CONVERTING MACROMOLECULAR REGULATORY MODELS FROM DETERMINISTIC TO STOCHASTIC FORMULATION

Pengyuan Wang, Ranjit Randhawa, Clifford A. Shaffer, Yang Cao, and William T. Baumann
Virginia Tech, Blacksburg VA, 24061
{`wpy|rrandhawa|shaffer|ycao|baumann`}`@vt.edu`

Keywords: Computational biology, cell cycle, pathway models, Systems Biology Markup Language (SBML)

Abstract

We describe procedures for converting a macromolecular regulatory model from the most common deterministic formulation to one suitable for stochastic simulation. To avoid error, we seek to automate as much of the process as possible. However, deterministic models often omit key information necessary to a stochastic formulation. In this paper we introduce how we implement conversion in the JigCell modeling environment. Our tool makes it easier for the modeler to include complete details. Stochastic simulations are known for being computationally intensive, and thus require high performance computing facilities to be practical. We provide the first stochastic simulation results for realistic cell cycle models, using Virginia Tech's System X supercomputer.

1. REGULATORY NETWORK MODELING

Mathematical modeling of macromolecular regulatory networks in terms of ordinary differential equations has helped to shed light on the detailed workings of the cell cycle and other intracellular processes [3, 11]. However, deterministic continuous models cannot explain some behavior of these systems. Thus, modelers are beginning to employ stochastic methods to improve the accuracy of the simulations, in particular to account for the differences in outcomes that appear in individual cells. The heart of such models are a set of chemical reactions, and there are well-known stochastic simulation techniques for sets of chemical reactions such as Gillespie's stochastic simulation algorithm (SSA) and its variants [2, 6, 7]. Since there currently exist models of value, it is natural to wish to run the existing models using stochastic simulation algorithms, even if those models were created with continuous, deterministic simulators in mind. Even in the future, it might be that modelers will develop initial models using deterministic ODEs for their relative speed and simplicity, before running a full stochastic simulation.

Unfortunately, ODE-based models are not cast in the right form for direct simulation using stochastic methods. The values of species in ODE models are usually written in concentration form, because concentrations are what experimenters measure in the lab. However, stochastic simulations require the model to be in terms of population because they account for individual molecules. Thus, a translation process must take place. For large complex models that could have over a hundred parameters, this process would be tedious and error prone to perform by hand. This might appear to be a trivial issue, but there are two reasons why it is not. First, there is non-trivial reasoning that must be applied to generate the correct scaling functions. Second, this scaling requires careful definition of units for all species, which is typically neglected by deterministic modelers. In this paper we describe our progress toward automatic conversion of models within the JigCell system [9].

Gillespie's stochastic simulation algorithm defines a propensity function that accounts for the probabilities that each reaction will fire. This takes the place of rate laws (velocities) for the reactions in a deterministic formulation of the model. Though they have different physical meanings, these two formulations are closely connected and can be converted one to the other. Our tools are also constrained by the fact that they represent both the deterministic and stochastic forms in the Systems Biology Markup Language (SBML) [8], which is the defacto standard within the systems biology modeling community today.

When building deterministic models, modelers often prefer to use normalized or scaled values for species concentrations, because often the modelers do not know the real value of a species' typical (characteristic) concentration or number of molecules. Thus, before being converted to a stochastic model, the model should be converted to real concentrations first. These scaling factors are typically lost in a deterministic model. Part of a conversion tool's job is to help the modeler correctly input these scaling factors.

Whether a model is written in terms of normalized concentration, real concentration, or population, it describes the same reaction network. Thus, essentially we are identifying the lost information that was not required for the ODE-based model, and changing the form of information describing the model. The key missing information typically is the definition of the units for various constants and state variables. Modelers make implicit assumptions for unit definitions, and many tools today do not support defining this information explicitly. During the conversion process, it is important to enforce units on every species and parameter, and always check unit consistency inside the model. Unit definitions include any scaling factors involved in the corresponding variable. The explicit definition of units is a necessary step in building large and complex models.

When implementing the automatic conversion function, we keep the following goals in mind. The toolkit should be smart enough to automatically infer the correct units for parameters and check unit consistency within the model. It should be convenient for the modeler to input whatever information cannot be inferred. Conversion between concentrations and populations should be automatic once all necessary information is available. It should be flexible so that once converted, the model can be modified at will without any restriction that might be implied from the original deterministic model. JigCell now achieves all these goals, and thereby greatly reduces the work load when modelers move between deterministic and stochastic models. Our technique allows us to do stochastic simulations on significant models.

A stochastic simulation usually requires significantly more computing resources than a deterministic simulation, and that simulation must be run thousands of times to produce ensemble averages. But the ensemble result is inherently easy to compute in parallel. We report here our initial results from using a stochastic simulation package to do simulation of both a small and a more detailed (and therefore larger) budding yeast cell cycle model on a parallel supercomputer. We also compare the time required by a deterministic simulation on the same reaction networks to contrast the difference in the computation time. To our knowledge, this is the first publication of stochastic simulation results on a non-trivial model of the cell cycle.

2. A SIMPLE CELL CYCLE MODEL

To illustrate the process of converting from a deterministic model to a form suitable for stochastic simulation, in this section we introduce a simplified version of a published cell cycle model [3].

The deterministic form of the model consists of six reactions, all of which use complex rate laws that systems biologists have hypothesized as reasonable approximations for more complex series of interactions going on within the cell. We refer to models using such complex rate laws as a “packed” formulation for the model. A stochastic model ideally will be defined by a collection of simple mass action (elementary) reactions. In our model, the packed form uses Michaelis-Menten and Hill functions representing the result of unmodeled elementary reactions. Though stochastic models are usually written in terms of elementary reactions, it is also possible to stochastically simulate reactions with more complex rates. While this may somewhat underestimate the molecular noise, it suffices to demonstrate the validity of our approach. The following is the deterministic form of the model.

$$\emptyset \xrightarrow{\kappa_1 m} Cycb \tag{1}$$

$$Cycb \xrightarrow{\kappa_2'[Cycb]_n} \emptyset \tag{2}$$

$$Cycb + Cdh1 \xrightarrow{\kappa_2''[Cycb]_n[Cdh1]_n} Cdh1 \tag{3}$$

$$Cdh1 \xrightarrow{\frac{\kappa_4[Cdh1]_n[Cycb]_n}{\Gamma_4+[Cdh1]_n}} Cdh1_P \tag{4}$$

$$Cdh1_P \xrightarrow{\frac{\kappa_3''[Cdh1_P]_n[Cdc20]_n}{\Gamma_3+[Cdh1_P]_n}} Cdh1 \tag{5}$$

$$\emptyset \xrightarrow{\kappa_5'+\frac{\kappa_5''[Cycb]_n^2}{\Gamma_5{}^2+[Cycb]_n^2}} Cdc20 \tag{6}$$

$$Cdc20 \xrightarrow{\kappa_6[Cdc20]_n} \emptyset \tag{7}$$

In this model, $Cdh1_P$ is the phosphorylated form for $Cdh1$. Parameter m represents the mass of the cell. The following notation is used through out this paper. For any species S, $[S]_n$ denotes its normalized concentration, and $[S]$ denotes the real concentration. N_S denotes the number of molecules (population) of species S. κ or Γ is a parameter of a normalized model. k and J are scaled versions of κ or Γ, respectively, used in the unnormalized model in terms of real concentrations.

This model includes typical reaction types, namely Mass Action, Michaelis-Menten and Hill functions. However, in practice modelers often use non-standard rate laws. For example, reaction 6 is essentially a combination of Mass Action and a Hill function, but the program cannot determine this. For another example, in SBML, κ_2' could be given an assignment computing any expression, which breaks the assumed meaning of the Mass Action reaction type. This behavior is common in our larger models. Lastly, in SBML, it is possible to define a parameter as a “species,” thus without checking units it is impossible to distinguish between species and parameters. Therefore, the only reliable way to parse arbitrary combinations of different reaction types is to follow the units of species and parameters.

The original deterministic model is written in terms of normalized concentrations, but lacks the following information to be able to be converted: the units, including scaling factors, of species; the units of parameters; and the initial volume of the cell. It is easy to enter the initial volume. However, since all species and parameters are inter-related by the reaction network, it is not easy to make sure all the units, once entered or changed, are consistent with each other. For example, if the modeler wants to change the characteristic concentration of $Cycb$ by changing its scaling factor, he or she must remember to change the units of κ_1, κ_4 and Γ_5 correctly. It takes careful reasoning to recognize that there is no need to change κ_2', κ_2'' or κ_5'', though it seems that they are also connected with $Cycb$. All of this would be a burden to determine by hand, but can be automated.

3. SOFTWARE TOOLS

Our conversion software is an extension of the JigCell Model Builder (JCMB) [13]. JCMB is part of the JigCell environment for developing and analyzing reaction network models [1, 9]. JCMB allows users to enter their models into a simple spreadsheet-like interface that organizes all the necessary information in an easy-to-read form. Its organization leads users to provide the complete information necessary, and its series of internal checks helps to minimize errors in complex models.

JigCell inputs and outputs models in the SBML language. While this is the defacto standard for models in systems biology, and while SBML is rapidly developing to support stochastic models, it still lacks certain key features. For example, SBML cannot describe stochastic events. SBML also has no direct method for allowing users to indicate whether a given model is in a deterministic or stochastic form. Fortunately, once a model is completely specified as described in the next section, it is suitable for simulation by both deterministic and stochastic simulators.

We use StochKit [10] to do stochastic simulation on the converted models. StochKit is an efficient, extensible stochastic simulation framework developed in C++. All simulation methods in StochKit are based on Gillespie's SSA and tau-leaping algorithms. We have integrated StochKit as one of the simulators available within JigCell.

4. APPROACH

In this section we first introduce general rules to convert from deterministic to stochastic form, and then we introduce how unit definitions provide the necessary information to accomplish the converting task.

4.1. Model conversion

In Gillespie's algorithm, the propensity function for each reaction replaces the rate law equation in the ODE model, and is written in terms of numbers of molecules. They share the same expression except when a reaction contains two or more of the same reactants. For example, $A+A \longrightarrow B$'s propensity function would be $kN_A(N_A-1)$. The deterministic reaction rate law would be kN_AN_A [6]. This is a minor issue, so we will focus on how to convert a model in terms of normalized concentration to a model in terms of number of molecules.

For a species S, let c_S be the scaling factor between its normalized concentration and real concentration, thus

$$[S] = c_S[S]_n \tag{8}$$

Multiplying the concentration by volume will produce the population,

$$N_S = V[S] \tag{9}$$

Here we define $V = (cell\ volume)\cdot(Avogadro\ number)$ to get N_S in terms of number of molecules other than $mole$. Depending on the model, V could be a constant or a variable. c_s is a constant. Different species may have different values for c_s. Consider Reaction (3). Its contribution to the differential equation of $Cycb$ is

$$\frac{d[Cycb]_n}{dt} = \ldots - \kappa_2''[Cycb]_n[Cdh1]_n \tag{10}$$

where $\ldots$ represents contribution to the ODE from other reactions. The first step is to convert it to a model in terms of real concentration.

$$\begin{aligned}\frac{d[Cycb]}{dt} &= \frac{dc_{Cycb}[Cycb]_n}{dt} = c_{Cycb}\frac{d[Cycb]_n}{dt}\\ &= \ldots - c_{Cycb}\kappa_2''\frac{[Cycb]}{c_{Cycb}}\frac{[Cdh1]}{c_{Cdh1}}\\ &= \ldots - \frac{\kappa_2''}{c_{Cdh1}}[Cycb][Cdh1]\\ &\triangleq \ldots - k_2''[Cycb][Cdh1] \end{aligned} \tag{11}$$

The conversion involves transformation of species and parameters, but the form of rate law expression is kept, because essentially it is just a scaling of the system. Parameters are used to digest any constants introduced during the conversion of species, and we shall see later that information for how to transform parameters is embedded in their units.

The second step is to convert the model to population. For any species S, the differential equation for population is

$$\frac{dN_S}{dt} = \frac{d(V[S])}{dt} = V\frac{d[S]}{dt} + [S]\frac{dV}{dt} \tag{12}$$

For the fixed volume case, only the first term $V\frac{d[S]}{dt}$ is not zero and the conversion is similar to the first step. But when the volume is changing, additional terms are required to make the ODE for population complete and correct. Here we only consider the fixed volume case. Then the model in terms of population would be

$$\frac{dN_{Cycb}}{dt} = V\frac{d[Cycb]}{dt} = \ldots - \frac{k_2''}{V}N_{Cycb}N_{Cdh1} \tag{13}$$

Therefore, when changing the model from normalized concentration to population, the rate law equation (or propensity function) for reaction 3 would be $\frac{k_2''}{V}N_{Cycb}N_{Cdh1}$.

Generally speaking, let f_N, f_R, f_M be the rate law function for a reaction in normalized concentration, real concentration and number of molecules. The rule for conversion is:

$$\frac{d[\hat{S}]_n}{dt} = \ldots + f_N(\bar{\kappa}, \overline{[S]_n}) \quad (14)$$

$$\frac{d[\hat{S}]}{dt} = \ldots + f_R(\bar{k}, \overline{[S]}) = \ldots + f_N(\bar{\kappa}, \overline{(\frac{[S]}{c_S})})c_{\hat{S}} \quad (15)$$

$$\frac{dN_{\hat{S}}}{dt} = \ldots + f_M(\bar{k}, \overline{N_S}, V) = \ldots + f_R(\bar{k}, \overline{(\frac{N_S}{V})})V \quad (16)$$

where $\bar{x}$ represents an array of all the parameters or species participating in the reaction, $\hat{S}$ is the species whose value will be changed by the reaction, and $c_{\hat{S}}$ is its scaling factor. If more than one species is changed by the same reaction, they must have the same scaling factor. k is defined from κ by digesting all the introduced scaling factors as shown in (11), thus $f_R = f_N$.

4.2. Implementation

We think of the scaling factor as an integral part of a species' unit. In SBML, every species and parameter has a value field and a reference to a "Unit Definition" as its unit. Modelers can define arbitrary "Unit Definitions" from a combination of a list of "Units," where "Units" are transformations of pre-defined base units [8]. For example,

$$\{u\} = (m_1 10^{s_1}\{u_{b_1}\})^{x_1}(m_2 10^{s_2}\{u_{b_2}\})^{x_2}\ldots(m_n 10^{s_n}\{u_{b_n}\})^{x_n} \quad (17)$$

where $\{u\}$ is the "Unit Definition," $\{u_{b_i}\}$ is the base unit, $(m_i \cdot 10^{s_i}\{u_{b_i}\})^{x_i}$ is a transformation. This is a very broad definition of unit, which one can take advantage of to store the scaling factor and do model conversion.

For example, in a deterministic model of normalized concentration, one can assign the unit of $Cycb$ to be $50 \cdot 10^{-9} \cdot Mole \cdot L^{-1}$, in other words, $50nM$. (In SBML, Unit Definition can only be defined from base units, not from other definitions, that is, it is not hierarchical. Thus, the model does not have a direct unit of $Molar$.) The meaning of this unit definition is that one unit of normalized concentration equals 50 times one unit of real concentration (which we assume to be nM). Thus, the scaling factor is 50.

When converting the model, instead of using different notations for different meanings of species and parameter values as shown above, there is always the same set of species and parameters. What will be changed are the units and values of species and parameters. Let the unit of a species or parameter S be u_S and the value be v_S. In the first step of the conversion, we need to change the unit of a species from normalized concentration to real concentration by removing the scaling factor from the Unit Definition, and multiplying it to the value field of the species. If we think of the unit and the value as a whole, we are only changing the form of the model.

To maintain the consistency of the units inside the model, the units of parameters should be changed as well. In most cases, the unit of a parameter is determined by all the other species participating in the reaction. Denoting the unit of quantity q by u_q and the time unit by u_t, we require

$$u_{\frac{dS}{dt}} = u_{f(\bar{A})} \quad (18)$$

$$\Rightarrow \frac{u_S}{u_t} = f(u_{\bar{A}}) \quad (19)$$

Here S could be any form of a species, be it in concentration or population, and f is the corresponding rate law function. $\bar{A}$ denotes all the arguments of f.

Thus, we are able to infer the right unit of a parameter, or how to change the unit of a parameter when changing the units of the corresponding species. For example, from (11) we can conclude:

$$\frac{u_{Cycb}}{u_t} = u_{k_2''} u_{Cycb} u_{Cdh1} \quad (20)$$

$$\Rightarrow u_{k_2''} = u_{Cdh1}{}^{-1} u_t^{-1} \quad (21)$$

Similar to species, any scaling of the unit of a parameter should be accompanied by the corresponding scaling of the parameters's value as well, otherwise the integrity of the model will be broken. Before the conversion, $u_{k_2''} = 50^{-1} nM^{-1} min^{-1}$. Converting to real concentration will change it to $u_{k_2''} = nM^{-1} min^{-1}$, and 50^{-1} will be multiplied to its value field. By examining units we are able to reason the correct scaling of parameters as we did in (11).

If there are multiple parameters appearing in the same reaction, such as Reaction (1), usually only the product result of these units can be automatically inferred, and the modeler must split the units between multiple parameters.

To convert the model from real concentration to population, since $N_S = V[S]$, we have:

$$v_{N_S} \cdot u_{N_S} = (v_V \cdot u_V)(v_{[S]} \cdot u_{[S]}) = (v_V \cdot v_{[S]})(u_V \cdot u_{[S]}) \quad (22)$$

$$\Rightarrow v_{N_S} = v_V v_{[S]},\ u_{N_S} = u_V u_{[S]} \quad (23)$$

So both the value and units of the species should be multiplied accordingly. By taking units into consideration we are making sure that the converted model is consistent with the original model.

Instead of adjusting the parameters' units and values in response to changes made on species, here we choose to compensate for multiplication of species by V by adjusting the rate law expression, as exemplified in (13) and (16), because V is a variable in the model.

To sum up, the converted model in terms of population is:

$$\emptyset \xrightarrow{k_1 mV} Cycb \quad (24)$$

$$Cycb \xrightarrow{k_2' N_{Cycb}} \emptyset \quad (25)$$

$$Cycb + Cdh1 \xrightarrow{\frac{k_2''}{V} N_{Cycb} N_{Cdh1}} Cdh1 \quad (26)$$

	Old		New	
Species	Value	Unit	Value	Unit
$Cycb$	0.0535	$50nM$	48	molecules
$Cdh1$	0.0536	$50nM$	48	molecules
$Cdh1_P$	0.946	$50nM$	852	molecules
$Cdc20$	0.111	$50nM$	100	molecules

Table 1. Initial conditions of species in the model. Population is calculated using $V = 18 molecules/nM$, that is, cell volume $= 30fL$.

	Old		New	
P.	Value	Unit	Value	Unit
k_1	0.01	$50nM/(min \cdot ng)$	5	$nM/(min \cdot ng)$
m	1.5	ng	1.5	ng
k'_2	0.04	$1/min$	0.04	$1/min$
k''_2	1	$1/(50nM \cdot min)$	0.02	$1/(nM \cdot min)$
k''_3	10	$1/min$	10	$1/min$
k_4	35	$1/min$	35	$1/min$
k'_5	0.005	$50nM/min$	0.25	nM/min
k''_5	0.2	$50nM/min$	10	nM/min
k_6	0.1	$1/min$	0.1	$1/min$
J_3	0.04	$50nM$	2	nM
J_4	0.04	$50nM$	2	nM
J_5	0.3	$50nM$	15	nM

Table 2. Parameters in the model

$$Cdh1 \xrightarrow{\frac{k_4 N_{Cdh1} N_{Cycb}}{J_4 V + N_{Cdh1}}} Cdh1_P \tag{27}$$

$$Cdh1_P \xrightarrow{\frac{k''_3 N_{Cdh1_P} N_{Cdc20}}{J_3 V + N_{Cdh1_P}}} Cdh1 \tag{28}$$

$$\emptyset \xrightarrow{k'_5 V + \frac{k''_5 V N^2_{Cycb}}{(J_5 V)^2 + N^2_{Cycb}}} Cdc20 \tag{29}$$

$$Cdc20 \xrightarrow{k_6 N_{Cdc20}} \emptyset \tag{30}$$

The species and parameters before and after the conversion are listed in Tables 1 and 2.

After setting up the units and getting the volume, the conversion process is simply: 1) Change the form of species from normalized concentration to population by examining their units. 2) For any parameter that has normalized concentration involved in its unit definition, change it to real concentration. 3) Change rate law equations by replacing every occurrence of a species S with $\frac{S}{V}$ and multiplying the whole rate law equation with V (refer to Equations (14-16)).

5. SOFTWARE SUPPORT

It is typical that modelers build their deterministic models with no unit information. It is difficult to input units correctly. Thus, we added a tool to JigCell to facilitate this process. In most cases, the modeler only needs to input the units of species by getting the typical (characteristic) concentration of each species from the literature, and the program will automatically fill in the units of parameters for the modeler using Equation (19).

The conversion tool consists of two components. The first is *unit checking*, which checks unit consistency inside the model and automatically fills in the units of parameters whenever they are able to be inferred. The tool prompts the modeler to input any necessary information. The second is *model conversion*, which converts the model by changing the value and units of species and parameters and adjusts the reaction rate law expression.

There are three places where JigCell will check the unit consistency inside the model. If all of these are satisfied then the model is ready to be converted, and the conversion process will guarantee the units of the converted model are still consistent. The three places are:

- Every two quantities (a parameter, a species, or the result of a sub-expression) connected by $+$ or $-$ in the rate law equation must have same units.
- All species whose value is changed by some reaction must have the same units (excluding enzymes or any modifiers because their values are not changed by the reaction).
- The unit of the calculation result of the rate law equation must be equal to the unit of the reaction rate, as in Equation (19).

Conversion takes two steps, as introduced above. The first step moves the scaling factor from the unit to the value field to change the model into real concentration. The second step changes the concentration of species into number of molecules by multiplying each species with V, and adjusting the rate law according to (16).

The conversion can be done automatically because: 1) the program can get all the information to convert the model, 2) the program can reason the correctness of the information, and 3) the conversion process can be done in polynomial time.

6. EXPERIMENTS

We used StochKit [10] to do stochastic simulation of the converted models. StochKit supports a variety of simulation methods based on Gillespie's SSA and tau-leaping algorithms. We did experiments using the basic SSA algorithm, which is the most accurate but slowest simulation method. However, so far as we know, advanced simulation algorithms based on SSA will accelerate the simulation speed by less than an order of magnitude. So our simulation results give an indication of the computational resources needed to do stochastic simulation on the cell cycle model. For Gillespie's algorithm, the space complexity is $O(N+M)$, where N is the number of species and M is the number of reactions.

StochKit can generate the trajectory of a single simulation or an ensemble result of many simulations. A single trajectory might be used to see the dynamics of the model, and the ensemble result is used to collect statistical data such as a histogram of the value of some variable at a certain time point. The ensemble simulation is easy to compute in parallel since any two cell that do not have a descendant relationship can be simulated simultaneously, which is true in most cases.

We report here the simulation results for two cell cycle models on Virginia Tech's parallel supercomputer System X. Both models are converted directly from the corresponding deterministic models. The first model is the simplified cell cycle model introduced above. The second one is a full-sized cell cycle model [3] consisting of 97 reactions.

In SBML, one can define events to change the state of the system upon triggering when some condition is met. For our simplified model, the mass of the cell is fixed and there are no events. But for the full-sized model, mass is growing and there are events defined to divide the cell. Events for the deterministic model and corresponding stochastic model should be the same. However, practically they are different due to the random nature of stochastic simulation, as illustrated in the following example. A typical deterministic event is:

```
if (A > threshold)
  then {event is triggered}
```

Here ">" means rising above a threshold.

To make the event work successfully in stochastic simulation, it has to be rewritten to tolerate the situation where the value of A oscillates around the threshold. For example:

```
if (A < minimum)
  then {minimum = A}
if (minimum < certain low value
    AND A > threshold)
  then {event is triggered; minimum = A}
```

The current version of SBML does not support using random numbers to describe random events. Thus our stochastic simulation of the cell cycle model has to divide the cell in a fixed fraction. Except for manually editing the events, all other parts of such a complex model are automatically converted by JigCell, including all the reactions, rules, species, parameters and their units.

We note that these models do not yet represent a fully realistic stochastic model, because 1) usually a realistic stochastic model also needs modifications of the reaction network itself, which is not the topic of this paper, and 2) we haven't collected all the scaling factors for these models. However, we do have most of the scaling factors for the full-sized model, which are calculated from the data given in [4] and [5]. Among 47 species in our model (different activation status of a protein are considered different species), five of them are actually introduced auxiliary variables. We do not yet know the

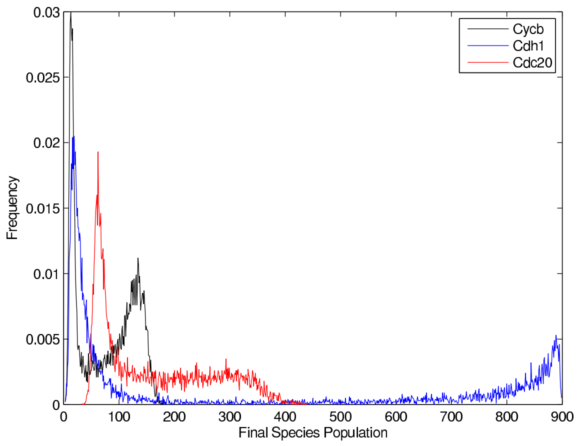

Figure 1. End point distribution of the simplified model

characteristic concentration for four of them (Cdh1, Cdh1i, Bub2 and PPX). For these species we arbitrarily set their scaling factors as 100, but all the others are computed with the right unit information found in the published literature. The volume of the cell is set to be $50fL$. Some compromises were made during calculation because there is no simple one-to-one relation between our model and their data, but all in all, this is the most detailed stochastic model we have seen and we think with such a model we can start to compare stochastic simulation results with experimental data.

The points of this experiment are 1) validate the (automatic) conversion of deterministic models to stochastic models, which is a necessary step in building realistic stochastic models, 2) demonstrate the computational resources needed to simulate these significant models and 3) explore statistical features in simulating them.

For the simplified model, we did 10,000 independent simulations of a cell starting from the same initial point (each simulation is called a run), and collected the state of the model at 200 minutes of simulation time. Fig. 1 shows the distribution of the end point state. The stochastic simulation provides a much richer description of the cell state at this time than a deterministic simulation, which would produce a single value for each species. The whole simulation took 145 seconds. We used 100 worker processors (2.3 GHz PowerPC 970FX) in parallel, each doing 100 simulation runs. The total time for all the worker processors was 12,305 seconds.

The full-sized model represents the current state-of-the-art in cell cycle simulation. We first compare the trajectories for deterministic and stochastic simulations on the converted model (see Figs. 2 and 3). Conversion to actual populations allows the modeler to examine the performance of the model not only in terms of abstract dynamics but also in real population numbers that can be compared to experimental data for easier model verification. The figures show representative species of the model as well as the mass. The two formula-

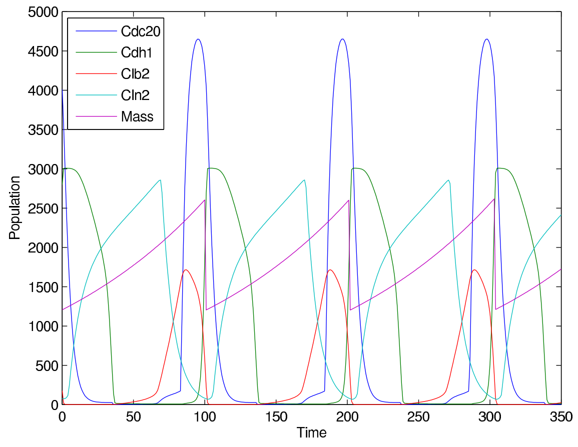

Figure 2. Deterministic simulation result of the converted full-sized model. (Mass is scaled 1000 times.)

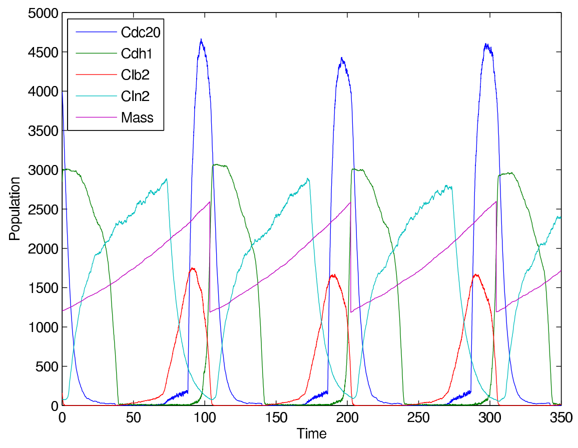

Figure 3. Stochastic simulation result of the converted full-sized model. (Mass is scaled 1000 times.)

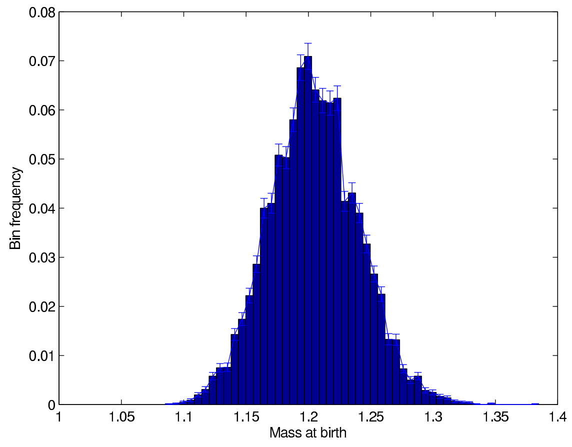

Figure 4. Histogram of mass at birth of the full-sized model

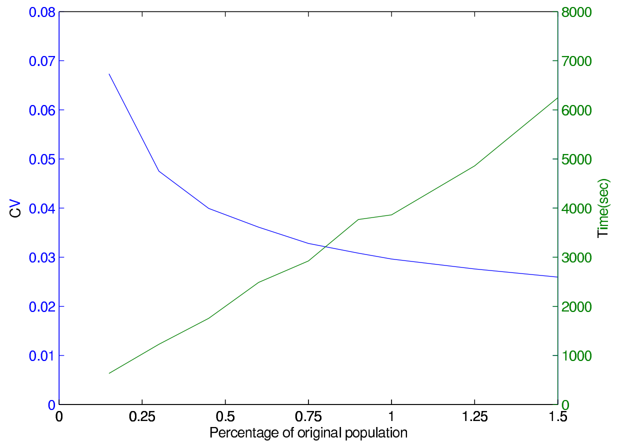

Figure 5. Coefficient of Variance of mass at birth and time spent on simulation on different populations.

tions match well. Populations for species range from below 100 molecules to over 10,000.

To examine statistical features, we did 10,000 independent simulations of the full-sized model starting from the same initial point, and collected the mass at birth after the third cell division (we discarded the first two divisions to winnow out the influence of the initial point). Fig. 4 shows the distribution of mass at birth. The mean mass at birth is 1.20, compared with 1.21 from the deterministic simulation (both numbers are normalized). The coefficient of variation (CV) of mass at birth is 2.96%. The whole simulation took 3862 seconds. We used 100 worker processors in parallel, each doing 100 simulation runs. The total time for all the worker processors were 382,267 seconds.

Noticing that both the characteristic concentration and cell volume are rough, which means the population of the stochastic model is not very accurate, we tried to perturb the population of the system to see how sensitive the simulation result is. Our method is to uniformly scale every population of the species. Fig. 5 illustrates how the result will change on different populations. The mean of mass at birth (not drawn) is quite stable. When the population becomes larger, the mean has a trend of decreasing extremely slowly from 1.21 to 1.20. Population has a larger impact on CV, which seems to be changing exponentially. Time spent on simulation grows linearly when the population gets larger. This result suggests the possibility that when the population of the system is very large one could save time by cutting the population while not hurting the statistical characteristics very much.

StochKit can be configured to use the standard C Library random number generator random() or the third-party random number generator SPRNG2.0 [12] (which is used by default). We found that these two generators will produce different simulation results. Compared with SPRNG, random() produces a long tail of outliers, which represents some rare events. The mean is 1.21 which is almost the same, but CV for

Model	Stochastic Time			Deterministic Time
	Wall	Total	Avg./run	
Simplified	145	12305	1.23	0.029
Full-sized	3862	382267	38.2	0.311

Table 3. Time for stochastic and deterministic simulation (measured in seconds)

random() becomes 6.02%, which is larger than for SPRNG. One might wonder whether SPRNG generates these same events after running long enough. We have done dozens of experiments on each random number generator with 10,000 runs for each simulation. Every time, random() will produce the outlier events, but SPRNG never does.

Table 3 shows timings for both versions of the cell cycle model, for both deterministic and stochastic simulations. LSODAR is used as the integrator for the deterministic simulation, and was run on an Intel Pentium 4 2.6GHz CPU. The deterministic simulations use the same stopping criteria as the stochastic simulations (200 time units, or the third division). In these simulations, both the deterministic and stochastic models describe the same reaction network. Thus, the comparisons give some indication for the difference in the computational resources needed. Not surprisingly, even a single run of the stochastic simulation (run on a single processor of System X) takes much more time than the deterministic simulation. However, multiple runs of the stochastic simulation (to gather ensemble statistics) can be computed in parallel.

7. CONCLUSIONS AND FUTURE WORK

It has been an open question for some time whether there are true differences between the "deterministic form" of a model and its "stochastic form". Our work on conversion of models shows that a fully specified model contains the information necessary to perform both stochastic and deterministic simulations, and that traditionally modelers have merely taken shortcuts when defining the deterministic form. As we have explained, much of our "conversion process" is in fact a requirement for full specification of all details of the model, including precise specification of units.

We introduced in this paper an automatic conversion process, which requires little expertise in biology or modeling to perform the conversion. We have performed the first stochastic simulations of a full-sized quasi-realistic cell cycle model that we are aware of. This opens the way toward studying noisy molecular level behaviors of cell cycles using stochastic models. Our next step is to make the full-sized model more realistic by enabling random division of the cell.

REFERENCES

[1] N. Allen, L. Calzone, K. Chen, A. Ciliberto, N. Ramakrishnan, C. Shaffer, J. Sible, J. Tyson, M. Vass, L. Watson, and J. Zwolak. Modeling regulatory networks at Virginia Tech. *OMICS, J. of Integrative Biol.*, 7:285–299, 2003.

[2] Y. Cao, H. Li, and L. Petzold. Efficient formulation of the stochastic simulation algorithm for chemically reacting systems. *J. Chem. Phys.*, 121:4059–67, 2004.

[3] K. Chen, L. Calzone, A. Csikasz-Nagy, F. Cross, B. Novak, and J. Tyson. Integrative analysis of cell cycle control in budding yeast. *Mol. Biol. Cell*, 15(8):3841–3862, 2004.

[4] F. R. Cross, V. Archambault, M. Miller, and M. Klovstad. Testing a mathematical model of the yeast cell cycle. *Mol. Biol. Cell*, 13(1):52–70, Jan. 2002.

[5] S. Ghaemmaghami, W.-K. Huh, K. Bower, R. W. Howson, A. Belle, N. Dephoure, E. K. O'Shea, and J. S. Weissman. Global analysis of protein expression in yeast. *Nature*, 425:737–741, Oct. 2003.

[6] D. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, 81:2340, 1977.

[7] D. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *J. Chem. Phys.*, 115:1716, 2001.

[8] M. Hucka, A. Finney, H. Sauro, and 40 additional authors. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinfo.*, 19(4):524–531, 2003.

[9] Jigcell website. `http://jigcell.biol.vt.edu`.

[10] H. Li, Y. Cao, L. Petzold, and D. Gillespie. Algorithms and software for stochastic simulation of biochemical reacting systems. *Biotechnology Progress*, 2007.

[11] G. Marlovits, C. Tyson, B. Novak, and J. Tyson. Modeling m-phase control in *xenopus* oocyte extracts: the surveillance mechanism for unreplicated dna. *Biophys. Chem.*, 72:169–184, 1998.

[12] M. Mascagni and A. Srinivasan. Algorithm 806: Sprng: A scalable library for pseudorandom number generation. *ACM Trans. Math. Soft.*, 26:436–461, 2000.

[13] M. Vass, C. Shaffer, N. Ramakrishnan, L. Watson, and J. Tyson. The JigCell Model Builder: a spreadsheet interface for creating biochemical reaction network models. *IEEE/ACM Trans. on Comp. Biol. and Bioinf.*, 2005.