algebra system to include biochemical arrows for signal transduction modelling," *Bioinf.*, vol. 19, no. 5, pp. 677–678, 2003.
25. H. Rubin, V. Kumar, and A. Owen, "Modeling and simulation of bioregulatory networks in BioCharon," tech. rep., DARPA BioSPICE Consortium, June 2002.
26. P. Dhar *et al.*, "Cellware–a multi-algorithmic software for computational systems biology," *Bioinf.*, vol. 20, no. 8, pp. 1319–1321, 2004.
27. I. Goryanin, T. C. Hodgman, and E. Selkov, "Mathematical simulation and analysis of cellular metabolism and regulation," *Bioinf.*, vol. 15, no. 9, pp. 749–758, 1999.
28. L. You, A. Hoonlor, and J. Yin, "Modeling biological systems using Dynetica – a simulator of dynamic networks," *Bioinf.*, vol. 19, pp. 435–436, 2003.
29. M. Tomita *et al.*, "E-Cell: software environment for whole-cell simulation," *Bioinf.*, vol. 15, pp. 72–84, 1999.
30. C. D. Cox *et al.*, "Analysis of noise in quorum sensing," *OMICS J. Integ. Bio.*, vol. 7, no. 3, pp. 317–334, 2003.
31. M. Nagasaki *et al.*, "Genomic Object Net: I. A platform for modelling and simulating biopathways," *Appl. Bioinf.*, vol. 2, no. 3, pp. 181–184, 2003.
32. P. Mendes, "GEPASI: a software package for modelling the dynamics, steady states, and control of biochemical and other systems," *Comp. Appl. Biosci.*, vol. 9, pp. 563–571, 1993.
33. P. Mendes and D. B. Kell, "MEG: a program for the modelling of complex, heterogeneous, cellular systems," *Bioinf.*, vol. 17, pp. 288–289, 2001.
34. H. Sauro, *Jarnac (Scamp II) — Reference Guide*. Control and Dynamical Systems, California Institute of Technology, Nov 2001.
35. E. Butterworth, "JSim." Computer software, National Simulation Resource, University of Washington, USA, `nsr.bioeng.washington.edu`, 2003.
36. B. G. Olivier and J. L. Snoep, "Web-based kinetic modelling using JWS online," *Bioinf.*, vol. 20, no. 13, pp. 2143–2144, 2004.
37. B. Aleman-Meza *et al.*, "KINSOLVER: A simulator for computing large ensembles of biochemical and gene regulatory networks." preprint, `gene.genetics.uga.edu/stc`, 2004.
38. J. R. Stiles, "Monte Carlo simulation of neurotransmitter release using MCell, a general simulator of cellular physiological processes," in *Computational Neuroscience* (J. M. Bower, ed.), pp. 279–284, New York: Plenum, 1998.
39. M. Schwehm, "Parallel stochastic simulation of whole-cell models," in *Proc. Int. Conf. on Sys. Bio.*, pp. 333–341, 2001.
40. J. Hurlebaus, W. Wiechert, and R. Takors, "MMT - a metabolic modeling tool for metabolic engineering," in *GCB 2001 German Conf. on Bioinf.*, 2002.
41. M. Schilstra, "NetBuilder." Computer software, University of Hertfordshire, UK, `strc.herts.ac.uk/bio/maria/NetBuilder`, 2002.
42. M. Hucka, A. Finney, H. Sauro, and H. Bolouri, "Introduction to the Systems Biology Workbench," tech. rep., JST ERATO Kitano Symbiotic Systems Project, May 2001.
43. H. Sauro, "Pasadena Twain." Computer software, Keck Graduate Institute, `www.sys-bio.org`, 2002.
44. H. Sauro, "Gillespie Service." Computer software, JST ERATO Kitano Symbiotic Systems Project, California Institute of Technology, USA,

`sbw.sourceforge.net/sbw/software`, 2002.
45. A. Finney, M. Hucka, H. Sauro, and H. Bolouri, "Gibson Service." Computer software, JST ERATO Kitano Symbiotic Systems Project, California Institute of Technology, USA, `sbw.sourceforge.net/sbw/software`, 2002. Included in the Systems Biology Workbench (M. Hucka *op. cit.*), based on source code from E. Lyons *op. cit.*
46. P. van der Zee, "Tau Leap." Computer software, University of Hertfordshire, UK, `strc.herts.ac.uk/bio/tauleap`, 2002.
47. P. Divalentin and M. F. Pettigrew, "SigTran." Computer software, Cell Systems Initiative, University of Washington, USA, `csi.washington.edu`, July 2002.
48. M. Antoniotti *et al.*, "Model building and model checking for biochemical processes," *Cell Biochem. Biophys.*, vol. 38, pp. 271–286, 2003.
49. E. Lyons, "Stochastirator." Computer software, Molecular Sciences Institute, Berkeley, CA, USA, `opnsrcbio.molsci.org/stochastirator/stoch-main.html`, 2000.
50. N. le Novère and T. S. Shimizu, "StochSim: modelling of stochastic biomolecular processes," *Bioinf.*, vol. 17, no. 6, pp. 575–576, 2001.
51. A. M. Kierzek, J. Zaim, and P. Zielenkiewicz, "The effect of transcription and translation initiation frequencies on the stochastic fluctuations in prokaryotic gene expression," *J. Bio. Chem.*, vol. 276, pp. 8165–8172, 2001.
52. C. van Gend and U. Kummer, "Automatic stochastic simulation of systems described by differential equations," in *Proc. Int. Conf. on Sys. Bio.*, pp. 326–332, 2001.
53. P. J. Goss and J. Peccoud, "Quantitative modeling of stochastic systems in molecular biology by using stochastic Petri nets," *Proc. Nat. Acad. Sci. USA*, vol. 95, pp. 6750–6755, 1998.
54. L. M. Loew and J. C. Schaff, "The Virtual Cell: a software environment for computational cell biology," *Trends Biotech.*, vol. 19, no. 10, pp. 401–406, 2001.
55. D. T. Gillespie, "A general method for numerically simulating the stochastic time evolution of coupled chemical reactions," *J. Comp. Phys.*, vol. 22, pp. 403–434, 1976.
56. M. A. Gibson and J. Bruck, "Efficient exact stochastic simulation of chemical systems with many species and many channels," *J. Phys. Chem. A*, vol. 104, pp. 1876–1889, 2000.
57. C. J. Morton-Firth and D. Bray, "Predicting temporal fluctuations in an intracellular signaling pathway," *J. Theo. Bio.*, vol. 192, pp. 117–128, 1998.
58. J. L. Bail, H. Alla, and R. David, "Hybrid Petri nets," in *Proc. 1st Int. European Control Conf.* (I. D. Landau, ed.), (Paris), pp. 1472–1477, Edition Hermes, 1991.
59. A. Regev, W. Silverman, and E. Shapiro, "Representation and simulation of biochemical processes using π-calculus process algebra," in *Proc. 6th Pacific Symposium on Biocomputing*, pp. 459–470, 2001.
60. T. D. Garvey *et al.*, "BioSPICE: access to the most current computational tools for biologists," *OMICS J. Integ. Bio.*, vol. 7, no. 4, pp. 441–420, 2003.
61. D. A. McQuarrie, "Stochastic approach to chemical kinetics," *J. Appl. Prob.*, vol. 4, p. 413, 1967.
62. W. H. Press *et al.*, *Numerical Recipes in C*. New Rochelle, NY, USA: Cambridge University Press, 1988.
63. J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*. Berlin, FRG: Springer-Verlag, second ed., 1993.
64. M. D. Patterson, "Implementing Runge-Kutta solvers in Java," tech. rep., Acadia University, 2002. Honors Undergraduate Thesis, Jodrey School of Computer Science.
65. M. D. Patterson and R. J. Spiteri, "OdeToJava." Computer software, Dalhousie University, Halifax, NS, Canada, `www.netlib.org/ode`, 2002.
66. J. R. Dormand and P. J. Prince, "A family of embedded Runge-Kutta formulae," *J.*

Comp. Appl. Math., vol. 6, pp. 19–26, 1980.
67. U. M. Ascher, S. J. Ruuth, and R. J. Spiteri, "Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations," *Appl. Numer. Math.*, vol. 25, pp. 151–167, 1997.
68. R. Srivastava, M. S. Peterson, and W. E. Bentley, "Stochastic kinetic analysis of the *Escherichia coli* stress circuit using $\sigma(32)$-targeted antisense," *Biotech. Bioeng.*, vol. 75, pp. 120–129, 2001.
69. K. Takahashi *et al.*, "A multi-algorithm, multi-timescale method for cell simulation," *Bioinf.*, vol. 20, no. 4, pp. 538–546, 2004.
70. F. K. Zimmermann and K.-D. Entian, *Yeast Sugar Metabolism*. Lancaster, PA, USA: North Holland, 1997.
71. G. K. Ackers, A. D. Johnson, and M. A. Shea, "Quantitative model for gene regulation by λ phage repressor," *Proc. Nat. Acad. Sci. USA*, vol. 79, no. 4, pp. 1129–1133, 1982.
72. H. H. McAdams and A. P. Arkin, "Stochastic mechanisms in gene expression," *Proc. Nat. Acad. Sci. USA*, vol. 94, pp. 814–819, 1997.
73. M. A. Gibson and J. Bruck, "A probabilistic model of a prokaryotic gene and its regulation," in *Computational Modeling of Genetic and Biochemical Networks* (J. M. Bower and H. Bolouri, eds.), ch. 2, MIT Press, 2001.
74. C. T. H. Baker, C. A. H. Paul, and D. R. Willé, "Issues in the numerical solution of evolutionary delay differential equations," *Adv. Comp. Math.*, vol. 3, pp. 171–196, 1995.
75. D. Orrell, S. Ramsey, P. de Atauri, and H. Bolouri, "A method for estimating stochastic noise in large genetic regulatory networks," *Bioinf.*, vol. 21, no. 2, pp. 208–217, 2005.
76. D. Orrell, P. de Atauri, S. Ramsey, and H. Bolouri, "Sources of stochastic noise in the galactose pathway." Unpublished, 2004.
77. M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, *et al.*, "The Systems Biology Markup Language (SBML): a medium for representation and exchange of biochemical network models," *Bioinf.*, vol. 19, no. 4, pp. 524–531, 2003.
78. M. Hucka, A. Finney, H. Sauro, and H. Bolouri, "Systems Biology Markup Language (SBML) Level 1: Structures and facilities for basic model definitions," tech. rep., JST ERATO Kitano Symbiotic Systems Project, May 2003.
79. W. J. R. Longabaugh, E. H. Davidson, and H. Bolouri, "Computational representation of developmental genetic regulatory networks," *Developmental Bio.*, 2005. in press.
80. A. Funahashi *et al.*, "CellDesigner: a process diagram editor for gene-regulatory and biochemical networks," *Biosilico*, vol. 1, no. 5, pp. 159–162, 2003.
81. H. Sauro, *A Introduction to JDesigner version 1.8*. Control and Dynamical Systems, California Institute of Technology, USA, 2003.
82. D. Gonze, J. Halloy, and A. Goldbeter, "Robustness of circadian rhythms with respect to molecular noise," *Proc. Nat. Acad. Sci. USA*, vol. 99, no. 2, pp. 673–768, 2002.
83. D. Orrell and H. Bolouri, "Control of internal and external noise in genetic regulatory networks," *J. Theo. Bio.*, vol. 230, no. 3, pp. 301–312, 2004.
84. P. Shannon *et al.*, "Cytoscape: A software environment for integrated models of biomolecular interaction networks," *Genome Res.*, vol. 13, pp. 2498–2504, 2003.
85. R. Stallman, "GNU lesser general public license." Computer software license agreement, Free Software Foundation, `www.gnu.org/copyleft`, 1999.