Methods 51 (2010) 251–256

Contents lists available at ScienceDirect

Methods

journal homepage: www.elsevier.com/locate/ymeth

Stochastic simulation and graphic visualization of mitotic processes

Melissa K. Gardner, David J. Odde *

Department of Biomedical Engineering, University of Minnesota, Minneapolis, MN 55455, USA

ARTICLE INFO

Article history:
Accepted 19 January 2010
Available online 22 January 2010

Keywords:
Mitosis
Microtubule
Simulation
Stochastic
Imaging
Yeast

ABSTRACT

Computational modeling can be extremely useful in interpreting experimental results. Here we describe how a relatively sophisticated stochastic model for microtubule dynamic instability in the mitotic spindle can be developed starting with straightforward rules and simple programming code. Once this model is developed, the method for comparing simulation results to experimental data must be carefully considered. The ultimate utility of any computational model relies on its predictive power and the ability to assist in designing new experiments. We describe how "deconstructing" the model through the use of quantitative animations contributes to a better qualitative understanding of model behavior. By extracting key qualitative elements of the model in this fashion, model predictions and new experiments can be more easily extracted from model results.

1. Microtubule dynamics in the mitotic spindle

Proper chromosome segregation during mitosis requires precise regulation of proteins within the mitotic spindle. The alignment of chromosomes at the spindle equator during mitosis, as well as their subsequent segregation to opposite poles during anaphase, depends on the attachment of chromosomes to dynamic microtubule plus-ends. Thus, dynamic microtubule (MT) plus-ends coordinate the motion of chromosomes throughout mitosis. Through green fluorescent protein tagging and fluorescence imaging, it is possible to image MTs, chromosomes, and the kinetochores that link MT plus-ends to chromosomes [1]. Although metaphase chromosomes in some organisms appear to be statically positioned at the spindle equator, it is clear from laser photobleaching experiments that MT plus-ends remain dynamic, with their component αβ tubulin subunits turning over rapidly [2–4]. Therefore, a central question in mitosis is how MTs can remain dynamic, with rapid turnover of their component subunits, while their length is regulated such that the chromosome positions remain relatively static, aligned around the spindle equator during metaphase.

In pursuit of this question, sophisticated genetic techniques can be used to perturb microtubule-associated proteins. In many cases, mutating, depleting, or overexpressing microtubule-associated proteins results in perturbation of chromosome alignment at the spindle equator [5–8]. These experiments provide clues regarding the identity of the proteins that may be important in regulating the length of MTs to establish chromosome positions at the spindle equator during metaphase, but the chromosome misalignment phenotypes do not provide direct evidence regarding the exact nature of microtubule dynamics in the spindle and how these dynamics may be regulated by microtubule-associated proteins.

Computational simulation of dynamic microtubule plus-ends can thus be used as a framework to develop a physically realistic model for how microtubule dynamics are regulated by microtubule-associated proteins. In order to simulate the behavior of any dynamic component in the cell, specific rules governing the behavior of the cellular component must be established. These rules may be based on physical constraints within the cell or cellular component (e.g., mass or energy balances), or can simply be defined based on phenomenological observations. In the case of microtubules, the dynamic behavior at MT plus-ends has been described based on *in vitro* experiments, and is termed "dynamic instability." [9].

Dynamic instability behavior at the tips of dynamic microtubules is characterized by random switching between growth phases and shortening phases. Specifically, MT tips will tend to grow at a constant velocity (V_g) until the growing MT tip suddenly and stochastically stops growing and begins to shorten. This transition is termed a "catastrophe" event, which is characterized by the frequency of occurrence, or the "catastrophe frequency" (k_c). Once a MT tip has a catastrophe event, the MT tip will then tend to shorten at a constant velocity (V_s) until the growing MT tip suddenly and stochastically stops shortening and begins to grow again. This second transition is termed a "rescue" event, and its frequency is defined as a "rescue frequency" (k_r). Thus, four parameters – V_g, V_s, k_c, and k_r, – can completely define the dynamic instability behavior at microtubule tips. In addition, by understanding how these parameters are regulated, it is possible to develop a physically realistic model for how dynamic instability at MT tips may be regulated by microtubule-associated proteins.

* Corresponding author.
E-mail address: oddex002@umn.edu (D.J. Odde).

1046-2023/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ymeth.2010.01.021

2. Stochastic simulation of microtubule dynamics

Quantitative modeling can be used to provide a framework for understanding how the parameters of dynamic instability as described above could be regulated to control the length of microtubules during mitosis. Therefore, it is useful to understand specifically how stochastic simulation code could be written to incorporate the parameters of dynamic instability and thus predict the resulting average microtubule length. The term "stochastic" implies that the simulation will reproduce both the mean and the standard deviation of microtubule lengths, such that the experimentally observed variation between individual microtubule lengths will be an integral part of the simulation. One method to accomplish this goal is via "Monte–Carlo" modeling, which incorporates random noise into an otherwise deterministic simulation through the use of a random number generator [10–12]. A description for how the Monte–Carlo method is used to simulate microtubule dynamic instability is as follows.

1) Both for scalability of the final program to larger numbers of microtubules, and to keep track of microtubule properties such as length, position, state, etc., it is advisable to establish a data structure to keep track of microtubules in the simulation. Although there are a variety of programming languages that could be used to simulate microtubules, the code shown here is compatible with MATLAB programming language. Note that the syntax will vary depending on the selected programming language. Therefore, in MATLAB a microtubule data structure would be established for one microtubule as follows. Note that text after the "%" sign is simply for commenting and annotation. In addition, see Fig. 1A for a depiction of the coordinate system used in this programming code.

```
Microtubule(1).length=XX  %starting length of microtubule μm
Microtubule(1).state=0  %starting state(0=growing, 1=shortening)
Microtubule(1).pole=0  %spindle pole attach side (0=left; 1=right)
Microtubule(1).ypos=XX  %y-coordinate pole attach point
Microtubule(1).zpos=XX  %z-coordinate pole attach point
```

2) Then, constant values for the parameters of dynamic instability need to be established, as follows:

```
Vg=2  %growth velocity in μm/min
Vs=2  %shortening velocity in μm/min
kc=15  %catastrophe frequency in min^-1
kr=15  %rescue frequency in min^-1
```

3) In addition, a value for the spindle length itself must be established, so that boundary conditions can be imposed on the growth of microtubules:

```
Spindle_length=2  %spindle length in μm
```

4) Finally, the time step and simulation duration need to be established. The time step is an important simulation parameter. To prevent convergence of the simulation to an incorrect value, the time step must be small enough to ensure that both catastrophe and rescue events are rare (<10% probability).

```
Duration=20  %duration in minutes
Tau = 0.0002%time step in minutes
```

5) Now the dynamic instability of a single microtubule attached to one pole in a spindle can be simulated via the Monte–Carlo method. Following are descriptions of core code lines that are used to simulate the dynamic instability of microtubules:

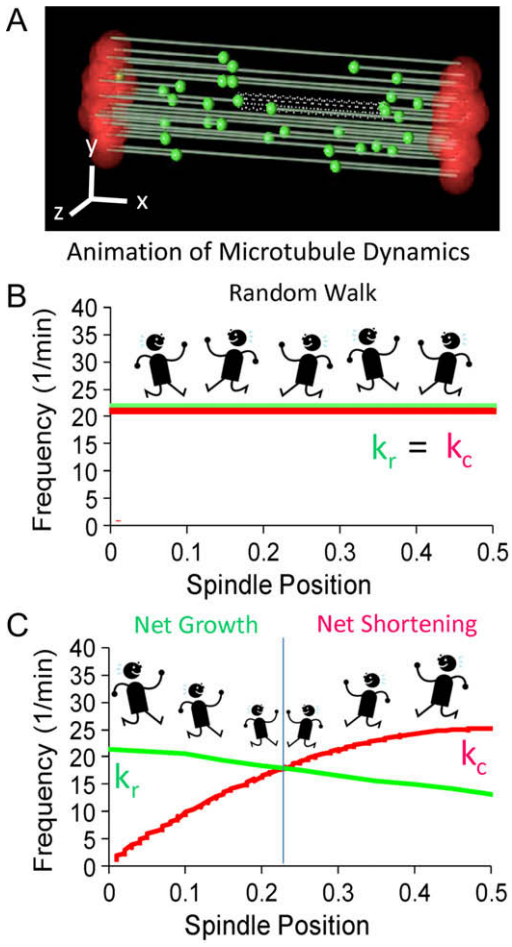

Fig. 1. Simulation of microtubule dynamics in the mitotic spindle. (A) A typical animated output of simulated spindle microtubule dynamics. The coordinate system for assignment of microtubules to spindle locations (y–z planes) and for quantification of spindle microtubule length (x axis) is shown in the lower left. (B) With equal catastrophe (k_c) and rescue (k_r) frequency, simulations of spindle microtubule dynamics result in a uniform distribution of microtubule lengths in the spindle. (C) Spatial regulation of k_r and k_c controls the mean length of microtubules in the spindle. In this simulation, the most frequent microtubule length is at spindle position ~0.22.

For a growing MT, the code to increase MT length from time step #1 to time step #2 is:

```
Microtubule.length(2)=
Microtubule.length(1)+ Vg*Tau
```

For a shortening MT, the code to decrease MT length from time step #1 to time step #2 is:

```
Microtubule.length(2)=
Microtubule.length(1)- Vs*Tau
```

Transitions between states are calculated by converting the frequency of transitions into a probability according to a first-order process, as follows:
For rescue:

```
Pr_rescue = 1-exp(-kr*Tau)
```

For catastrophe:

```
Pr_catastrophe = 1-exp(-kc*Tau)
```

Then, a uniformly distributed random number between 0 and 1 is generated to determine whether or not a transition occurs. In MATLAB, this is a built-in function called "rand", so:

```
Test_transition = rand
```
A transition will occur if the transition probability is less than
```
Test_transition.
```

6) All of the above described code is put together in a logical order in time, such that at each time step a growing microtubule will add length and then decide whether a catastrophe occurs, while a shortening microtubule will lose length and then decide whether a rescue occurs. All code is put into a "for" loop, which allows for repetition of all code until the simulation duration is complete. Thus, the following is a sample of complete code for simulation of one dynamic microtubule over the duration of time duration.

```
  for i = 1: duration/Tau
    if Microtubule.state==0 %Microtubule is
growing
     Microtubule.length(i + 1)=Microtubule.
length (i)+ Vg*Tau
     Pr_catastrophe = 1-exp(-kc*Tau) %Proba-
bility of catastrophe
       Test_transition = rand %calculate uni-
form random number
       if Pr_catastrophe < rand%is pr less than
random number?
       Microtubule.state = 1%Microtubule short-
ening next step
     end
    else %Microtubule state is shortening
       Microtubule.length(i+1)=Microtubule.
length (i)- Vs*Tau
       Pr_rescue = 1-exp(-kr*Tau)%Probability
of catastrophe
       Test_transition = rand %calculate uni-
form random number
       if Pr_rescue < rand %is pr less than
random number?
         Microtubule.state = 0%Microtubule
growing next step
       end
    end
  end
```

3. Boundary conditions

The above described code allows for stochastic growth and shortening phases of microtubule growth, thus simulating dynamic instability behavior at a cellular-level scale according to the parameters of dynamic instability that are defined within the computer program. Thus, at the completion of the program, the microtubule will achieve a final length that will vary each time the computer program is run. However, the program as written above places no constraints on the final length of simulated microtubules – microtubules could grow very long, or even achieve a negative length that has no physical corollary. To prevent physically unreasonable simulated microtubule lengths, it is advisable to place "boundary conditions" within the mitotic spindle simulation [13,14]. For example, if a microtubule depolymerizes to zero length, it will not depolymerize further, and a reasonable rule would be that the microtubule would then switch back to a polymerizing state. Code to establish this boundary condition would be placed at the start of each time step, and could be written as follows:

```
if Microtubule.length(i)<= 0;
  Microtubule.length(i) = 0;
   Microtubule.state=0 %Microtubule switch to
growing state
end
```

Similarly, it may be advisable to limit the longest length of each microtubule. If microtubules are not generally experimentally observed to be longer than the length of the spindle, one suggestion would be to limit the maximum microtubule length to be the length of the spindle, as follows.

```
if Microtubule.length(i)>= Spindle_length;
    Microtubule.length(i) = Spindle_length;
    Microtubule.state = 1 %Microtubule switch to
shortening state
end
```

4. Scale-up to multiple microtubules

In general, multiple microtubules will be simulated for a given mitotic spindle. For example, in budding yeast there are ~16 kinetochore microtubules attached to each pole. Once a data structure is established as described above, it is straightforward to scale up the simulation for multiple microtubules. Following is an example of code to scale the simulation up to 16 microtubules.To establish initial conditions:

```
for mt_numb = 1: 16
  Microtubule(mt_numb).length(i)=XX %starting
length of microtubule μm
  Microtubule(mt_numb).state=0 %starting
state(0= growing, 1=shortening)
  Microtubule(mt_numb).pole=0 %spindle pole
attach side (0=lft; 1=rht)
  Microtubule(mt_numb).ypos=XX %y-coordinate
pole attach point
  Microtubule(mt_numb).zpos=XX %z-coordinate
pole attach point
end

To run a simulation:

for i = 1: duration/Tau
  for mt_numb = 1: 16
      .
    <Code here as described in above
    sections, substituting
    Microtubule(mt_numb).XXX for
    Microtubule.XXX in all instances>
      .
  end
end
```

5. Models for the spatial regulation of dynamic instability

The MATLAB code as described above simulates the length progression in time of microtubules attached to a pole in the mitotic spindle. This code is based on the four parameters of dynamic instability, and the values of these parameters can be varied in simulation to better understand their effect on the length distribution of microtubules in the mitotic spindle. Because this is a stochastic simulation, replicates of the simulation for each parameter set should be run so that both a mean and a standard deviation for microtubule lengths can be calculated. Note that because of the randomness introduced into the simulation via the Monte–Carlo technique, the results from

a single run can be misleading. In our previous work, we found that for constant and equal parameters of dynamic instability (i.e., $V_g = V_s$, and $k_c = k_r$), the microtubule plus-ends exhibit a random walk along the length of the spindle, such that there is no bias of the MTs towards net growth or shortening (Fig. 1B)[15,16]. In this case, given a large number of simulations, the mean length of microtubules will be approximately half of the spindle length, with a relatively large standard deviation. An interesting question is then to explore how microtubule dynamics may be regulated *in vivo* to result in the relatively precise congression of chromosomes that is observed during metaphase in many organisms. One way that microtubule dynamics may be regulated is if one or more parameters of dynamic instability were spatially regulated within the mitotic spindle. For example, microtubule dynamics may change as a function of microtubule length, which would narrow the distribution of lengths [5,6,8,17,18]. Conversely, microtubule dynamics could be regulated as a function of distance away from a spindle pole or the spindle equator, as in a polar ejection force model, or regulated via a spatial concentration or phosphorylation gradient of a molecule that regulates microtubule dynamics, such as Op18/stathmin or Ran-GTP [19–22]. The mathematical form for how a given parameter of dynamic instability varies spatially within the mitotic spindle could be simple or complex, depending on the assumptions made in building the model [16,23–25]. In the example code provided below, it is assumed that the catastrophe frequency (k_c) varies linearly as a function of the length of the microtubule, while the remainder of the dynamic instability parameters remain constant. In this example, it is assumed that $k_c = 0.25\ \text{min}^{-1}$ when a microtubule length = 0 μm, and that k_c increases linearly as a function of microtubule length with slope $30\ \mu\text{m}^{-1}\text{min}^{-1}$. Therefore, k_c varies spatially as a function of microtubule length according to $k_c = 30\,l + 0.25$, where l is microtubule length in μm. Using this function, the code for calculating catastrophe frequency would be as follows at each time step:

```
kc = ((Microtubule(mt_numb).length(i))*30) +
0.25
```

and, as before:

```
  Pr_catastrophe = 1-exp(-kc*Tau)  %Probability
of catastrophe
  Test_transition = rand  %generate random num-
ber between 0 and 1
  if Pr_catastrophe < rand  %is pr less than ran-
dom number?
  Microtubule(mt.numb).state=1     %Microtubule
shortening next step
end
```

By spatially varying catastrophe frequency, MT ends do not exhibit a random walk along the length of the spindle, but rather are attracted to the specific points within the spindle where $k_c = k_r$ (i.e., points where catastrophe frequency is balanced by rescue frequency, so there is no drive towards net growth or shortening). This type of situation is illustrated in Fig. 1C, where both catastrophe and rescue frequency vary spatially along the length of the spindle, and the most likely microtubule length is at spindle position 0.22.

Thus, with the simple programming code described above, many different models for the regulation of microtubule dynamics within the spindle can be tested.

6. Reporting results

The stochastic simulation described above results in a distribution of microtubule lengths at each time step in the simulation. Note that the starting values for the microtubule lengths (at time 0), may have no relation to the final steady-state length distribution, so it is important that a "warm-up" period is allowed to achieve steady-state prior to recording results. Once steady-state is achieved, the simulated mean microtubule length and standard deviation should remain constant, with no net trend upward or downward regardless of the simulation duration.

The simplest method to report results will be to calculate the average microtubule length and standard deviation at the end of the simulation. At the end of the simulation, all microtubule lengths can be recorded into a single array, as follows.

```
MT_lengths = [Microtubule(:).length(i)]
```

Here, the colon operator (:) selects all of the microtubules, and (i) represents the value at the last time point in the simulation (once the duration loop is complete). Similar to various other programming languages, MATLAB has built-in functions that will allow for calculation of statistics on the microtubule length array. MATLAB Code for calculating the mean and standard deviation of the microtubule length array are as follows:

```
Average_MT_length = mean(MT_lengths)
Std_Dev_MT_length = std(MT_lengths)
```

In addition, a histogram can be generated to review the distribution of final microtubule lengths using the following lines of MATLAB code.

```
figure;
hist(MT_lengths)
```

An example of a typical histogram output from MATLAB is shown in Fig. 2A.

7. Comparison to experimental results

Simulations of microtubule dynamic instability provide a framework for hypothesis testing to better understand how microtubule dynamics are regulated during mitosis. Using the simple programming code described above, different microtubule length distributions can be simulated according to various rules for the parameters of dynamic instability. Thus, by comparing the results of these simulations to experimental data, a better understanding of *in vivo* microtubule dynamics is possible. An important step in this type of hypothesis testing is to devise an accurate method for comparing simulation results to *in vivo* experiments.

The ability to accurately and quantitatively compare simulation results to experimental results is a critical step in gaining new insights into cellular processes through modeling. Therefore, this step requires careful and creative thinking. In some systems, it is possible to directly measure microtubule lengths and length distributions via direct observation of fluorescent kinetochore and spindle pole proteins [8], although multiple attachments of microtubules to kinetochores may complicate this analysis. Another alternative is to use fluorescently tagged plus-end binding proteins, and/or to tag the microtubule lattice itself with fluorescent tubulin. Here, *in vivo* microtubule length distributions can be directly measured.

In budding yeast cells, the small size of the mitotic spindle combined with the high density of microtubules and kinetochores within the spindle leads to difficulties in resolving individual microtubules or kinetochores with fluorescence microscopy. In this case, we have developed a method termed "model-convolution" in which simulated images of fluorescent proteins are convolved with

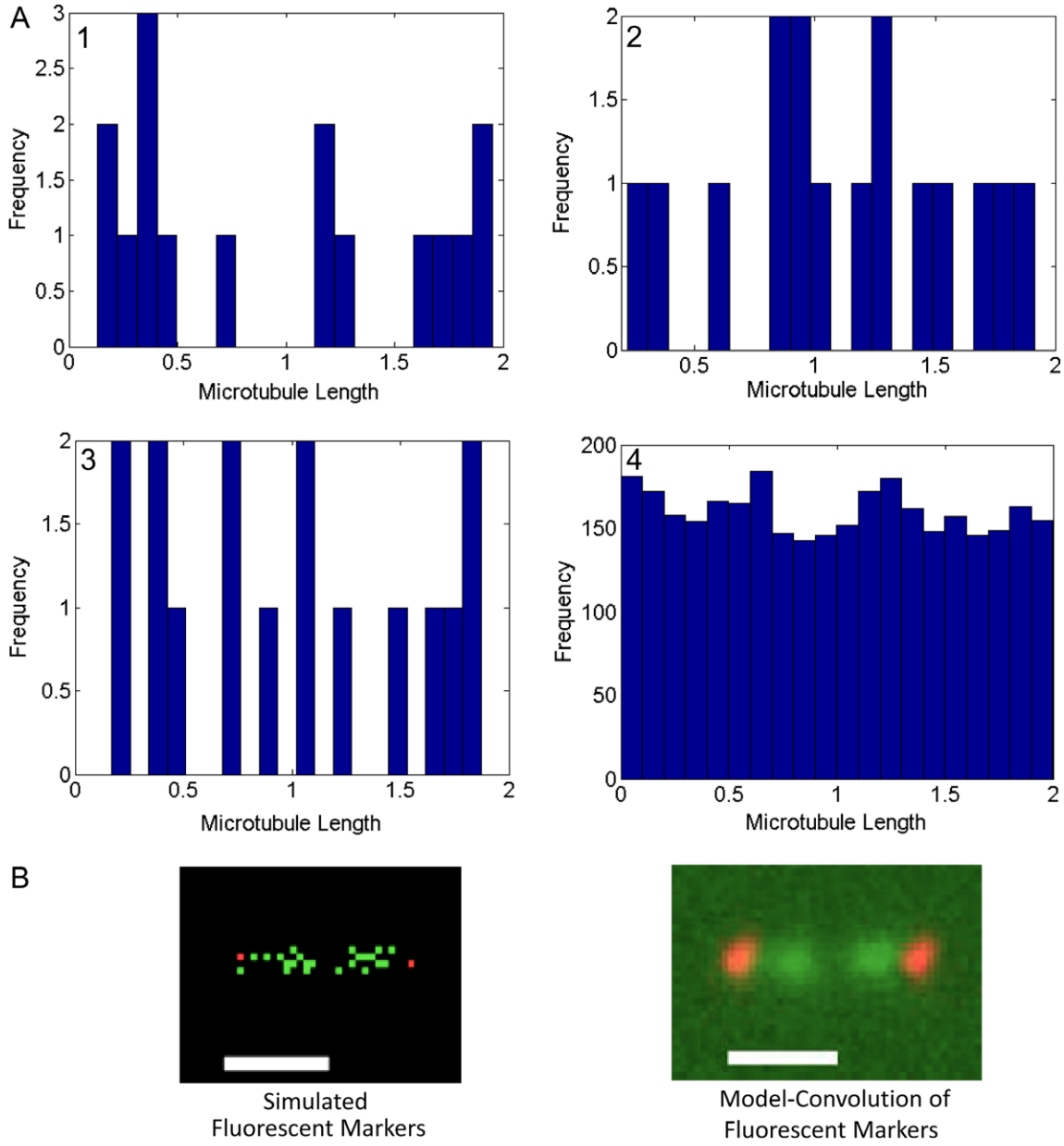

Fig. 2. Methods for comparing simulation results to experiments. (A) Histograms showing microtubule lengths at the conclusion of MT dynamics simulations. In panes 1–3 are results for single runs with 16 kinetochore microtubules ($k_c = k_r$), while pane 4 shows results averaged over 200 runs with 16 kinetochore microtubules ($k_c = k_r$). (B) Model-convolution can be used to predict fluorescence distribution patterns for closely spaced fluorescent proteins.

the experimentally measured microscope point spread function to allow for a direct quantitative comparison between simulated and experimental images (Fig. 2B)[1,16]. An Image J (National Institutes of Health) plug-in is available for performing this type of analysis on www.umn.edu/~oddex002/.

Regardless of the approach, an essential component of hypothesis testing through computational modeling is the collection of high quality quantitative experimental data that can be directly compared to simulation results.

8. Deconstructing the model: Graphic visualization

Once a model is developed that can reproduce experimental results, the utility of the model lies in (1) providing an informative framework for better understanding experimental results and (2) providing predictions that lead to further experiments for hypothesis and model testing. Although building a model is in itself an informative experience, the ultimate test of the model will be in its predictive power. We have found that, regardless of the model's success in matching experimental results, a key factor in the ultimate success of the modeling effort is the ability to "deconstruct" the model. By deconstructing the model, key parameters and rules in the model can be distinguished from less important factors. Importantly, how the parts of the system affect the overall process are better understood by directly observing model results upon perturbation of parameters or model assumptions. For example, by varying one parameter of dynamic instability (e.g., by varying the value of *Vg* from a low value to a very high value), the effect of this parameter on the overall MT length distribution is determined.

One powerful method for deconstructing a model is to develop a graphic visualization of the model results (Fig. 1A). By studying quantitative animations of model results, it is possible to directly observe the simulated interactions of model components, which

leads to an improved understanding of how the model is working and can provide insights that will lead to new experiments. For example, by selecting parameters of dynamic instability that lead to long microtubules in the yeast spindle simulation described above, it is clear from simulation animations that experimental MT-associated fluorescence would be highest in the middle of the spindle due to an excess of microtubules from both poles crossing the spindle equator. Experiments with GFP-tubulin show that MT-associated fluorescence is low at the spindle equator, thus strongly suggesting that microtubules are short and do not frequently cross the spindle equator.

Although many programming languages have a built-in capability to allow for graphic visualization, we have found that the most efficient method for graphic rendering of our simulations is through the use of data visualization software that can interface with the numerical simulation output. One freely available shareware that is available for this purpose is OpenDX (www.openDX. org). To render animations using this software, text files are written during the spindle simulation that include the 3D locations of every object in the simulation (i.e., spindle poles, microtubule subunits, kinetochores, etc.) at selected time points in the simulation. In addition, a state variable is written to the text file that allows for further encoding of information into the final animation (e.g., a 0 or 1 could be written to the text file for a given microtubule depending on whether the microtubule is in a growing or a shortening state). Once these files are written, they are read in by OpenDX and transformed into an animated movie (Fig. 1A). The OpenDX software relies on a Graphical User Interface that allows the user to select built-in software components for animating objects. These built-in components are selected and then manually connected to create a program for animating results. The properties of each component can be modified by selecting the box, which brings up a menu of options for each. In general, OpenDX is a simple, free software with adequate help available for on-line training and debugging.

9. Conclusions

Stochastic computational modeling of mitotic processes allows for rigorous hypothesis testing and can be extremely useful in designing experiments. We demonstrate here that a relatively sophisticated stochastic model for microtubule dynamic instability within the mitotic spindle can be developed starting with simple rules and straightforward programming code. Once a model is developed, it is important that the model results are accurately compared to experimental results to assess the validity of the model. The ultimate test of the model's utility will be its predictive power and its effectiveness in generating new ideas for further experimentation. We have found that a qualitative understanding of the model behavior, or a "deconstruction" of the model results, is critical to this effort. Importantly, the insights gained from studying animated movies that are generated from quantitative simulations can be at least as informative as graphs and other quantitative measures in the effort to gain a better understanding of key aspects of a biological process.

References

[1] M.K. Gardner, D.J. Odde, K. Bloom, Methods 41 (2007) 232–237.
[2] P. Maddox, K. Bloom, E.D. Salmon, Nat. Cell Biol. 2 (2000) 36–41.
[3] P. Maddox, A. Straight, P. Coughlin, T.J. Mitchison, E.D. Salmon, J. Cell Biol. 162 (2003) 377–382.
[4] C.G. Pearson, M.K. Gardner, L.V. Paliulis, E.D. Salmon, D.J. Odde, K. Bloom, Mol. Biol. Cell 17 (2006) 4069–4079.
[5] M.K. Gardner, D.C. Bouck, L.V. Paliulis, J.B. Meehl, E.T. O'Toole, J. Haase, A. Soubry, A.P. Joglekar, M. Winey, E.D. Salmon, K. Bloom, D.J. Odde, Cell 135 (2008) 894–906.
[6] M.I. Mayr, S. Hummer, J. Bormann, T. Gruner, S. Adio, G. Woehlke, T.U. Mayer, Curr. Biol. 17 (2007) 488–498.
[7] J.R. McIntosh, E.L. Grishchuk, R.R. West, Annu. Rev. Cell Dev. Biol. 18 (2002) 193–219.
[8] J. Stumpff, G. von Dassow, M. Wagenbach, C. Asry, L. Wordeman, Dev. Cell 14 (2008) 252–262.
[9] T. Mitchison, M. Kirschner, Nature 312 (1984) 237–242.
[10] D.T. Gillespie, J. Phys. Chem. 81 (1977) 2340–2361.
[11] P. Bayley, M. Schilstra, S. Martin, FEBS Lett. 259 (1989) 181–184.
[12] Y.D. Chen, T.L. Hill, Proc. Natl. Acad. Sci. USA 82 (1985) 1131–1135.
[13] I.V. Gregoretti, G. Margolin, M.S. Alber, H.V. Goodson, J. Cell Sci. 119 (2006) 4781–4788.
[14] G. Margolin, I.V. Gregoretti, H.V. Goodson, M.S. Alber, Phys. Rev. E Stat. Nonlin. Soft. Matter. Phys. 74 (2006) 041920.
[15] M.K. Gardner, C.G. Pearson, B.L. Sprague, T.R. Zarzar, K. Bloom, E.D. Salmon, D.J. Odde, Mol. Biol. Cell 16 (2005) 3764–3775.
[16] B.L. Sprague, C.G. Pearson, P.S. Maddox, K.S. Bloom, E.D. Salmon, D.J. Odde, Biophys. J. 84 (2003) 1–18.
[17] J. Howard, A.A. Hyman, Curr. Opin. Cell Biol. 19 (2007) 31–35.
[18] V. Varga, J. Helenius, K. Tanaka, A.A. Hyman, T.U. Tanaka, J. Howard, Nat. Cell Biol. 8 (2006) 957–962.
[19] G.J. Brouhard, A.J. Hunt, Proc. Natl. Acad. Sci. USA 102 (2005) 13903–13908.
[20] C.L. Rieder, E.A. Davison, L.C. Jensen, L. Cassimeris, E.D. Salmon, J. Cell Biol. 103 (1986) 581–591.
[21] P. Kalab, R. Heald, J. Cell Sci. 121 (2008) 1577–1586.
[22] L. Cassimeris, Curr. Opin. Cell Biol. 14 (2002) 18–24.
[23] G. Civelekoglu-Scholey, D.J. Sharp, A. Mogilner, J.M. Scholey, Biophys. J. 90 (2006) 3966–3982.
[24] A.P. Joglekar, A.J. Hunt, Biophys. J. 83 (2002) 42–58.
[25] J. Liu, A. Desai, J.N. Onuchic, T. Hwa, Proc. Natl. Acad. Sci. USA 105 (2008) 13752–13757.