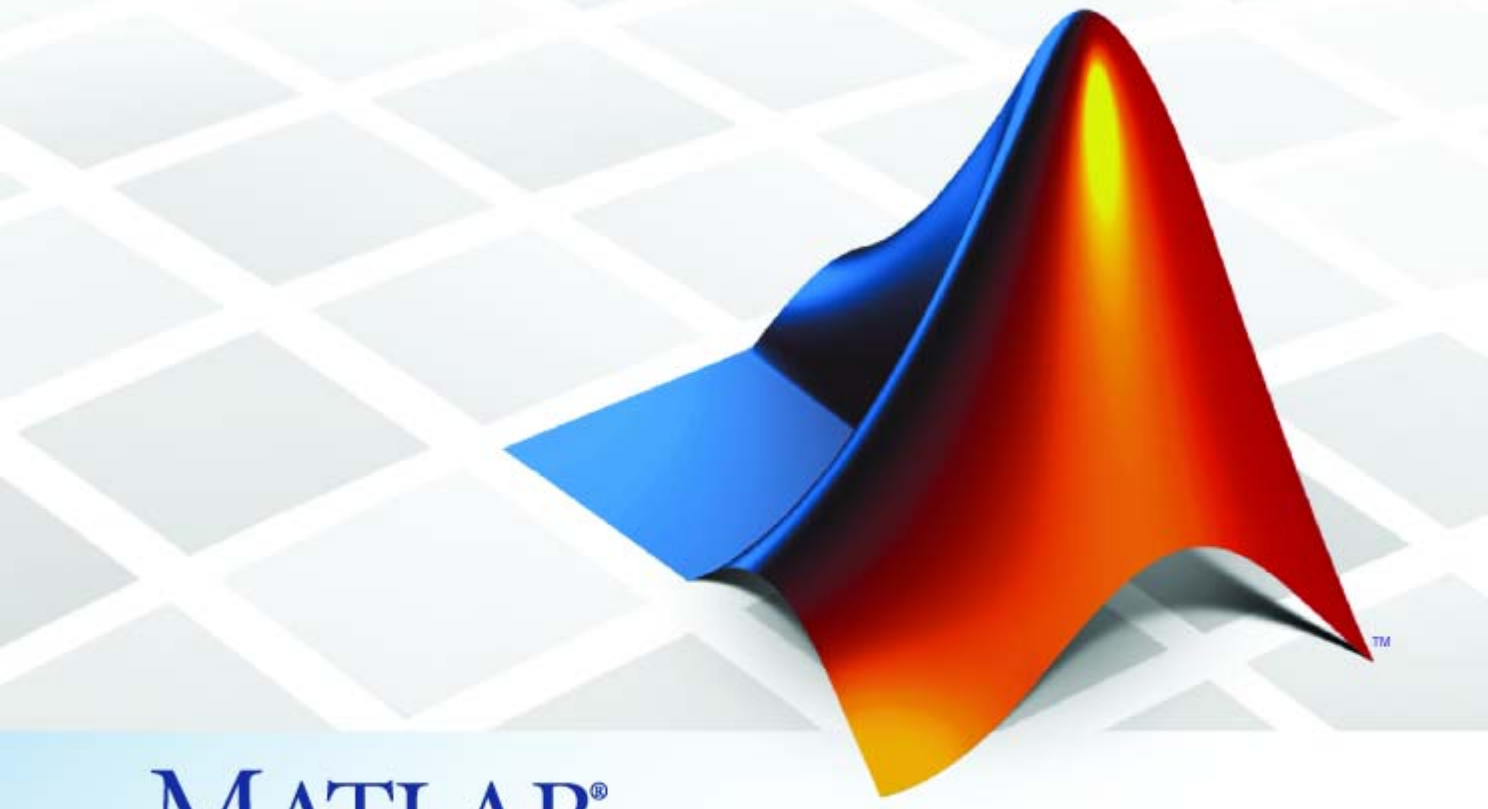
SimBiology® 2
Model Reference



MATLAB®

The MathWorks™
Accelerating the pace of engineering and science

How to Contact The MathWorks

`www.mathworks.com`	Web
`comp.soft-sys.matlab`	Newsgroup
`www.mathworks.com/contact_TS.html`	Technical Support

`suggest@mathworks.com`	Product enhancement suggestions
`bugs@mathworks.com`	Bug reports
`doc@mathworks.com`	Documentation error reports
`service@mathworks.com`	Order status, license renewals, passcodes
`info@mathworks.com`	Sales, pricing, and general information

508-647-7000 (Phone)

508-647-7001 (Fax)

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098

For contact information about worldwide offices, see the MathWorks Web site.

SimBiology® Model Reference

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See `www.mathworks.com/trademarks` for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

The MathWorks products are protected by one or more U.S. patents. Please see `www.mathworks.com/patents` for more information.

Revision History

September 2005	Online only	New for Version 1.0 (Release 14SP3+)
March 2006	Online only	Updated for Version 1.0.1 (Release 2006a)
May 2006	Online only	Updated for Version 2.0 (Release 2006a+)
September 2006	Online only	Updated for Version 2.0.1 (Release 2006b)
March 2007	Online only	Rereleased for Version 2.1.1 (Release 2007a)
September 2007	Online only	Rereleased for Version 2.1.2 (Release 2007b)
October 2007	Online only	Updated for Version 2.2 (Release 2007b+)
March 2008	Online only	Updated for Version 2.3 (Release 2008a)
October 2008	Online only	Updated for Version 2.4 (Release 2008b)

Contents

1 Minimal Cascade Model for a Mitotic Oscillator

2 Model of the Yeast Heterotrimeric G Protein Cycle

3 M-Phase Control in *Xenopus* Oocyte Extracts

Index

1

Minimal Cascade Model for a Mitotic Oscillator

Albert Goldbeter modified a model with enzyme cascades [Goldbeter and Koshland 1981] to fit cell cycle data from studies with embryonic cells [Goldbeter 1991]. He used this model to demonstrate thresholds with enzyme cascades and periodic behavior caused by negative feedback.

There are two SimBiology® model variations using Goldbeter's model. The first model uses the differential rate equations directly from Goldbeter's paper. The second model is built with reactions using Henri-Michaelis-Menten kinetics.

Goldbeter Model

In this section...

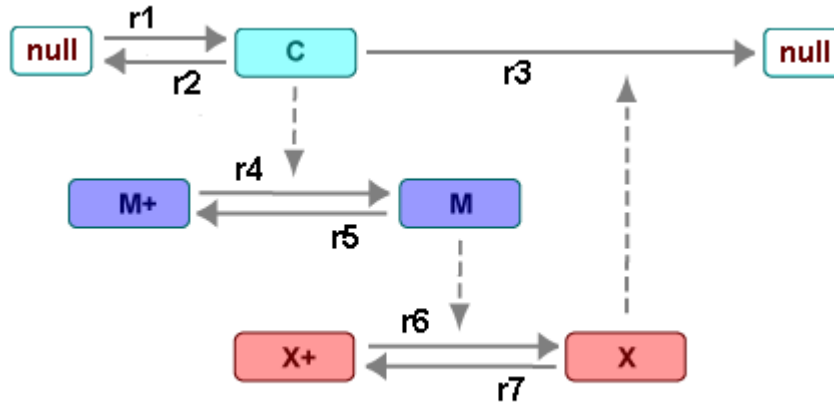
About the Goldbeter Model

Albert Goldbeter created a simple cell division model from studies with embryonic cells [Goldbeter 1991]. This model demonstrates thresholds with enzyme cascades and periodic behavior caused by negative feedback.

There are six species in Goldbeter's minimal mitotic oscillator model [Goldbeter 1991].

- C — Cyclin. The periodic behavior of cyclin activates and deactivates an enzyme cascade.
- M+, M — Inactive (phosphorylated) and active forms of cdc2 kinase. Kinases catalyze the addition of phosphate groups onto amino acid residues.
- X+, X — Inactive and active (phosphorylated) forms of a cyclin protease. Proteases degrade proteins by breaking peptide bonds.

The reactions are labeled `r1` to `r7` on the following diagram.



This model shows:

- How thresholds with cdc2 kinase activation (M+ -> M) and protease activation (X+ -> X) can occur as the result of covalent modification (for example, phosphorylation or dephosphorylation), but without the need for positive feedback.
- How periodic behavior with cdc2 kinase activation can occur with negative feedback and the time delay associated with activation/deactivation enzyme cascades.

Reaction Descriptions and Model Assumptions

The following list describes each of the reactions in Goldbeter's minimal mitotic oscillator with some of the simplifying assumptions. For a more detailed explanation of the model, see [Goldbeter 1991].

- Cyclin (`C`) is synthesized at a constant rate (r1) and degraded at a constant rate (r2).
- Cyclin (`C`) does not complex with cdc2 kinase (`M`).
- Cyclin (`C`) activates cdc2 kinase (M+ -> M) by increasing the velocity of the phosphatase that activates the kinase. Inactive cdc2 kinase (`M+`) is activated by removing inhibiting phosphate groups (r4).

- The amount of deactivating kinase (not modeled) for the cdc2 kinase (`M`) is constant. Active cdc2 kinase (M) is deactivated by adding inhibiting phosphate group (r5).
- The activation of cyclin protease (`X+ -> X`) by the active cdc2 kinase (`M`) is direct without other intervening cascades. Cyclin protease (`X`) is activated by adding phosphate groups (r6).
- The amount of deactivating phosphatase (not modeled) for the cyclin protease (X) is constant. Active cyclin protease (`X`) is deactivated by removing the activating phosphate groups (r7).
- The three species of interest are cyclin (`C`), active dephosphorylated cdc2 kinase (`M`), and active phosphorylated protease (`X`). The total amounts of (`M + M+`) and (`X + X+`) are constant.

Mathematical Model

Goldbeter's minimal mitotic oscillator model is defined with three differential rate equations and two algebraic equations that define changing parameters in the rate equations.

Differential Rate Equation 1, Cyclin (C)

The following differential rate equation is from [Goldbeter 1991] for cyclin (C).

$$\frac{dC}{dt} = vi - vdX\frac{C}{Kd + C} - kdC$$

Differential Rate Equation 2, Kinase (M)

The following differential rate equation is for cdc2 kinase (`M`). Notice that (`1 - M`) is the amount of inactive (phosphorylated) cdc2 kinase (`M+`).

$$\frac{dM}{dt} = V1\frac{(1-M)}{K1+(1-M)} - V2\frac{M}{K2+M}$$

$$V1 = \frac{VM1[C]}{Kc+[C]}$$

Differential Rate Equation 3, Protease (X)

Differential rate equations for cyclin protease (X). Notice that `(1 - X)` is the amount of inactive (unphosphorylated) cyclin protease (X+).

$$\frac{dX}{dt} = V3\frac{(1-X)}{K3+(1-X)} - V4\frac{X}{K4+X}$$

$$V3 = VM3[M]$$

SimBiology Model with Rate Rules

In this section...

SimBiology Model with Rules

In the literature, many biological models are defined using differential rate and algebraic equations. With SimBiology software, you can enter the equations directly as SBML rules. The example in this section uses Goldbeter's mitotic oscillator to illustrate this point.

Writing differential rate equations in an unambiguous format that a software program can understand is a fairly simple process.

- Use an asterisk to indicate multiplication. For example, `k[a]` is written `k*a`.
- Remove square brackets that indicate concentration from around species. The units associated with the species will indicate concentration (`moles/liter`) or amount (`moles`, `molecules`).

  SimBiology software uses square brackets around species and parameter name to allow names that are not valid MATLAB® variable names. For example, you could have a species named `glucose-6-phosphate dehydrogenase` but you need to add brackets around the name in reaction rate and rule equations.
- Use parentheses to clarify the order of evaluation for mathematical operations. For example, do not write a Henri-Michaelis-Menten rate as `Vm*C/Kd + C`, because `Vm*C` is divided by `Kd` before adding `C`, and then `C` is added to the result.

The following equation is the rate rule for "Differential Rate Equation 1, Cyclin (C)" on page 1-4:

```
dC/dt = vi - (vd*X*C)/(Kd + C) - kd*C
```

The following equations are the rate and algebraic rules for "Differential Rate Equation 2, Kinase (M)" on page 1-4:

```
dM/dt = (V1*Mplus)/(K1 + Mplus) - (V2*M)/(K2 + M)
V1 = (VM1*C)/(Kc + C)
Mplus = Mt - M
```

The following equations are the rate and algebraic rules for "Differential Rate Equation 3, Protease (X)" on page 1-5:

```
dX/dt = (V3*Xplus)/(K3 + Xplus) - (V4*X)/(K4 + X)
V3 = VM3*M
Xplus = Xt - X
```

Species

The following table is a list of species in the model with their initial amounts.

The two parameters `V1` and `V3` are in the species list. You could enter the parameters in the parameter table with the `ConstantAmount` check boxes cleared. Here, the parameters are modeled as species but without reactions.

Name	InitialAmount	ConstantAmount	BoundaryCondition
C	0.01	☐	☐
M	0.01	☐	☐
Mplus	0.99	☐	☐
Mt	1.0	☐	☐
X	0.01	☐	☐
Xplus	0.99	☐	☐
Xt	1.0	☐	☐
V1	0.0	☐	☐
V3	0.0	☐	☐

Parameters

The following table is a list of parameters in the model with their initial values. The **ConstantValue** property is selected for all the parameters.

Name	Value	ConstantValue
vi	0.025	☑
kd	0.01	☑
vd	0.25	☑
Kd	0.02	☑
VM1	3.0	☑
K1	0.0050	☑
Kc	0.5	☑
V2	1.5	☑
K2	0.0050	☑
VM3	1.0	☑
K3	0.0050	☑
V4	0.5	☑
K4	0.0050	☑

Rules

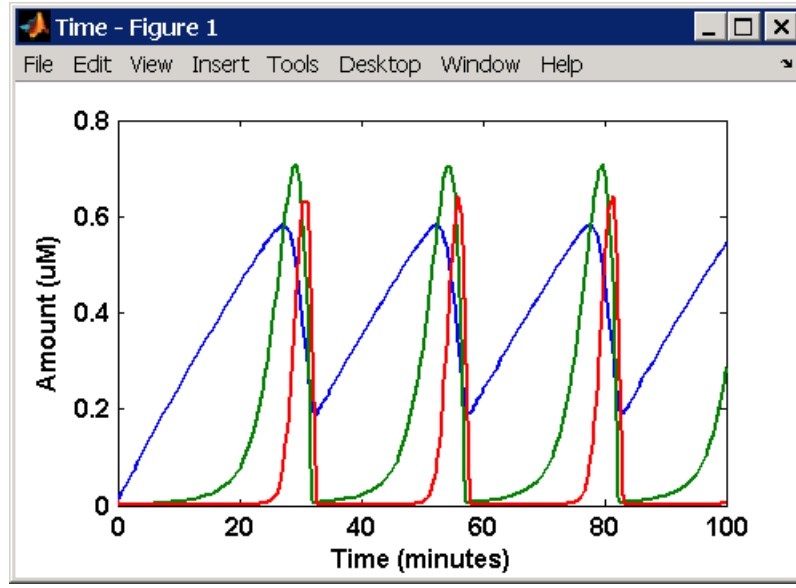
The active (`M`) and inactive (`Mplus`) forms of the kinase are assumed to be part of a conserved cycle with the total concentration (Mt) remaining constant during the simulation. You need only one differential rate equation with a mass balance algebraic equation to define the amounts of both species. Similarly, the active (`X`) and inactive (`Xplus`) forms of the protease are part of a second conserved cycle.

In the SimBiology desktop, you enter rate rules of the form `dX/dt = Expression` as `X = Expression`, and algebraic rules of the form `"X = Expression"`, where X is the independent variable, as `Expression - X`.

Name	Rule	RuleType
Cyclin	C = vi - (vd*X*C)/(Kd + C) - kd*C	rate
Kinase	M = (V1*Mplus)/(K1 + Mplus) - (V2*M)/(K2 + M)	rate
	Mt - Mplus - M	algebraic
	(VM1*C)/(Kc + C) - V1	algebraic
Protease	X = (V3*Xplus)/(K3 + Xplus) - (V4*X)/(K4 + X)	rate
	Xt - Xplus - X	algebraic
	VM3*M - V3	algebraic

SimBiology Simulation with Rules

This is a simulation of Goldbeter's minimal mitotic oscillator using differential rate and algebraic equations. Simulate with the `ode15s` solver and plot species `C`, `M`, and `X`. For a description of the model, see "SimBiology Model with Rules" on page 1-6.



SimBiology Model with Reactions

In this section...

Converting Differential Rate Equations to Reactions

In the literature, many models are defined with differential rate equations. With SimBiology software, creating the differential equations from reactions is unnecessary; you can enter the reactions and let the software calculate the equations.

Some models are defined with differential rate equations, and you might need the reactions to be compatible with your model. Two rules you can use to convert differential rate equations to reactions are:

- **For a positive term** — The species described by the equation is placed on the right as a product, and the species in the term are placed on the left as reactants.
- **For a negative term** — The species described by the equation is placed on the left as a product, and the species in the term are also placed on the left as reactants.

  You need to determine the products using additional information, for example, a reaction diagram, a description of the model, or an understanding of a reaction. If a reaction is catalyzed by a kinase, then you can conclude that the product has one or more additional phosphate groups.

A simple first-order reaction has differential rate equation `dR/dt = +kr[P] - kf[R]`. The negative term implies that the reaction is `R -> ?` with an unknown product. The positive term identifies the product and completes the reaction, `R <-> P`.

Reactions R1 to R3 from Equation E1

The differential rate equation 1 is repeated here for comparison with the reactions. See "Differential Rate Equation 1, Cyclin (C)" on page 1-4.

$$\frac{dC}{dt} = vi - vdX\frac{C}{Kd + C} - kdC$$

The reaction and reaction rate equations from the differential rate equation E1 are given below:

```
r1      reaction: null -> C
   reaction rate: vi

r2      reaction: C -> null
   reaction rate: kd*C

r3      reaction: C -> null
   reaction rate: (vd*X*C)/(Kd + C)
```

Reactions R4 and R5 from Equation E2

The differential rate equation 2 and algebraic equation 2 are repeated here for comparison with the reactions. See "Differential Rate Equation 2, Kinase (M)" on page 1-4.

$$\frac{dM}{dt} = V1\frac{(1 - M)}{K1 + (1 - M)} - V2\frac{M}{K2 + M}$$

$$V1 = \frac{VM1[C]}{Kc + [C]}$$

The reaction and reaction rate equations from the differential rate equation E2 are given below:

```
r4      reaction: Mplus -> M
   reaction rate: V1*Mplus/(K1 + Mplus)
  algebraic rule: V1 = VM1*C/(Kc + C)

r5      reaction: M -> Mplus
```

```
reaction rate: V2*M/(K2 + M)
```

Reactions R6 and R7 from Equation E3

The differential rate equation for equation 3 and algebraic equation 3 is repeated here for comparison with the reactions.

$$\frac{dX}{dt} = V3\frac{(1-X)}{K3+(1-X)} - V4\frac{X}{K4+X}$$

```
V3 = VM3*[M]
```

The reaction and reaction rate equations from the differential rate equation E3 are given below:

```
r6      reaction: Xplus -> X
   reaction rate: V3*Xplus]/(K3 + Xplus)
  algebraic rule: V3 = VM3*M

r7      reaction: X -> Xplus
   reaction rate: V4*X/(K4 + X)
```

Calculating Initial Values for Reactions

After you converted the differential rate equations to the reactions and reaction rate equations, you can start to fill in initial values for the species (reactants and products) and parameters.

The initial values for parameters and amounts for species are listed with four different units in the same dimension:

- A — Original units in the Goldbeter 1991 paper.
- B — Units of concentration with time converted to second. When converting a to b, use `1 minute = 60 second` for parameters.

$$\frac{\text{X uM}}{\text{minute}} \text{ x } \frac{\text{1e-6 mole/liter}}{\text{1 uM}} \text{ x } \frac{\text{1 minute}}{\text{60 second}} = \frac{\text{Y mole}}{\text{liter*second}}$$

- C — Units of amount as moles. When converting concentration to moles, use a cell volume of `1e-12` liter and assume that volume does not change.

$$\frac{\text{Y mole}}{\text{liter*second}} \text{ x } \frac{\text{1e-12 liter}}{} = \frac{\text{Z mole}}{\text{second}}$$

- D — Units of amount as molecules. When converting amount as moles to molecules, use `6.022e23 molecules = 1 mole`.

$$\frac{\text{Z mole}}{\text{second}} \text{ x } \frac{\text{6.022e23 molecule}}{\text{1 mole}} = \frac{\text{N molecules}}{\text{second}}$$

With dimensional analysis on and unit conversion off, select all of the units for one letter. For example, select all of the As. If dimensional analysis and unit conversion are on, you can mix and match letters and get the same answer.

Reaction 1 Cyclin Synthesis

R1		Value	Units
reaction	null -> C	—-	—-
reaction rate	vi	—-	A. uM/minute
		—-	B. mole/(liter*second)
		—-	C. mole/second
		—-	D. molecule/second
parameters	vi	0.025	A. uM/minute
		4.167e-10	B. mole/(liter*second)
		4.167e-22	C. mole/second
		205	D. molecule/second
species	C	0.01	A. uM
		1e-8	B. mole/liter
		1.0e-20	C. mole
		6.022e+3	D. molecule

Reaction 2 Cyclin Undifferentiated Degradation

R2		Value	Units
reaction	`C -> null`	—-	—-
reaction rate	`kd*C`	—-	A. uM/minute
		—-	B. mole/(liter*second)
		—-	C. mole/second
		—-	D. molecule/second
parameters	`kd`	0.010	A. 1/minute
		1.6667e-4	B, C, D. 1/second
species	`C`	0.01	A. uM
		1e-8	B. mole/liter
		1.0e-20	C. mole
		6.022e+3	D. molecule

Reaction 3 Cyclin Protease Degradation

R3		Value	Units
reaction	`C -> null`	—-	—-
reaction rate	`(vd*X*C)/(Kd + C)`	—-	A. uM/minute
		—-	B. mole/(liter*second)
		—-	C. mole/second
		—-	D. molecule/second
parameter	`vd`	0.25	A. 1/minute
		0.0042	B, C, D. 1/second
parameter	`Kd`	0.02	A. uM
		2.0e-8	B. mole/liter
		2.0e-020	C. mole

R3		Value	Units
		12044	D. molecule
species	`C` (substrate)	0.01	A. uM
		1e-8	B. mole/liter
		1.0e-20	C. mole
		6.022e+3	D. molecule
species	`X` (enzyme)	0.01	A. uM
		1e-8	B. mole/liter
		1.0e-20	C. mole
		6.022e+3	D. molecule

Reaction 4 Cdc2 Kinase Activation

R4		Value	Units
reaction	`Mplus -> M`	—-	—-
reaction rate	`(V1*Mplus)/(K1 + Mplus)`	—-	A. uM/minute
		—-	B. mole/(liter*second)
		—-	C. mole/second
		—-	D. molecule/second
algebraic rule	`V1 = (VM1*C)/(Kc + C)`	—-	
parameter	V1 (variable by rule)	0.00	A. uM/minute
			B. mole/(liter*second)
			C. mole/second
			D. molecule/second
parameter	`VM1`	3.0	A. uM/minute
		5.0e-8	B. mole/(liter*second)

R4		Value	Units
		5.0000e-020	C. mole/second
		30110	D. molecule/second
parameter	`Kc`	0.5	A. uM
		5.0000e-7	B. mole/liter
		5.0e-19	C. mole
		3.011e+5	D. molecule
parameter	`K1`	0.005	A. uM
		5e-9	B. mole/liter
		5e-21	C. mole
		3.011e+3	D. molecule
species	`Mplus` (inactive substrate)	0.99	A. uM
		9.9e-7	B. mole/liter
		9.9e-19	C. mole
		5.962e+5	D. molecule
species	`M` (active product)	0.01	A. uM
		1e-8	B. mole/liter
		1.0e-20	C. mole
		6.022e+3	D. molecule
species	`C`	0.01	A. uM
		1e-8	B. mole/liter
		1.0e-20	C. mole
		6.022e+3	D. molecule

Reaction 5 Cdc2 Kinase Deactivation

R5		Value	Units
reaction	`M -> M_plus`	—-	—-
reaction rate	`(V2*M)/(K2 + M)`	—-	A. uM/minute
		—-	B. (mole/liter-second)
		—-	C. mole/second
		—-	D. molecule/second
parameter	`V2`	1.5	A. uM/minute
		2.5000e-008	B. mole/liter-second
		2.5000e-020	C. mole/second
		15055	D. molecule/second
parameter	`K2`	0.005	A. uM
		5.0000e-009	B. mole/liter
		5.0000e-021	C. mole
		3011	D. molecule
		1.0e-20	C. mole
species	`Mplus` (inactive)	0.99	A. uM
		9.9e-7	B. mole/liter
		9.9e-19	C. mole
		5.962e+5	D. molecule
species	`M` (active)	0.01	A. uM
		1e-8	B. mole/liter
		1.0e-20	C. mole
		6.022e+3	D. molecule

Reaction 6 Protease Activation

R6		Value	Units
reaction	`Xplus -> X`	—-	—-
reaction rate	`(V3*Xplus)/(K3 + Xplus)`	—-	A. uM/minute
		—-	B. mole/(liter*second)
		—-	C. mole/second
		—-	D. molecule/second
algebraic rule	`V3 = VM3*M`	—-	
parameter	`V3` (variable by rule)		A. uM/minute
			B. mole/liter-second
			C. mole/second
			D. molecule/second
parameter	`VM3`	1.0	A. 1/minute
		0.0167	B, C, D. 1/second
parameter	`K3`	0.005	A. uM
		5e-9	B. mole/liter
		5e-21	C. mole
		3.011e+3	D. molecule
species	`Xplus` (inactive substrate)	0.99	A. uM
		9.9e-7	B. mole/liter
		9.9e-19	C. mole
		5.962e+5	D. molecule
species	`X` (active product)	0.01	A. uM
		1e-8	B. mole/liter
		1.0e-20	C. mole

R6		Value	Units
		6.022e+3	D. molecule
species	`M` (enzyme)	0.01	A. uM
		1e-8	B. mole/liter
		1.0e-20	C. mole
		6.022e+3	D. molecule

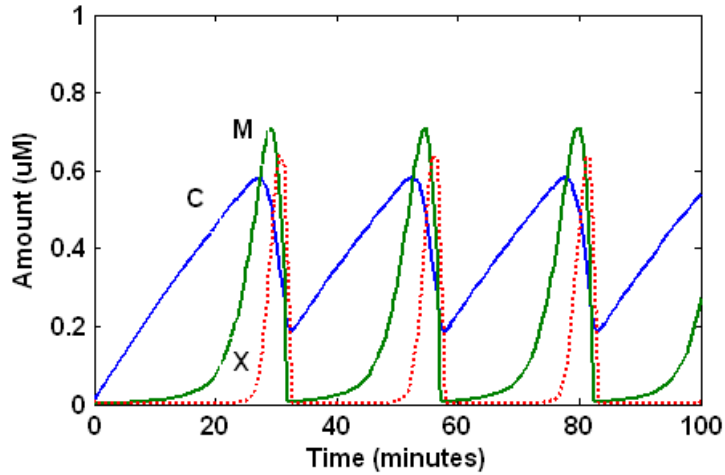
Reaction 7 Protease Deactivation

R7		Value	Units
reaction	`X -> X_plus`	—-	—-
reaction rate	`(V4*X)/(K4 + X)`	—-	A. uM/minute
		—-	B. mole/(liter*second)
		—-	C. mole/second
		—-	D. molecule/second
parameter	`V4`	0.5	A. uM/minute
		8.3333e-009	B. mole/(liter*second)
		8.3333e-021	C. mole/second
		5.0183e+003	D. molecule/second
parameter	`K4`	0.005	A. uM
		5e-9	B. mole/liter
		5e-21	C. mole
		3011	D. molecule
species	`Xplus` (inactive)	0.99	A. uM
		9.9e-7	B. mole/liter
		9.9e-19	C. mole
		5.962e+5	D. molecule
species	`X` (active)	0.01	A. uM

R7	Value	Units
	1e-8	B. mole/liter
	1.0e-20	C. mole
	6.022e+3	D. molecule

SimBiology Simulation with Reactions

This is a simulation of Goldbeter's minimal mitotic oscillator with rate and algebraic equations. Simulate with the ode15s solver and plot species `C`, `M`, and `X`. For a description of the model, see "SimBiology Model with Reactions" on page 1-10.



References

[1] Goldbeter A. (1991), "A minimal cascade model for the mitotic oscillator involving cyclin and cdc2 kinase," Proceedings of the National Academy of Sciences USA, 88:9107-9111.

[2] Goldbeter A., Koshland D. (1981), "An amplified sensitivity arising from covalent modification in biological systems," Proceedings of the National Academy of Sciences USA, 78:6840-6844.

[3] Goldbeter A., Koshland D. (1984), "Ultrasensitivity in biochemical systems controlled by covalent modification," The Journal of Biological Chemistry, 259:14441-14447.

[4] Goldbeter A., home page on the Web, `http://www.ulb.ac.be/sciences/utc/GOLDBETER/agoldbet.html`

[5] Murray A.W., Kirschner M.W. (1989), "Cyclin synthesis drives the early embryonic cell cycle," Nature, 339:275-280.

2

Model of the Yeast Heterotrimeric G Protein Cycle

Objectives

SimBiology software lets you build a model using a conceptual framework of biochemical reactions that describe a biological process. You can plot experimental data on top of your model's simulation results to investigate the validity of your model, make predictions based on the model, and test your hypotheses.

Using concepts and data from the published work of Yi and colleagues [Yi et al. 2003], this tutorial shows you how to:

1. Build a model using the SimBiology graphical user interface (GUI).
2. Apply an alternate value during simulation to create a variation of the model (for example, wild-type versus mutant).
3. Simulate and save the data from the two models.
4. Compare the two simulations.
5. Compare the simulation results with the experimental data.

Background on G Protein Cycles

In this section...

G Proteins

Cells rely on signal transduction systems to communicate with each other and to regulate cellular processes. G proteins are GTP-binding proteins that are involved in the regulation of many cellular processes. There are two known classes of G proteins: the monomeric G proteins (one GTPase), and the heterotrimeric G proteins (three different monomers). The G proteins usually facilitate a step requiring energy. This energy is supplied by the hydrolysis of GTP by a GTPase activating protein (GAP). The exchange of GDP for GTP is catalyzed by a guanine nucleotide releasing protein (GNRP) [Alberts et al. 1994].

$$Gprotein + GTP \underset{GNRP}{\overset{GAP}{\rightleftharpoons}} Gprotein + GDP$$

G protein-coupled receptors (GPCRs) are the targets of many pharmaceutical agents. Some estimates suggest that 40 to 50% of currently marketed drugs target GPCRs and that 40% of current drug discovery focus is on GPCR targets. Some examples include those for reducing stomach acid (ranitidine which targets histamine H2 receptor), migraine (sumatriptan, which targets a serotonin receptor subtype), schizophrenia (olanzapine, which targets serotonin and dopamine receptors), allergies (desloratadine, which targets histamine receptors). One approach in pharmaceutical research is to model signaling pathways to analyze and predict both downstream effects and effects in related pathways. This tutorial examines model building and analysis of the G protein cycle in the yeast pheromone response pathway using the SimBiology desktop.

G Proteins and Pheromone Response

In the yeast *Saccharomyces cerevisiae*, G protein signaling in pheromone response is a well characterized signal transduction pathway. The pheromone secreted by *alpha* cells activates the G protein-coupled α-factor receptor (Ste2p) in *a* cells which results in a variety of cell responses including cell-cycle arrest and synthesis of new proteins. The authors of the study performed a quantitative analysis of this cycle, compared the regulation of G protein activation in wild-type yeast haploid *a* cells with cells containing mutations that confer supersensitivity to α-factor. They analyzed the data in the context of cell-cycle arrest and pheromone-induced transcriptional activation and developed a mathematical model of the G protein cycle that they used to estimate rates of activation and deactivation of active G protein in the cell.

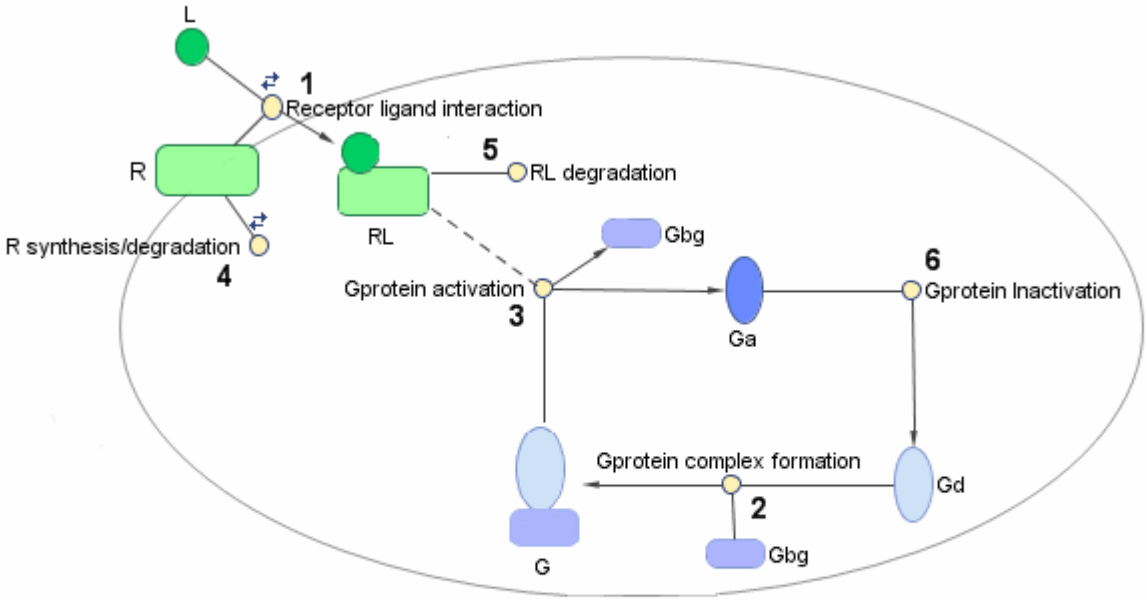
Modeling a G Protein Cycle

In this section...

Reactions Overview

Systems biologists represent biological pathways and processes as reactions with reaction rates, and treat the components of these pathways as individual species.

The G protein cycle in the yeast pheromone-response pathway can be condensed into a set of biochemical reactions. These reactions are complex formation, transformation, or disassociation reactions that Yi and colleagues [Yi et al. 2003] use to simplify and describe the system. In this example, α-factor, α-factor receptor, and the G protein subunits are all treated as species participating in reactions. The system can be graphically represented as follows.



Graphical Representation of the G protein cycle in yeast pheromone response. The numbers represent reaction numbers referenced in the text. L = Ligand (alpha factor), R = alpha-factor receptor, Gbg = free levels of G-beta:G-gamma complex, Ga = active G-alpha-GTP, Gd = inactive G-alpha-GDP, G = inactive Gbg:Gd complex.

The following table shows you the reactions used to model the G protein cycle and the corresponding rate constants (rate parameters) for each reaction. For reversible reactions, the forward rate parameter is listed first.

No.	Name	Reaction	Rate Parameters
1	Receptor-ligand interaction	`L + R <-> RL`	`kRLm, kRL`
2	Heterotrimeric G protein formation	`Gd + Gbg -> G`	`kG1`
3	G protein activation	`RL + G -> Ga + Gbg + RL`	`kGa`

No.	Name	Reaction	Rate Parameters
4	Receptor synthesis and degradation	`R <-> null`	`kRdo`, `kRs`
5	Receptor-ligand degradation	`RL -> null`	`kRD1`
6	G protein inactivation	`Ga -> Gd`	`kGd`

Note that in reaction 3 (G protein activation), `RL` appears on both sides of the reaction. This is because `RL` is treated as a modifier or catalyst, and the model assumes that there is no synthesis or consumption of `RL` in this reaction.

The authors use a set of ordinary differential equations (ODEs) to describe the system. In the software, you can represent the biological pathway as a system of biochemical reactions and the software creates the ODEs for you. Alternatively, if you have a set of ODEs that describe your system you can enter these as rate rules. For an example of modeling using rate rules, see "SimBiology Model with Rate Rules" on page 1-6.

Assumptions, Experimental Data, and Units in the G Protein Model

The authors have obtained experimental data either through their own measurements or through published literature. As with any other model, the G protein cycle model simplifies the biological process while also trying to reconcile the experimental data. Consider these points:

- Reaction 2 — Binding and formation of the heterotrimeric G protein complex is treated as a single-step reaction.
- Reaction 3 — Activation of G protein is modeled as a single-step. Guanine nucleotide exchange factors (GEFs) are not modeled.
- Reactions 3 and 6 — The parameters for the rate of G protein activation and deactivation (`kGa` and `kGd`) have been estimated based on the dose response curves in the reference paper. The SimBiology model being built in this tutorial directly uses those values.

- Reactions 4 and 5 — Receptor synthesis and degradation are handled purely as two simple reaction steps.
- Reaction 6 — Deactivation of G protein by the regulator of G protein signaling (RGS) protein Sst2p is modeled as a single step. Sst2p is not modeled.

  The reaction is modeled with an estimated reaction rate of `0.11` s^{-1}) in the Sst2p containing wild-type strain. The uncatalyzed reaction rate is estimated to be `0.004` s^{-1} in a strain with a deletion of SST2 (*sst2Δ*, mutant strain).
- Free GDP, GTP, and Pi are not included in the model.

This tutorial shows you how to plot the experimental data over the simulation plot of the active G protein fraction. You can estimate the values of the experimental data of interest for this example from the coordinates of the plots found in Figure 5 of the reference paper [Yi et al. 2003]. The following values were obtained by comparing the coordinates of the standards with those of the unknowns in the figure.

Time	Fraction of Active Gα (Experimental)
0	0.00
10	0.35
30	0.40
60	0.36
110	0.39
210	0.33
300	0.24
450	0.17
600	0.20

Note The SimBiology **Dimensional Analysis** feature is not used in this tutorial. For this tutorial, the values of all species are converted to have the unit `molecule`, and all rate parameters are converted to have either the unit `1/second` or the units `1/(molecule*second)`, depending on whether the reaction is first or second order. You should leave the **InitialAmountUnits** box for species and the **ValueUnits** box for rate parameters empty for the models in this tutorial.

Building the G Protein Cycle Model

In this section...

Tutorial Goals

This section shows you how to build the example yeast heterotrimeric G protein models using the SimBiology desktop GUI (graphical user interface). For an overview of the SimBiology desktop, click here. The SimBiology desktop is also called "the desktop" in the SimBiology documentation.

This section assumes that you are starting with an untitled `Project` and a default `Untitled Model Session` in the SimBiology desktop.

If you are running the `gprotein.sbproj` file that contains the models in this tutorial, see "Simulating the G Protein Cycle Model" on page 2-27 for details about the simulation and analysis of these models.

This example uses the yeast G protein cycle [Yi et al. 2003] to illustrate model building and analysis. The goals of this tutorial are the following:

1. Build a model of the wild-type strain (TMY101) that has the SST2 gene. This strain shows a catalyzed rate of deactivation of Gα. Gα is represented as Ga in the model.
2. Store alternate values in a **Variant** to create a different model for the mutant strain (*sst2Δ* , TMY111) that shows an uncatalyzed rate of G-Protein inactivation.

3 Simulate and save the data from the two models.

4 Compare the active G protein fractions in the two simulations.

5 Compare the simulation results for active G protein fractions with experimental data.

For additional help in each procedure, select **Help > SimBiology Desktop Help**.

Opening the SimBiology Desktop

The procedures in this tutorial are performed in the SimBiology desktop. The desktop provides access to command-line functionality through a graphical user interface. You can open the desktop from the MATLAB Command Window.

1 At the MATLAB command line, type:

```
sbiodesktop
```

The SimBiology desktop opens. Use the **Project Explorer** in the left pane to navigate.

2 Select **File > New Project** to open the New Project dialog box.

3 From the **Project Type** list, select **Simulation (recommended)** and click **OK**.

When you select an option, the **Project Description** pane shows you the items included in the project.

A new project, with an untitled model session, opens in the desktop.

4 In the **Project Explorer**, click **SimBiology Model** to open the **SimBiology Model** pane.

5 Click the **Settings** tab.

6 In the **Model Name** box, type the name for your model and press **Enter**.

```
Yeast_G_Protein_wt
```

Saving Your Work as a SimBiology Project File

Project (`.sbproj`) is the file format used to save one or more model sessions. Projects let you save custom settings, notes, and data associated with your models.

Save your work as a project now so that you can access this file later.

1. Select **File > Save Project As** to open the Save SimBiology Project dialog box.
2. Browse to the folder in which you want to save your projects, enter a name for the project file, and then click **Save**.

```
yeast_g_protein_cycle.sbproj
```

Adding a Reaction to the SimBiology Model

The next steps show how to add a reaction and determine the reaction rate equation for your model.

This example shows the first reaction.

Name	Reaction	Rate Parameters
Receptor-ligand interaction	L + R <-> RL	kRLm, kRL

1. In the **Project Explorer**, expand **SimBiology Model** and click **Reactions** to open the **Reactions** pane.
2. Enter the reaction in the **Enter Reaction** box, and click **Add**.

```
L + R <-> RL
```

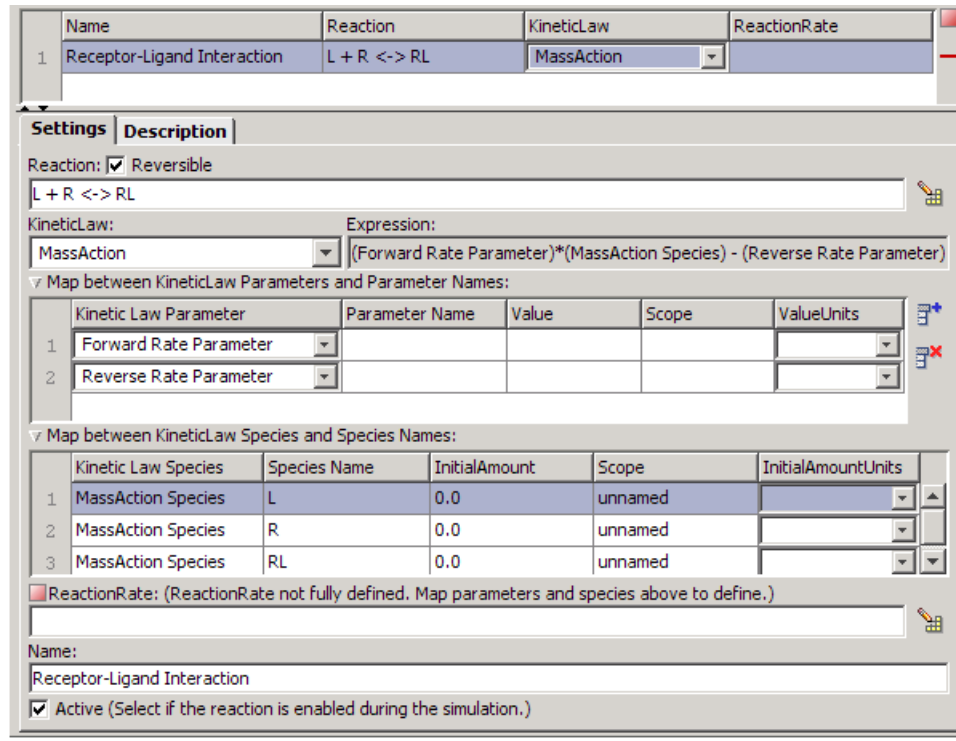
   A red indicator appears to the right of the table. Move the pointer over the indicator for the reaction, to get more information about the reason for the indicator. In this case, the indicator shows that the reaction rate is as yet undefined. The next section shows how to define the reaction rate.

3. Double-click the **Name** box, and type the name for your reaction. For example:

```
Receptor-Ligand Interaction
```

4 In the reaction table, from the **KineticLaw** list, select `MassAction`.

Your screen should now resemble the following figure.



Notice the following in the **Settings** tab:

- Because this reaction is reversible, the **Reversible** check box is selected by default when you enter the reaction.
- All the reactions in this example model are included in the simulation and, therefore, have the **Active** check box selected. By default, the **Active** check box is selected.

Tip Use spaces between the species and the characters in the reaction. If you have a reaction with different stoichiometry, for example, `2 A + B <-> 3 AB`, you must have a space between the stoichiometric coefficient and the species name for the reaction rate to be accurately determined. Otherwise, the coefficients are considered as part of the species name.

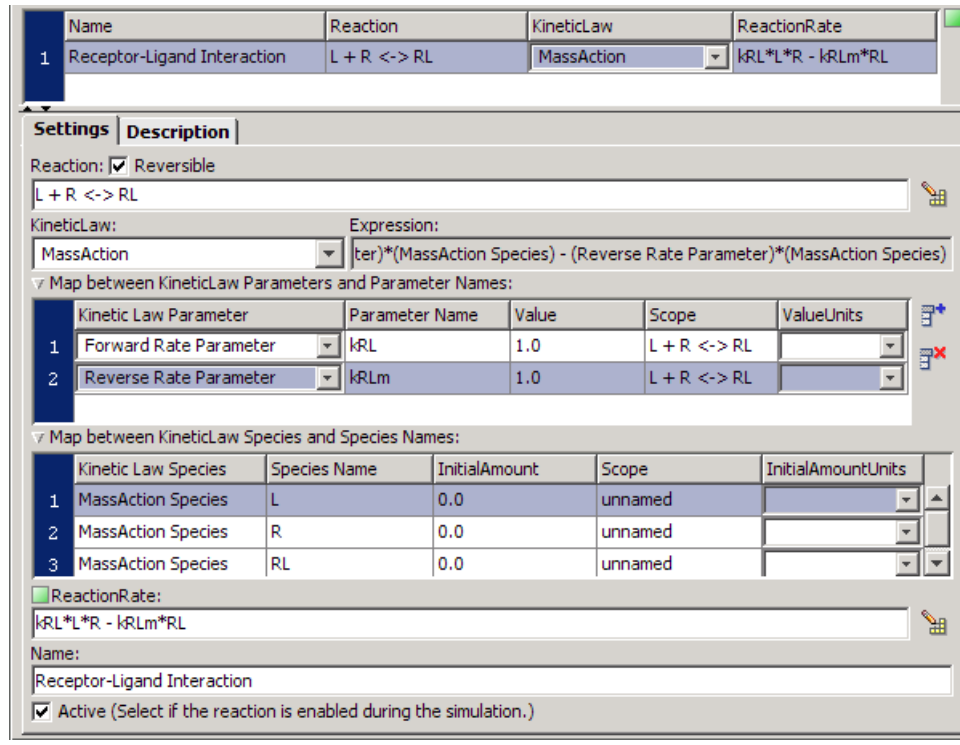
Determining the Reaction Rate Equation

The desktop populates the reaction rate column after you specify the kinetic law and the rate parameters of the reaction.

To assign and configure the kinetic law and the rate parameters, do the following in the **Reactions** pane:

1. In the **Map between KineticLaw Parameters and Parameter Names** section, locate the **Forward Rate Parameter** row, double-click the **Parameter Name** cell, type `kRL` and press **Enter**.
2. In the **Reverse Rate Parameter** row, double-click the **Parameter Name** cell, type `kRLm` and press **Enter**.

.



- The desktop updates the **ReactionRate** column to show `kRL*L*R - kRLm*RL`, and the indicator is now green showing that this reaction is completely defined.
- For this example, **Scope** of all the parameters is at the kinetic law level. The desktop displays the reaction in the **Scope** box.
- The desktop automatically selects the species when you select `MassAction` kinetic law. For other kinetic laws, you should enter the species to be included in the rate equation.

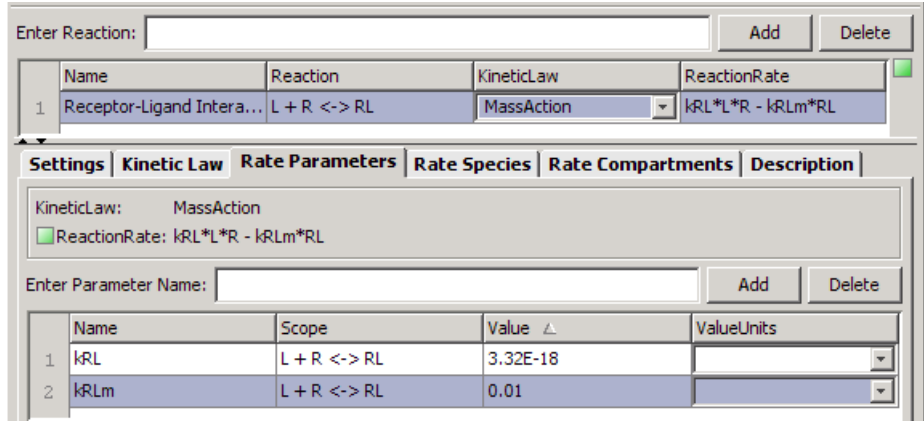
Using Undo and Redo

Notice the status bar at the bottom of the desktop. This shows that the most recent action on the desktop.

To undo or redo an action or a sequence of actions, right-click the listing in the status bar (for example, **Set ParameterVariableNames**) and select the action to perform. There are some actions that the desktop cannot undo (for example, adding or deleting blocks in the **Diagram**). These actions clear the sequence stack and you can no longer access the previous sequence of undo and redo actions.

3 In the **Value** cell, enter the following parameter values:

`kRL` = `3.32E-18` and `kRLm` = `0.01`



Setting the Compartment Name

All models created in the desktop contain a compartment by default. The process of adding a reaction automatically adds the reaction species to a compartment in the model. If there are multiple compartments in the model, you must specify the reactants and products using qualified names (*`compartmentName.speciesName`*). For example, `nucleus.DNA` denotes the species `DNA` in the compartment `nucleus`.

This example contains only one compartment. To rename the compartment:

1 In the **Project Explorer**, click **Compartments**.

2 In the **Name** cell, double-click and type a name for the compartment, and then press **Enter**.

```
yeast_cell
```

The compartment table updates with the new name.

Use the default values for the **Owner**, **Capacity**, and **CapacityUnits** cells.

Why Use Default Values

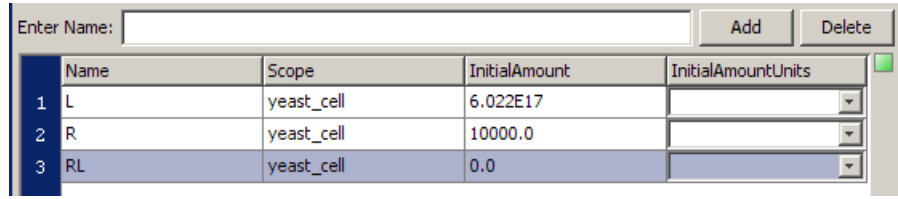
- **Owner** lets you define whether the compartment is within another compartment. This model has only one compartment within which you define all the species. No **Owner** means that the compartment is a top-level compartment.
- **Capacity** lets you enter the compartment size, such as volume, length, or area. In this model, species are defined in amounts, the reaction rate dimensions are amount/time, and the model assumes that compartment volume is constant. Thus, all the values and the units are internally consistent with each other and the compartment volume can use the default value of `1.0`.
- **CapacityUnits** lets you specify the units for **Capacity**, which are not needed for this example.

Setting Initial Amounts of Species

You can set the initial amounts of all the model species in the **Species** pane.

Name	InitialAmount
L	6.022E17
R	10000.0
RL	0.0

1. In the **Project Explorer**, expand **Compartments** and click **yeast_cell Species**.
2. In the **Species** pane, double-click in each **InitialAmount** cell and enter the preceding values.



Notice that the **Scope** column lists the name of the compartment containing the species.

You now have a complete reaction with all components added and defined.

Completing the SimBiology Model

In this section...

Reactions

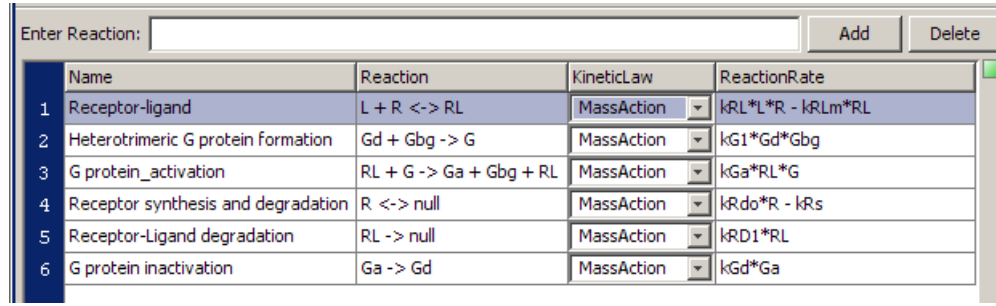
The previous sections showed you how to the enter the first reaction for the yeast G protein cycle model and configure the reaction rate equation. Repeat the procedures to add the rest of the reactions, parameters, and species values as described in the previous sections, and create a rule to specify the ratio of active G protein that corresponds to the ratio determined experimentally in the referenced study [Yi et al. 2003].

Add reactions 2 to 6 listed in this table, set the kinetic law for each reaction to `MassAction`, create parameters, and configure the reaction rate using the procedure for "Determining the Reaction Rate Equation" on page 2-14.

No.	Name	Reaction	Rate Parameters
1	Receptor-ligand interaction	`L + R <-> RL`	`kRL`, `kRLm`
2	Heterotrimeric G protein formation	`Gd + Gbg -> G`	`kG1`
3	G protein activation	`RL + G -> Ga + Gbg + RL`	`kGa`
4	Receptor synthesis and degradation	`R <-> null`	`kRdo`, `kRs`

No.	Name	Reaction	Rate Parameters
5	Receptor-ligand degradation	`RL -> null`	`kRD1`
6	G protein inactivation	`Ga -> Gd`	`kGd`

Your reaction table should resemble the following figure.



The next section describes other ways you can build reactions in the desktop. If you want to continue building the model, skip the next section and go to "Parameters" on page 2-22.

Alternative Ways to Build Reactions in the Desktop

This optional section shows you some alternative methods to build a reaction in the **Reactions** pane. Use the examples above to try out these methods. If you have finished entering the reactions for the model, proceed to "Parameters" on page 2-22.

The **Reactions** pane has several dialog boxes that are convenient for building reactions.

Building a Reaction

You can graphically build a reaction using the Reaction Builder.

1. If you are not already in the **Reactions** pane, in the **Project Explorer** click **Reactions**.

2 Click **Build** to open the Reaction Builder dialog box.

3 Select a species from the **Available Species** list and click the **Reactant** or **Product** button.

4 To edit stoichiometric relationships, click in the **Stoich** column and type.

5 Select the **Reversible** check box if the reaction is reversible.

6 Click **Add** to continue editing within the Reaction Builder. Click **OK** to finish and return to the **Reactions** pane.

Creating a Binding Reaction

Use the **Bind** button to create a bound product from two reactant species.

1 If you are not already in the **Reactions** pane, in the **Project Explorer** click **Reactions**.

2 In the **Enter Reaction** box, enter the reactant species.

```
lactose + lactase
```

3 Click the 

button. The software writes the product as a compound name, with a colon between reactant names to indicate binding.

```
lactose:lactase
```

Creating an Unbinding Reaction

Use the **Unbind** button to create two product species from a bound reactant.

1 If you are not already in the **Reactions** pane, in the **Project Explorer** click **Reactions**.

2 In the **Enter Reaction** box, enter the reactant species.

```
lactose:lactase
```

3 Click 

. The software writes the product names with a + between them to indicate unbinding.

```
lactose + lactase
```

Parameters

In the **Reactions** pane, after you create parameters in the **Settings** tab, you can set the value of the parameter in the parameters table.

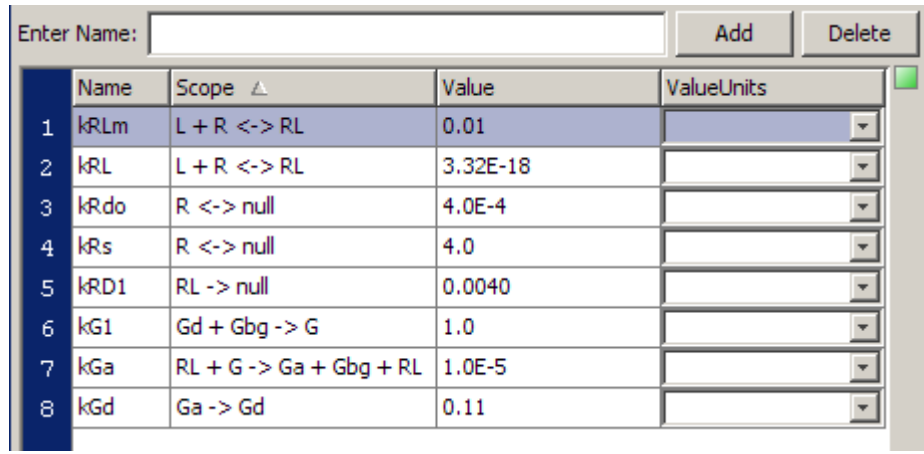
Alternatively, you can set all the parameters in the **Parameters** pane. In the **Project Explorer**, click **Parameters** to open the **Parameters** pane.

Use the following values for rate parameters.

Parameter Value Table

Name	Value
kRLm	0.01
kRL	3.32E-18
kRdo	4.0E-4
kRs	4.0
kRD1	0.0040
kG1	1.0
kGa	1.0E-5
kGd	0.11

To be consistent with units for `kRL`, `RL` and `L`, the value for `kRL` is converted from the published value, `2.0E6M`$^{-1}$`s`$^{-1}$, to `3.32E-18` with units `1/(molecule*second)` (assuming a volume of unity).



	Name	Scope	Value	ValueUnits
1	kRLm	L + R <-> RL	0.01	
2	kRL	L + R <-> RL	3.32E-18	
3	kRdo	R <-> null	4.0E-4	
4	kRs	R <-> null	4.0	
5	kRD1	RL -> null	0.0040	
6	kG1	Gd + Gbg -> G	1.0	
7	kGa	RL + G -> Ga + Gbg + RL	1.0E-5	
8	kGd	Ga -> Gd	0.11	

Species

Set the species amounts in the **Species** pane. In the **Project Explorer**, click **yeast_cell Species** to access this pane.

- Note that the process of adding reactions resulted in species automatically being added to the species list, with default amounts set to `0.0`. Double-click each **InitialAmount** cell to change the values to those given in the table.
- The amount of `L` (α-factor) used in the experiments is `1 M`. This value when converted to `molecule` (assuming a volume of unity) is `6.022E17`. This is now internally consistent with the units for the species, `RL`, and the parameter, `kRL`.

Species Initial Amounts

Name	InitialAmount (Molecule)
L	6.022E17
R	10000.0
G	7000.0
Gd	3000.0
Gbg	3000.0

Species Initial Amounts (Continued)

Name	InitialAmount (Molecule)
Ga	0.0
RL	0.0

To replicate the published results, the model needs the definition of the ratio of active G protein. Call this ratio `Ga_frac`. `Ga_frac` is `Ga/Gt`, where `Ga` is active G protein (Gα-GTP) and `Gt` is the total amount of G protein in a cell. This relationship is defined using a rule, and the procedure to create this rule is described in "Creating a Rule for the G Protein Model" on page 2-25.

Define two additional species, called `Ga_frac` and `Gt`.

Additional Species

Name	InitialAmount (Molecule)
Ga_frac	0.0
Gt	10000.0

To add a new species:

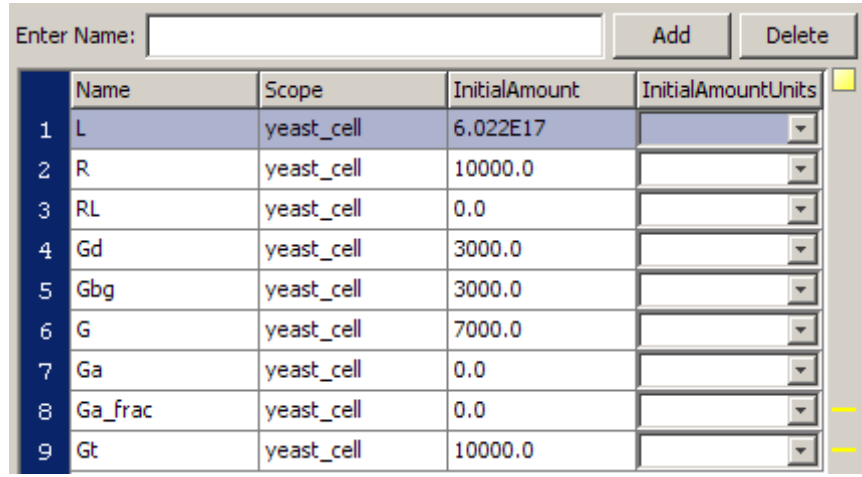
1. In the **Project Explorer**, click **yeast_cell Species** to open the **Species** pane.
2. In the **Enter name** box, enter the name of a new species, and then click **Add** or press **Enter**.

   ```
   Ga_frac
   ```

   The species table updates with the new entry and its row is selected. Note that the species is now available in the **Settings** tab.
3. In the **Initial Amount** cell, enter a value for the amount of the species. For `Ga_frac`, leave this at the default value (`0.0`).
4. Repeat steps 1 to 3 for `Gt`. **InitialAmount** is `10000.0`.

5 In the **Settings** tab, select the **ConstantAmount** check box only for `Gt`, because the amount of `Gt` does not vary during the simulation.

Your species table should resemble this.



	Name	Scope	InitialAmount	InitialAmountUnits
1	L	yeast_cell	6.022E17	
2	R	yeast_cell	10000.0	
3	RL	yeast_cell	0.0	
4	Gd	yeast_cell	3000.0	
5	Gbg	yeast_cell	3000.0	
6	G	yeast_cell	7000.0	
7	Ga	yeast_cell	0.0	
8	Ga_frac	yeast_cell	0.0	
9	Gt	yeast_cell	10000.0	

Notice the yellow indicator; move your pointer over the indicator to see the warning message. The message shows that the species is never used, because `Ga_frac` and `Gt` are not yet defined in a rule. After you define the rule as described in the next section, the indicator shows one green square.

For additional help in each procedure, select **Help > SimBiology Desktop Help** for context-sensitive help.

Creating a Rule for the G Protein Model

A SimBiology rule is a mathematical expression that modifies a species amount, compartment volume, or a parameter value. Use an algebraic rule to define the value of the species `Ga_frac` in the model.

1 In the **Project Explorer**, click **Rules** to open the **Rules** pane.

2 In the **Enter Rule** box, type the expression, and then click **Add** or press **Enter**.

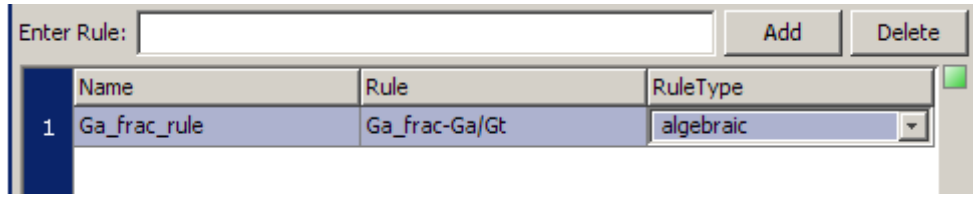
```
Ga_frac-Ga/Gt
```

The rule is entered as an expression.

The **RuleType** for this rule is the default (`algebraic`).

3 (Optional) Give your rule a name. Double-click and type the following in the **Name** box:

```
Ga_frac_rule
```



This completes the section on building the G protein cycle model for the wild-type strain.

Verifying the Model

While you are building your model in the SimBiology desktop, you can click 

at any time to generate a list of any errors and warnings in the model. The errors and warnings appear in the **Errors and Warnings** pane. For more information about verification see "Verifying that the Model Has No Warnings or Errors" in the SimBiology User's Guide.

Simulating the G Protein Cycle Model

In this section...

Setting Conditions Before Simulating the G Protein Model

In previous sections, this tutorial described building a model for a G protein cycle. This model uses the G protein cycle in the yeast pheromone response pathway. This section describes conditions for simulation and the simulation results for this model.

Consider the following points about simulating this model:

- Yi. et al. show data up to `600s` for the active G protein time course. To replicate these results, change the simulation settings from the default `10 second` to `600 second`. This change remains active for this model unless you change it back to the default.
- The ligand species `'L'` has values that are magnitudes higher than those of many of the other species. Don't log data for `'L'` so that while plotting you can enable instant visualization of the other species through proper scaling of plots. To do this, define **Data Logging** to stop logging data for `'L'`.

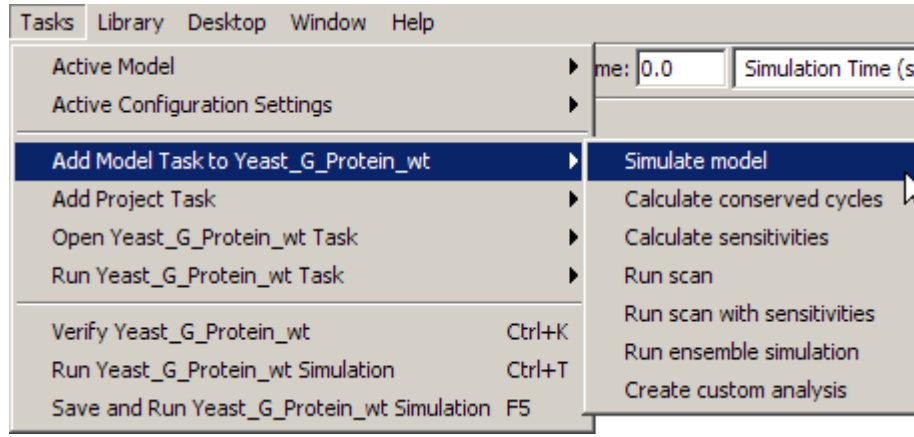
The first procedure (below) describes how to change the simulation stop time. The second procedure is about recording a subset of data ("Specifying Which Data Is Recorded" on page 2-29).

Adding a Simulation Task

When you created a new project as shown in "Opening the SimBiology Desktop" on page 2-11, the option chosen added a **Simulation** task to the project. You can skip to "Changing the Simulation Stop Time" on page 2-28.

If you have a project without a **Simulation** task, do the following to add a **Simulation** task:

- In the **Task** menu, select **Add Model Task to Yeast_G_Protein_wt > Simulate model**.



The SimBiology desktop adds the task to the **Project Explorer**, opens the task pane, and highlights the new task.

Changing the Simulation Stop Time

Change the stop time to replicate the simulation used in the reference paper [Yi et al. 2003], and to facilitate comparison with the experimental results presented in the study.

You will modify the `default` settings for this example. If you do not want to modify the default in your models, you can add another configuration set that you can then modify.

1. In the **Project Explorer**, expand **Model Variable Settings** and click **Configuration Settings**.

2. In the **Settings** tab, enter the stop time in the **Stop** box and press **Enter**.

```
600.0
```

Leave the following properties as the default (you may have to scroll to see some of the properties mentioned below):

- **SolverType**(`ode15s(stiff/NDF)`)

- **AbsoluteTolerance** (`1.0E-6`)
- **RelativeTolerance** (`0.0010`)
- **MaxStep** (`[]`)
- **DimensionalAnalysis** (check box selected)
- **UnitConversion** (check box clear)
- **DefaultSpeciesDimension** (`concentration`). This value is important if you are accounting for volume in your model. This example ignores volume and thus the value assigned to this property is not relevant.

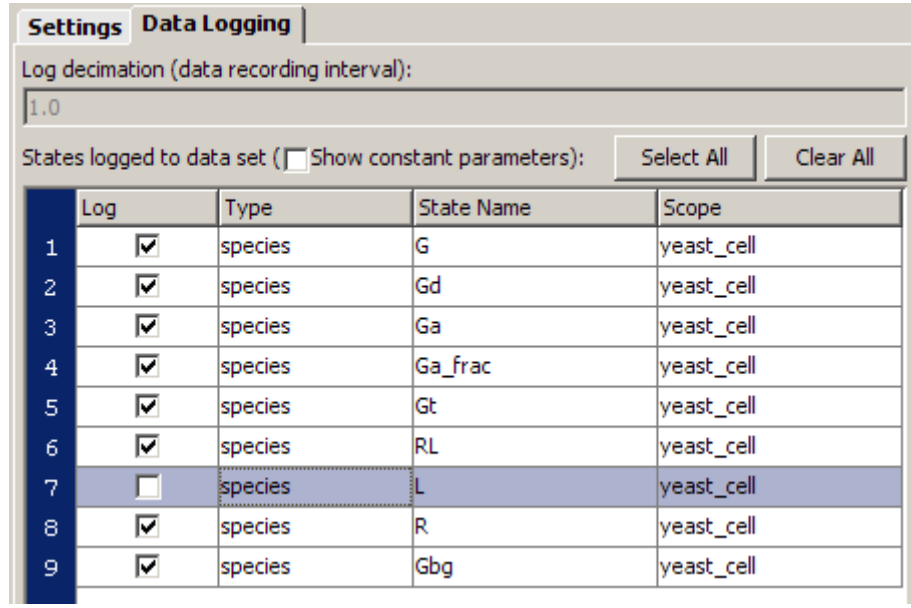
You can also change the solver type to use and the stop time for the simulation in the SimBiology toolbar. The desktop changes the setting of the `Active` configuration set when you change settings in the **Toolbar**.

Specifying Which Data Is Recorded

You can specify the species names for which SimBiology should *log*, that is, record the simulation data.

As mentioned in the previous section, you will modify the `default` settings for this example. If you do not want to modify the default in your models, you can add another configuration set that you can then modify.

1. In the **Configuration Settings** pane, click the **Data Logging** tab.
2. Select the check boxes for the species to log. Because, all the species check boxes are selected by default, clear the check box for `'L'`.



	Log	Type	State Name	Scope
1	☑	species	G	yeast_cell
2	☑	species	Gd	yeast_cell
3	☑	species	Ga	yeast_cell
4	☑	species	Ga_frac	yeast_cell
5	☑	species	Gt	yeast_cell
6	☑	species	RL	yeast_cell
7	☐	species	L	yeast_cell
8	☑	species	R	yeast_cell
9	☑	species	Gbg	yeast_cell

Tips for Use and Points to Consider

- You can save all these simulation settings as one custom simulation setting. See the context-sensitive help for **Simulation Settings**. To access context-sensitive help, select **Help > SimBiology Desktop Help**.
- The default `ode15s(stiff/NDF)` is adequate for modeling of many biological pathways. You might, however, need a different solver for some models. For more information on choosing solver types, see "Selecting a Solver" in the SimbBiology User's Guide documentation.
- You can choose from three stochastic solvers: `stochastic (SSA)`, `implicit tau`, and `explicit tau`. Try one of the stochastic solvers with this model and see how it compares with `ode15s`. For information, see "Stochastic Solvers" in the SimBiology User's Guide documentation. You can also see how the stochastic solvers compare with each other.
- For a counter that tracks the simulation, look in the lower-right corner of the SimBiology desktop.

- Click the following links to learn more about absolute and relative tolerance. These are links to SimBiology reference pages with definitions for `AbsoluteTolerance` and `RelativeTolerance`.

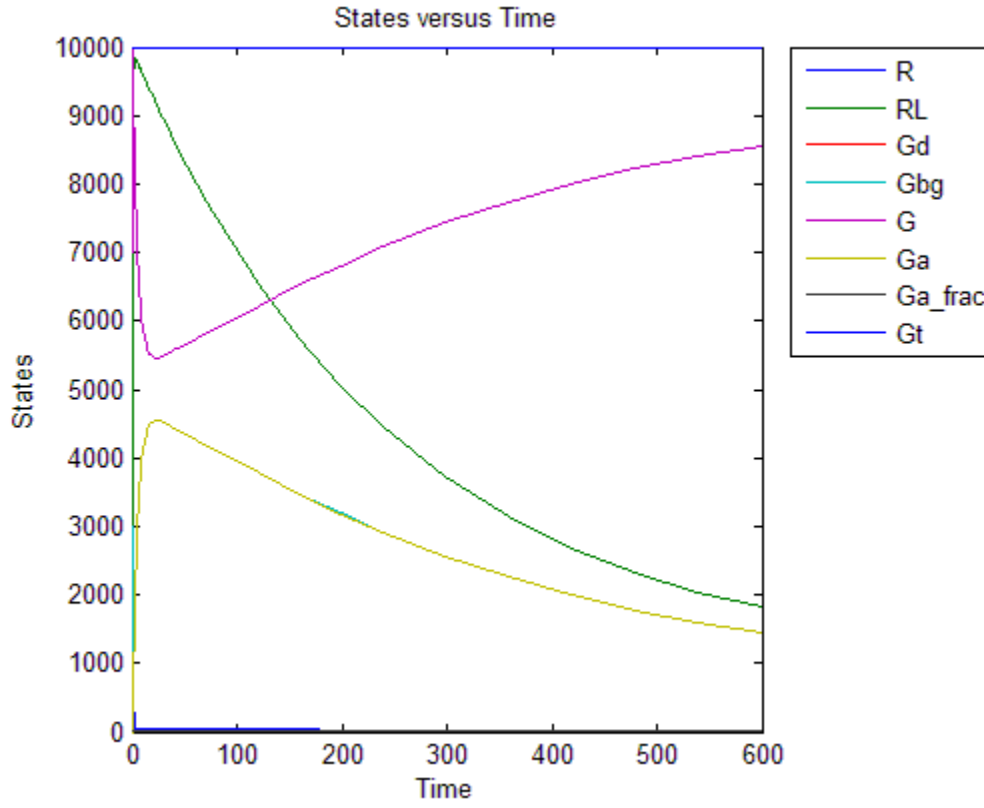
Simulation Results for the Wild-Type Strain Model

Simulate the model you have built and see your results. To simulate the model:

1. In the **Project Explorer**, click the **Simulation** task to open the **Simulation** pane.

2. Click  (**Run**).

   Your plot should resemble the following figure:



Saving Simulation Data

You can optionally save the data from the most recent simulation run. Unless you save the data for each simulation run, it is overwritten by the data for the next run.

1. In the **Project Explorer**, under **Simulation**, click **Data** for the wild-type model.
2. In the **Data** pane, click **Save** to open the Save Data dialog box.

3 Specify a name for your data, and then click **Save**.

```
wt_model_run1
```

The desktop adds the saved data under the **Simulation** task in the **Project Explorer**.

Creating the Mutant Strain Using a Variant

In this section...

Modeling the Mutant Strain

The deletion in SST2 results in uncatalyzed G protein deactivation (Reaction 6; `Ga -> Gd`). From a modeling perspective, this means a change in the rate of the reaction. This section shows you how represent the value for the mutant strain in a **Variant** and simulate the model using the variant value.

Note An additional simplifying assumption of this model is that there are no changes in the initial amounts of species or the rate of any other reaction.

Applying Alternate Values Using Variants

1. In the **Project Explorer**, expand **Model Variable Settings**, and click **Variants** to open the **Variants** pane.
2. In the **Enter Name** box, type a name for the variant, and then click **Add** or press **Enter**. For example:

   ```
   mut_value
   ```

3. Add content to the variant:

   a. In the **Settings** tab, click  (Add table row). The table updates with a row containing information about a model component.

   b. From the **Type** list, select `parameter`. The **Property** list updates to show the property available for changing.

   c. In the **Name** cell, type the name of the component.

```
Gprotein Inactivation.kGd
```

The parameter `kGd` is at the kinetic law level, and not the model level. Thus, you must specify the parameter in the format *`ReactionName.ParameterName`*.

d In the **Value** cell, type a value to apply using the variant.

```
0.004
```

See Also

"Storing and Applying Alternate Model Values Using Variants" in the SimBiologyUser's Guide.

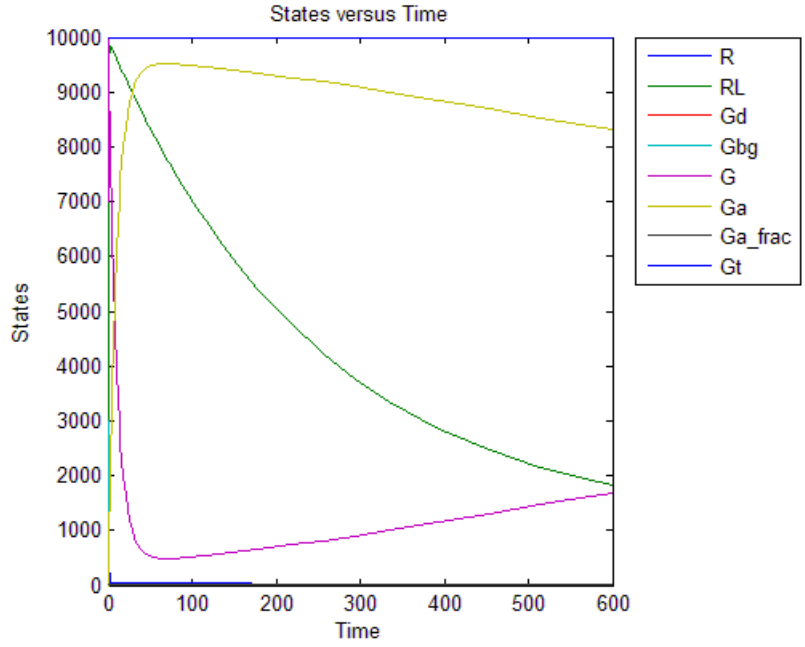
Simulation Results for the Model of the Mutant Strain

To simulate the model of the mutant strain, apply the variant and simulate as follows:

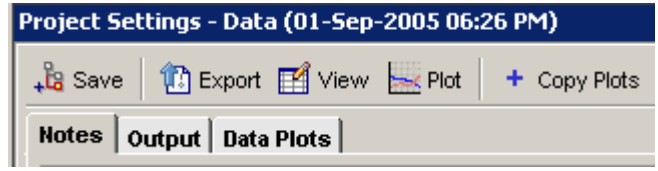
1 If the **Simulation** pane is not already open, in the **Project Explorer** select **Simulation** to open the pane.

2 In the **Variants** table, select the **Use in Task** check box for `mut_value`.

3 Click  (**Run**). Your plot should resemble the following figure.



4 In the **Data** pane for the latest simulation, click **Save** to open the Save Data dialog box.



5 Specify a name for your data, and then click **Save**.

```
mut_model_run1
```

The desktop adds the saved data under the **Simulation** task in the **Project Explorer**.

The simulation results for the wild-type strain are described in “Simulation Results for the Wild-Type Strain Model” on page 2-31.

Creating a Custom Plot-Type to View Simulation Results

In this section...

Creating a Custom Plot

To keep data plots from each model simulation distinct and to facilitate comparison, you can use customized plots. This example shows how to create and save a custom plot in the **Plot Types** library for use in plotting the simulation data with a different line style (dashed lines).

1. Select **Library > Show Library Explorer**. The **Library Explorer** opens.
2. In the **Library Explorer**, click **Plots Types**. The **Plot Types** pane opens.
3. In the **Enter Name** box, type a name for the custom plot and press **Enter**.

   ```
   Time plot line style
   ```

   The desktop adds the new plot to the plot types table.
4. In the **Enter Plot Type code below or copy Plot Type code from** section, replace the code in the editor window with the following:

```
function Time(tobj, y, lstyle)
%TIME Plots states versus time.
%
%    Plot the results of the simulation for the species with the specified
%    names versus time. Data from each run is plotted into one axes. Use the
%    Time Subplot type to plot data from each run into its own subplot. The
%    string '<all>' can be used to indicate that all species should be
%    plotted. LStyle specifies the line style that can be used.
%
%    See also GETDATA, SELECTBYNAME.

if (length(tobj) > 1)
```

```
    sbioplot(tobj, @timeplotdata, [], y, lstyle);
else
    timeplotdata(tobj, [], y, lstyle);
end

%-----------------------------------------------------
function [handles, names] = timeplotdata(tobj, x, y, lstyle)

colors    = get(gca, 'ColorOrder');
numColors = length(colors);
handles = [];
for i=1:length(tobj)
    % Get the data from the next run.
    nexttobj = tobj(i);

    % Get the simulation data associated with the species
    % specified in y.
    if strcmpi(y, '<all>')
        [time, data, names] = getdata(nexttobj);
    else
        [time, data, names] = selectbyname(nexttobj, y);
    end

    % Error checking.
    if size(data,2) == 0
        error('Species specified do not exist.');
    end

    % Plot data. If there is only one state use different colors for runs.
    if(size(data,2) ==1)
        hLine = plot(time, data, 'color',colors(mod(i-1,numColors)+1,:), 'LineStyle', lstyle);
    else
        hLine = plot(time, data, 'LineStyle', lstyle);
    end
    handles = [handles hLine];
end

% Label.
hold off;
xlabel('Time');
```

```
ylabel('States');
title('States versus Time');

if length(tobj) == 1
    leg = legend(names, 'Location', 'NorthEastOutside');
    set(leg, 'Interpreter', 'none');
end
```

Code Modifications for Line Style

This code uses the `Time` plot type as a starting point and contains the modifications highlighted next.

```
function Time(tobj, y, lstyle)
%TIME Plots states versus time.
%
%    Plot the results of the simulation for the species with the specified
%    names versus time. Data from each run is plotted into one axes. Use the
%    Time Subplot type to plot data from each run into its own subplot. The
%    string '<all>' can be used to indicate that all species should be
%    plotted. LStyle specifies line style that can be used.
%
%    See also GETDATA, SELECTBYNAME.

if (length(tobj) > 1)
    sbioplot(tobj, @timeplotdata, [], y, lstyle);
else
    timeplotdata(tobj, [], y, lstyle);
end

%-----------------------------------------------------------
function [handles, names] = timeplotdata(tobj, x, y, lstyle)

colors    = get(gca, 'ColorOrder');
numColors = length(colors);
handles = [];
for i=1:length(tobj)
    % Get the data from the next run.
    nexttobj = tobj(i);

    % Get the simulation data associated with the species
    % specified in y.
    if strcmpi(y, '<all>')
        [time, data, names] = getdata(nexttobj);
    else
        [time, data, names] = selectbyname(nexttobj, y);
    end

    % Error checking.
    if size(data,2) == 0
        error('Species specified do not exist.');
    end

    % Plot data. If there is only one state use different colors for runs.
    if(size(data,2) ==1)
        hLine = plot(time, data, 'color',colors(mod(i-1,numColors)+1,:), 'LineStyle', lstyle);
    else
        hLine = plot(time, data, 'LineStyle', lstyle);
    end
    handles = [handles hLine];
end

% Label.
hold off;
xlabel('Time');
ylabel('States');
title('States versus Time');

if length(tobj) == 1
    leg = legend(names, 'Location', 'NorthEastOutside');
    set(leg, 'Interpreter', 'none');
end
```

5. Modify the code to customize the plot type, by typing in the section containing the code. If you change the arguments, the desktop updates the **List of Arguments Passed to Plot Type Code** table with the new list.

6. Specify additional information for the `lstyle` argument by double-clicking its row in the table. The Define Argument lstyle dialog box opens.

7. From the **Define the values that the plot type argument can be configured to** list, select `Enumerations`. This option lets you specify a comma-separated list of supported values.

8. In the **Comma separated list of supported values** box, type the values that can be entered for this plot type. The `Line Style` argument takes the following values:

   ```
   -, --, -., :, none
   ```

9. From the **Default value of the plot type argument lstyle** list, select a default value. For this example select `--`. This list is populated with the values entered in the **Comma separated list of supported values** box.

10. Click **OK** to close the Define Argument lstyle dialog box.

11. Double click the row containing the `tobj` argument. The Define Argument tobj dialog box opens.

12. From the **Define the values that the plot type argument can be configured to** list, select `Simulation Result`.

13. Click **OK** to close the Define Argument tobj dialog box.

14. Double click the row containing the `y` argument. The Define Argument y dialog box opens.

15. From the **Define the values that the plot type argument can be configured to** list, select `Logged Data`.

16. Select the **State Names (what is being logged)** check box. Leave the **Default value of the plot type argument y** as `<all>`.

17. Click **OK** to close the Define Argument y dialog box.

18 Click **Save Now** to save your new plot type. Plot types are automatically saved at regular time intervals. If **Save Now** is disabled, this means that the desktop has automatically saved the plot type.

This plot type is available for use with any task.

Visualizing Results for the Mutant Strain Using a Custom Plot

Plot the simulation data for the model of the mutant strain with dashed lines. If you have the example open and have simulated the model for the mutant strain you do not need to rerun the simulation. You can use the saved data from "Simulation Results for the Model of the Mutant Strain" on page 2-35. Follow the steps described in "Plotting the Results" on page 2-43.

If you do not have the example open do the following:

1 Load the example project by typing the following at the command line:

```
sbioloadproject gprotein
```

The model is stored in a variable called m1.

2 Open the SimBiology desktop with the model loaded by typing:

```
sbiodesktop(m1)
```

The SimBiology desktop opens with **Model Session-Heterotrimeric_G_Protein_wt**.

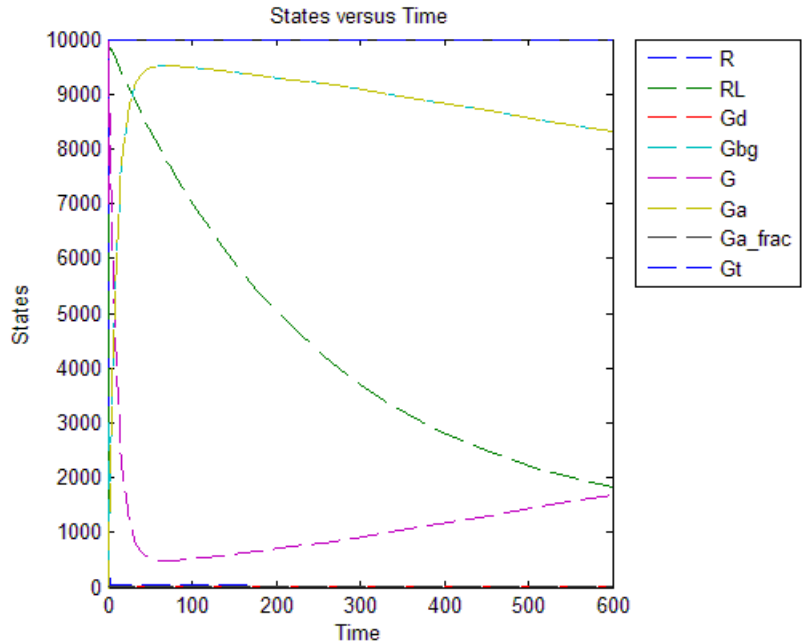
3 Select **File > Save Project As**. The Save SimBiology Project dialog box opens.

4 Specify a name (for example, `gprotein_ex`) and location for your project, and click **Save**.

Simulate the model as shown in "Simulation Results for the Model of the Mutant Strain" on page 2-35, and then continue with the steps described in "Plotting the Results" on page 2-43.

Plotting the Results

1. In the **Project Explorer**, under the **Simulation** task, click `mut_model_run1`.
2. In the **Data** pane , click the **Plots** tab.
3. From the **Plot Type** list, select `Time plot line style` and click **Add Plot Type**.
4. Under **Arguments**, select `--` from the **lstyle** list.
5. In the plot type table, clear the **Create Plot** check box for the `Time` plot.
6. Click **Plot**. Your figure should resemble the following.



Plotting Species from Two Different Data Sets

In this section...

Procedures Described in This Section

This section shows you the following:

- How to compare the active G protein fractions in the two simulations. The procedures in this section show you how to create a plot showing species from the two data sets in this tutorial.
- How to plot the data without having to rerun the simulation, and more about how to generate custom plots.

Plotting the Active G Protein Fraction from the Wild-Type Strain Model

You can find the simulation data under **Simulation** task for the model. You can also save the data from previous simulations. See "Saving Simulation Data" on page 2-32 for more information.

Specify that the species `Ga_frac` should be plotted for the wild-type model.

1. In the **Project Explorer**, under the **Simulation** task, click `wt_model_run1`. The data saved for the wild-type strain opens.
2. In the **Data** pane, click the **Plots** tab.
3. From the **Plot Type** list, select the `Time` plot and click the **Add Plot Type** button.

4 In the **Arguments** section, click  . The Select Values for y dialog box opens.

5 Click **Clear All**.

6 Select `yeast_cell.Ga_frac` and click **OK**.

7 In the plot type table, clear the **Create Plot** check boxes for the other plots, and click **Plot**. Leave the figure window open.

Creating a Custom Plot to Compare the Data

Create a custom plot that specifies that the species `Ga_frac` should be plotted with dashed lines for the *sst2Δ* model, and add a legend indicating `Ga_frac` from wild-type (`Ga_frac_wt`) and `Ga_frac` from *sst2Δ* (`Ga_frac_mut`).

1 Select **Library > ShowLibrary Explorer**. The **Library Explorer** opens.

2 In the **Library Explorer**, click **Plots Types**. The **Plot Types** pane opens.

3 In the **Enter Name** box, type a name for the custom plot and press **Enter**.

```
Time plot for Ga_frac comparisons
```

The desktop adds the new plot to the plot types table.

4 In the **Enter Plot Type code below or copy Plot Type code from** section, from the list select `Time plot with dashed line`.

5 Click **Apply**. The Plot Types command window updates to show you the code for the `Time plot with dashed line` plot.

6 Locate and change the following lines of code:

```
leg = legend(names, 'Location', 'NorthEastOutside');
set(leg, 'Interpreter', 'none');
```

to

```
leg = legend({'Ga_frac_wt','Ga_frac_mut'}, 'Location', 'NorthEastOutside');
set(leg, 'Interpreter', 'none');
```

7 Click **Save Now**. If **Save Now** is not available (grayed out), this means that the plot type has been automatically saved.

   This plot type is available for use with any task for this project.

Now you can plot `Ga_frac` data from the wild-type strain with `Ga_frac` data from the *sst2Δ*.

Plotting the Active G Protein Fraction from the Model of the Mutant Strain

Plot the *sst2Δ* `Ga_frac` data using dashed lines.

1 In the **Project Explorer**, under the **Simulation** task, click `mut_model_run1`

2 In the **Data** pane , click the **Plots** tab.

3 From the **Plot Type** list, select the `Time plot for Ga_frac comparisons` plot and click **Add Plot Type**.

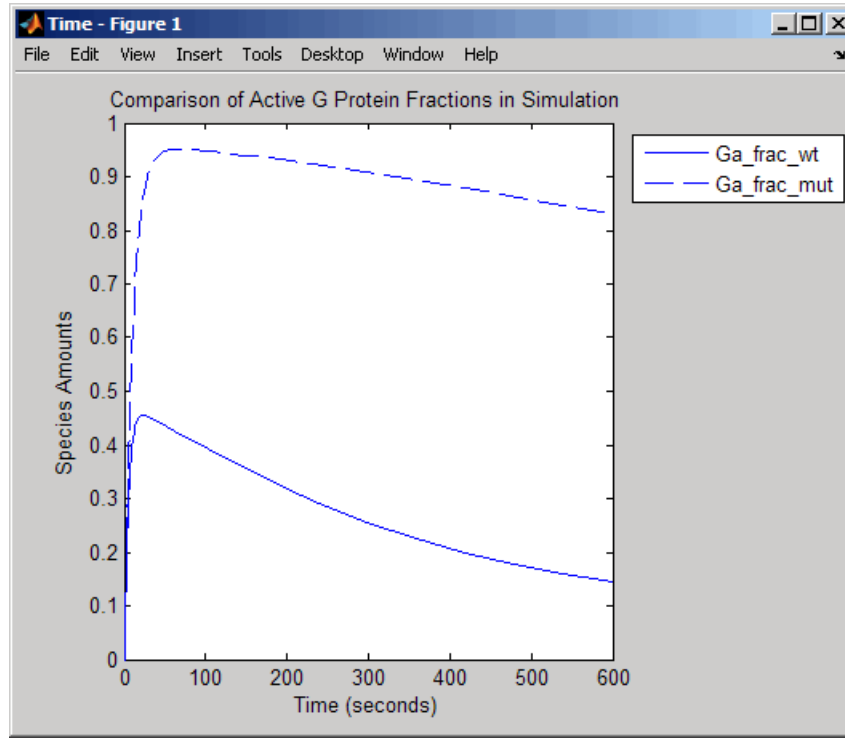
4 In the **Arguments** section, click  . The Select Values for y dialog box opens.

5 Click **Clear All**.

6 Select `yeast_cell.Ga_frac` and click **OK**.

7 In the plot type table, clear the **Create Plot** check boxes for the other plots.

8 For the newly added plot, from the **Plot Behavior** list, select `Add to current axes` and click **Plot**. This option lets you add data to the most recently generated plot. You must have the figure window open to exercise this option.


Time - Figure 1
File Edit View Insert Tools Desktop Window Help
Comparison of Active G Protein Fractions in Simulation
Species Amounts
Time (seconds)
Ga_frac_wt
Ga_frac_mut
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
100
200
300
400
500
600

Plotting Experimental Data with Simulation Data

In this section...

About the Experimental Data

You can work with your experimental data and plot your data from the SimBiology desktop. The section describes how to store the experimental data and use the custom plotting features to plot the data with your simulation data.

This example uses the yeast G protein model built in this tutorial, based on a reference paper published by Yi and colleagues [Yi et al. 2003]. The experimental data used here are also from the same reference paper.

Creating a Custom Plot for Experimental Data

1. Select **Library > Library Explorer**. The **Library Explorer** opens.
2. In the **Show Library Explorer**, click **Plots Types**. The **Plot Types** pane opens.
3. In the **Enter Name** box, type a name for the custom plot and press **Enter**.

   ```
   Ga_frac Experimental Data Plot
   ```

   The desktop adds the new plot to the plot types table.
4. In the **Plot Types** pane, copy and add the following code into the plot types editor window:

   ```
   % 1. Store the time and state data
   %(Obtained from Fig. 5 of reference paper.)
     x = [0 10 30 60 110 210 300 450 600];
     y = [0 0.35 0.4 0.36 0.39 0.33 0.24 0.17 0.2];
   ```

```
% 2. Store the estimated error values.
%(Obtained from Fig. 5 of reference paper.)
  L = [0 0.0100 0 0.0100 0.0200 0.0200 0.0300 0.0200 0.0200];
  U = [0 0.0100 0 0.0200 0.0100 0.0180 0.0350 0.0300 0.0100];

% 3. Plot the experimental data.
  errorbar(x,y,L,U,'LineStyle','none', 'Marker', '.');
  legH3 = legend('Ga_frac_sim','Ga_frac_exp' , 'Location','NorthEastOutside');
  set(legH3, 'Interpreter','none')

% To make a better picture,
  axis([0 600 0 0.5]);
```

Explanation of the Code

In step 1 you store the data as vectors in two variables, one for the experimental values of active G protein fractions (`x`), and the other for the time points (`y`). By writing scripts in this command window, you can store and process your experimental data before plotting.

In step 2 you store the estimated upper (U) and lower (L) bounds of the error values of each data point.

In step 3, the function `errorbar` enables you to plot `x` versus `y` with error bars `L(i)+U(i)` long. `x`, `y`, `L`, and `U` must be the same size. When they are vectors, each error bar is a distance of L(i) below and U(i) above the point defined by (x(i),y(i)). For more information, see `errorbar`.

The code also specifies legend location, marker style, line style, and axis scale. For more information, see `legend` and `axis`.

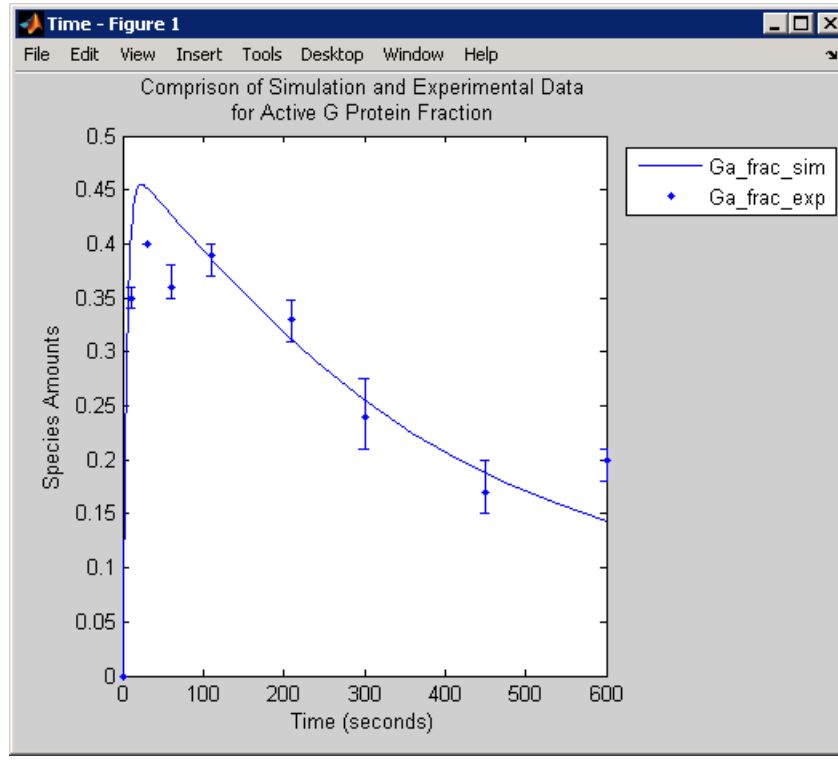
5 Click **Save Now**. If **Save Now** is not available (grayed out), this means that the plot type has been automatically saved.

This plot type is available for use with any task for this project.

Plotting the Data

You can add the new custom plot to the data plot for the model of the wild-type strain and plot the simulation data for `Ga_frac` with the experimental data.

1. In the **Project Explorer**, click **Data** for the wild-type model (`.wt_model_run1`)
2. In the **Data** pane, click the **Plots** tab.
3. From the **Plot Type** list, select the `Ga_frac Experimental Data Plot` plot and click **Add Plot Type**.
4. Select the **Create Plot** check box for the following:
   a. For the `Time` plot in which only `Ga` is specified in the **Y Arguments** list, select `New Figure` from the **Plot Behavior** list.
   b. For the `Ga_frac Experimental Data Plot`, select `Add to current axes` from the **Plot Behavior** list.
5. In the plot type table, clear the **Create Plot** check boxes for the other plots and click **Plot**.



For more information on the plot used in step 4a, see “Plotting the Active G Protein Fraction from the Wild-Type Strain Model” on page 2-44.

References

[1] Tau-Mu Yi, Hiroaki Kitano, and Melvin I. Simon. A quantitative characterization of the yeast heterotrimeric G protein cycle. PNAS (2003) vol. 100, 10764-10769.

[2] Alberts, B., Bray, D., Lewis, J., Raff, M., Roberts, K., and Watson, J.D. Molecular Biology of the Cell, 3rd edition, Garland Publishing, 1994.

3

M-Phase Control in *Xenopus* Oocyte Extracts

John Tyson's Computational Cell Biology Lab created a mathematical model for M-phase control in *Xenopus* oocyte (frog egg) extracts [Marlovits et al. 1998]. The M-phase control model shows principles by which you can apply phosphorylation and regulatory loops in your own models. Publications typically list systems of ordinary differential equations (ODEs) that represent a model system. This example shows you how to interpret these ODEs in the form of reaction pathways that are easier to represent and visualize in SimBiology software.

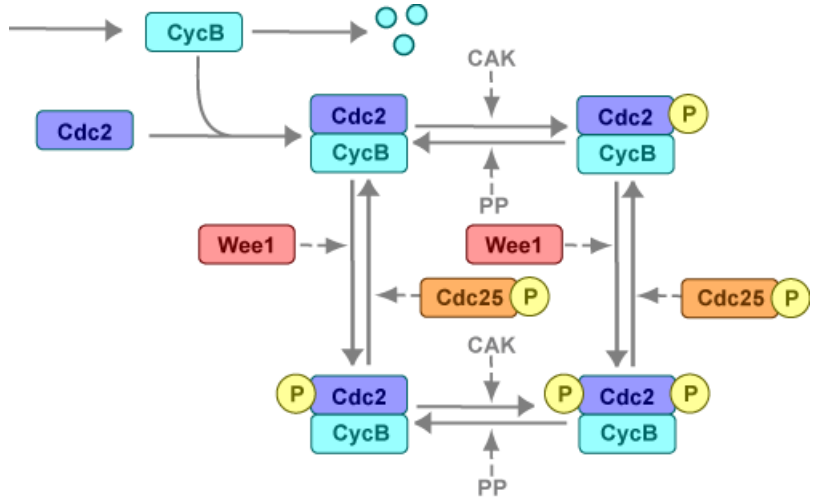
The model is centered around M-phase promoting factor (MPF). There are two positive feedback loops where MPF increases its synthesis and a negative feedback loop where MPF decreases its amount by increasing its degradation.

M-Phase Control Model

In this section...

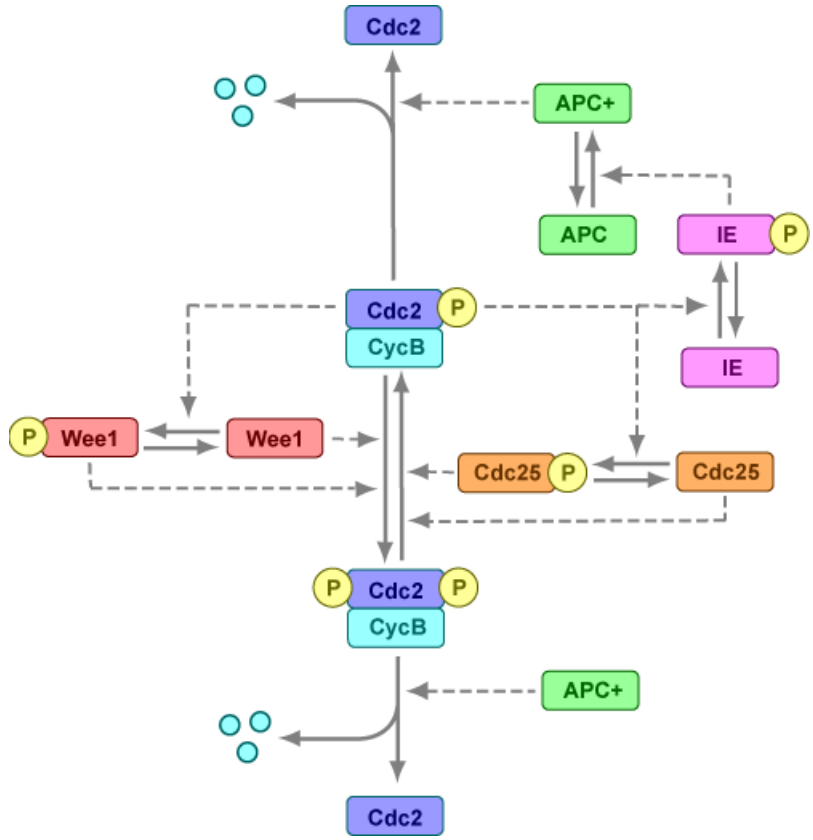
Synthesis Reactions

Cyclin B (CycB) dimerizes with Cdc2 kinase (Cdc2) to form M-phase promoting factor (MPF).



Regulation Reactions with Active MPF

Positive feedback loops with M-phase promoting factor (MPF) activate the Cdc25 phosphatase and deactivate the Wee1 kinase. A negative feedback loop with MPF activates anaphase-promoting complex (APC) that regulates the degradation of the Cyclin B subunit.


Cdc2
APC+
APC
IE
P
Cdc2
P
CycB
IE
P
Wee1
Wee1
Cdc25
P
Cdc25
P
Cdc2
P
CycB
APC+
Cdc2

M-Phase Control Equations

In this section...

About the Rate Equations in This Example

Models in systems biology are commonly described in the literature with differential rate equations. However, SimBiology software defines a model using reactions. This section shows you how to convert models published in the literature to a SimBiology format. The equation numbers match the published paper for this model [Marlovits et al. 1998]. Equations that are missing in the sequence involve the Cdk inhibitor (CKI) protein, which is not currently modeled in the SimBiology version.

Converting Differential Equations to Reactions

The rules for writing reaction and reaction rate equations from differential rate equations include not only the equations but also an understanding of

the reactions. `dx/dt` refers to the species the differential rate equation is defining. `kinetics` refers to the species in the reaction rate.

- Positive terms: Rate species are placed on right side of the reactions; reaction rate equation species are placed on the left.

$$kinetics \rightarrow \frac{dx}{dt}$$

- Negative terms: Rate species are placed on the left side of the reaction because the species are being used up in some way; reaction rate equation species are also placed on left. You need to deduce the products from additional information about the model.

$$\text{kinetics or } (\frac{\text{dx}}{\text{dt}}) \rightarrow \text{products?}$$

The following table will help you deduce the products for a reaction. In this example, by convention, phosphate groups on the right side of a species name are activating while phosphate groups on left are inhibiting.

Enzyme	Description	Reaction
`wee1`	Kinase, add inhibiting phosphate group	MPF —> P-MPF
`cdc25`	Phosphatase, remove inhibiting phosphate group	P-MPF —> MPF + P
`kcak`	Kinase, add activating phosphate group	MPF —> MPFp
`kpp`	Phosphatase, remove activating phosphate group	MPF-P —> MPF + P
`MPF`	Kinase, add activating or inhibiting phosphate group	Wee1/Cdc25/IE —> X-P or P-X
`ki`	Add inhibiting Cki	Cki + MPF —> Cki:MPF
`kir`	Remove inhibiting Cki	Cki:MPF —> Cki + MPF

Equation 1, Cyclin B

Differential rate equation for cyclin B [Marlovits et al. 1998].

$$\frac{d[\mathrm{CycB}]}{dt} = +\mathrm{k1} -\mathrm{k2}[\mathrm{CycB}] -\mathrm{k3}[\mathrm{Cdc2}][\mathrm{CycB}]$$

Rate rule using SimBiology format for the differential rate equation 1. For a model using this rule, see "SimBiology Model with Rate and Algebraic Rules" on page 3-13.

```
Rule 1  [CycB] = k1 - K2*[CycB] - k3*[Cdc2]*[CycB]
```

Reaction and reaction rate equations derived from the differential rate equation. For a model using these reactions, see "SimBiology Model with Reactions and Algebraic Rules" on page 3-21.

```
Reaction 1   AA -> CycB           v = k1
Reaction 2   CycB -> AA           v = K2*[CycB]
Reaction 3   Cdc2 + CycB -> MPF   v = k3*[Cdc2]*[CycB]
```

Equation 2, M-Phase Promoting Factor

Differential rate equation for M-phase promoting factor (MPF) [Marlovits 1998]. Note that the parameter name `kcakr` [Marlovits et al. 1998] is changed to `kpp` [Borisuk 1998] in the following reaction equations. MPF is a heterodimer of cdc2 kinase and cyclin B.

$$\begin{aligned}\frac{d[\mathrm{MPF}]}{dt} = &+\mathrm{k3}[\mathrm{Cdc2}][\mathrm{CycB}] -\mathrm{K2}[\mathrm{MPF}] \\ &+\mathrm{kpp}[\mathrm{MPFp}] -\mathrm{kcak}[\mathrm{MPF}] \\ &+\mathrm{Kcdc25}[\mathrm{pMPF}] -\mathrm{Kwee1}[\mathrm{MPF}] \\ &+\mathrm{kir}[\mathrm{Cki{:}MPF}] -\mathrm{ki}[\mathrm{MPF}][\mathrm{Cki}]\end{aligned}$$

Rate rule using SimBiology format for the differential rate equation 1. For a model using this rule, see "SimBiology Model with Rate and Algebraic Rules" on page 3-13.

```
Rule 2 MPF = kpp*MPFp - (Kwee1 + kcak + K2)*MPF +
Kcdc25*pMPF + k3*Cdc2*CycB
```

Reaction and reaction rate equations derived from the differential rate equation. For a model using these reactions, see "SimBiology Model with Reactions and Algebraic Rules" on page 3-21. A reaction name in parentheses denotes a reaction repeated in another differential rate equation.

```
(Reaction 3) Cdc2 + CycB -> MPF     v = k3*[Cdc2]*[CycB]
Reaction 4   MPF -> Cdc2 + AA       v = K2*[MPF]
Reaction 5   MPFp -> MPF            v = kpp*[MPFp]
Reaction 6   MPF -> MPFp            v = kcak*[MPF]
Reaction 7   pMPF -> MPF            v = Kcdc25*[pMPF]
Reaction 8   MPF -> pMPF            v = Kwee1*[MPF]
```

Equation 3, Inhibited M-Phase Promoting Factor

Differential rate equation for inhibited M-phase promoting factor (pMPF) [Marlovits 1998].

$$\frac{d[\text{pMPF}]}{dt} = -\text{K2}[\text{pMPF}] + \text{kpp}[\text{pMPFp}] - \text{kcak}[\text{pMPF}] + \text{Kwee1}[\text{MPF}] - \text{Kcdc25}[\text{pMPF}] + \text{kd}[\text{Cki:pMPF}]$$

Rate rule using SimBiology format for the differential rate equation 3. For a model using this rule, see "SimBiology Model with Rate and Algebraic Rules" on page 3-13.

```
Rule 3 pMPF = Kwee1*MPF - (Kcdc25 + kcak +
K2)*pMPF + kpp*pMPFp
```

Reaction and reaction rate equations derived from the differential rate equation. For a model using these reactions, see "SimBiology Model with Reactions and Algebraic Rules" on page 3-21.

```
Reaction 11  pMPF -> Cdc2 + AA      v = K2*[pMPF]
Reaction 12  pMPFp -> pMPF          v = kpp*[pMPFp]
Reaction 13  pMPF -> pMPFp          v = kcak*[pMPF]
(Reaction 8) MPF -> pMPF            v = Kwee1*[MPF]
(Reaction 7) pMPF -> MPF            v = Kcdc25*[pMPF]
```

Equation 4, Inhibited and Activated M-Phase Promoting Factor

Differential rate equation for inhibited and activated M-phase promoting factor (pMPFp) [Marlovits 1998].

$$\frac{d[\text{pMPFp}]}{dt} = \text{-K2[pMPFp]} + \text{kcak[pMPF] -kpp[pMPFp]} + \text{Kwee1[MPFp] -Kcdc25[pMPFp]} + \text{kd[Cki:pMPFp]}$$

Rate rule using SimBiology format for the differential rate equation. For a model using this rule, see "SimBiology Model with Rate and Algebraic Rules" on page 3-13.

```
Rule 4 pMPFp = Kwee1*MPFp - (kpp + Kcdc25 + K2)*pMPFp
+ kcak*pMPF
```

Reaction and reaction rate equations derived from the differential rate equation. For a model using these reactions, see "SimBiology Model with Reactions and Algebraic Rules" on page 3-21.

```
Reaction 15   pMPFp -> Cdc2 + AA       v = K2*[pMPFp]
(Reaction 13) pMPF -> pMPFp            v = kcak*[pMPF]
(Reaction 12) pMPFp -> pMPF            v = kpp*[pMPFp]
Reaction 16   MPFp -> pMPFp            v = Kwee1*[MPFp]
Reaction 17   pMPFp -> MPFp            v = Kcdc25*[pMPFp]
```

Equation 5, Activated M-Phase Promoting Factor

Differential rate equation for activated M-phase promoting factor (MPFp) [Marlovits 1998].

$$\frac{d[\text{MPFp}]}{dt} = \text{-K2[MPFp]} + \text{kcak[MPF] -kpp[MPFp]} + \text{Kcdc25[pMPFp] -Kwee1[MPFp]} + \text{kir[CKI:MPFp] -ki[CKI][MPFp]}$$

Rate rule using SimBiology format for the differential rate equation 1. For a model using this rule, see "SimBiology Model with Rate and Algebraic Rules" on page 3-13.

```
Rule 5  MPFp = kcak*MPF - (kpp + Kwee1 + K2)*MPFp
+ Kcdc25*pMPFp
```

Reaction and reaction rate equations derived from the differential rate equation. For a model using these reactions, see "SimBiology Model with Reactions and Algebraic Rules" on page 3-21.

```
Reaction 19    MPFp -> MPF + AA        v = K2*[MPFp]
(Reaction 6)  MPF -> MPFp             v = kcak*[MPF]
(Reaction 5)  MPFp -> MPF             v = kpp*[MPFp]
(Reaction 17) pMPFp -> MPFp           v = Kcdc25*[pMPFp]
(Reaction 16) MPFp -> pMPFp           v = Kwee1*[MPFp]
```

Equation 11, Cell Division Control 25

Differential rate equation for activating and deactivating Cdc25 [Marlovits 1998].

$$\frac{d[\mathrm{Cdc25p}]}{dt} = +\frac{\mathrm{k25}[\mathrm{MPFp}][\mathrm{Cdc25}]}{\mathrm{Km25}+[\mathrm{Cdc25}]} - \frac{\mathrm{k25r}[\mathrm{Cdc25p}]}{\mathrm{Km25r}+[\mathrm{Cdc25p}]}$$

Rate rule in SimBiology format for the differential rate equation 1. For a model using this rule, see "SimBiology Model with Rate and Algebraic Rules" on page 3-13. Note that since there isn't a rate rule for `Cdc25`, its amount is written as `(TotalCdc25 - Cdc25p)`.

```
Rule 11 Cdc25p = (k25*MPFp*(TotalCdc25 - Cdc25p))/(Km25 +
(TotalCdc25 - Cdc25p)) - (k25r*PPase*Cdc25p)/(Km25r + Cdc25p)
```

Reaction and reaction rate equations derived from the differential rate equation. For a model using these reactions, see "SimBiology Model with Reactions and Algebraic Rules" on page 3-21.

```
Reaction 36 Cdc25 -> Cdc25p,  v = k25*[MPFp]*[Cdc25]/(Km25 + [Cdc25])
Reaction 37 Cdc25p -> Cdc25,  v = k25r*[Cdc25p]/(Km25r + [Cdc25p])
```

Equation 12, Wee1 Activation/Deactivation

Differential rate equation for activating and deactivating Wee1 kinase [Marlovits 1998]. The kinase (MPFp) phosphorylates active Wee1 (Wee1) to its inactive form (Wee1p). The dephosphorylation of inactive Wee1 (Wee1p) is by an unknown phosphatase.

$$\frac{d[Wee1]}{dt} = -\frac{kw[MPFp][Wee1]}{Kmw + [Wee1]} + \frac{kwr[Wee1P]}{Kmwr + [Wee1P]}$$

Rate rule in SimBiology format for the differential rate equation 1. For a model using this rule, see "SimBiology Model with Rate and Algebraic Rules" on page 3-13.

```
Rule 12  Wee1p = (kw*MPFp*(TotalWee1 - Wee1p))/(Kmw + (TotalWee1 - Wee1p))
                                      - (kwr*Wee1p)/(Kmwr + Wee1p)
```

Reaction and reaction rate equations derived from the differential rate equation. For a model using these reactions, see "SimBiology Model with Reactions and Algebraic Rules" on page 3-21.

```
reaction 38 Wee1 -> Wee1p, v = (kw*[MPFp]*[Wee1])/(Kmw + [Wee1])
reaction 39 Wee1p -> Wee1, v = (kwr*[Wee1p])/(Kmwr + [Wee1p])
```

Equation 13, Intermediate Enzyme Activation/Deactivation

Differential rate equation for activating and deactivating the intermediate enzyme (IE) [Marlovits 1998]. The active kinase (MPFp) phosphorylates the inactive intermediate enzyme (IE) to its active form (IEp).

$$\frac{d[IEp]}{dt} = +\frac{kie[MPFp][IE]}{Kmie + [IE]} - \frac{kier[IEp]}{Kmier + [IEp]}$$

Rate rule in SimBiology format for the differential rate equation 1. For a model using this rule, see "SimBiology Model with Rate and Algebraic Rules" on page 3-13.

```
Rule 13 IEp = (kie*MPFp*(TotalIE - IEp))/(Kmie + (TotalIE - IEp))
                 - (kier*IEp)/(Kmier + IEp)
```

Reaction and reaction rate equations derived from the differential rate equation. For a model using these reactions, see “SimBiology Model with Reactions and Algebraic Rules” on page 3-21.

```
reaction 40 IE -> IEp, v = (kie*[MPFp]*[IE])/(Kmie + [IE])
reaction 41 IEp -> IE, v = (kier*[IEp])/(Kmier + [IEp])
```

Equation 14, APC Activation/Deactivation

Differential rate equation for [Marlovits 1998].

$$\frac{d[APCa]}{dt} = +\frac{kap[IEP][APCi]}{Kmap + [APCi]} - \frac{kapr[APCa]}{Kmapr + [APCa]}$$

Rate rule in SimBiology format for the differential rate equation 1. For a model using this rule, see “SimBiology Model with Rate and Algebraic Rules” on page 3-13.

```
Rule 14 APCa = (kap*IEp*(TotalAPC - APCa))/(Kmap + (TotalAPC - APCa))
                - (kapr*APCa)/(Kmapr + APCa)
```

Reaction and reaction rate equations derived from the differential rate equation. For a model using these reactions, see “SimBiology Model with Reactions and Algebraic Rules” on page 3-21.

```
Reaction 42 APCi -> APCa, v = (kap*[IEp]*[APCi])/(Kmap + [APCi])
Reaction 43 APCa -> APCi,  v = (kapr*[APCa])/(Kmapr + [APCa])
```

Equation 17, Rate Parameter K2

Algebraic equation to define the rate parameter `K2` [Marlovits 1998]. Inactive APC (APCi) is catalyzed by IE (intermediate enzyme) to active APC (APCa).

$$k2 = V2'[APC] + V2''[APC']$$

Algebraic rule in SimBiology format for the algebraic equation 17. For a model using this rule, see “SimBiology Model with Rate and Algebraic Rules” on page 3-13.

```
Algebraic Rule 17  V2i*(TotalAPC - APCa) + V2a*APCa - K2
```

Algebraic rule when simulating with reactions. For a model using this rule with reactions, see “SimBiology Model with Reactions and Algebraic Rules”

on page 3-21. `V2'` is renamed to `V2i` and `V2` is renamed to `V2a`. APCi (APC) is the inactive form of the enzyme while APCa (APC') is the active form. `K2` is the independent variable.

```
Algebraic Rule 1 (V2i*APCi) + (V2a*APCa) - K2
```

Equation 18, Rate Parameter Kcdc25

Algebraic equation to define the rate parameter `Kcdc25` [Marlovits 1998]. Inactive Cdc25 (`Cdc25`) is phosphorylated by MPF to active Cdc25 (Cdc25p).

$$kcdc25 = V_{25}'[Cdc25] + V_{25}''[Cdc25p]$$

Algebraic rule in SimBiology format for the algebraic equation 18. For a model using this rule, see "SimBiology Model with Rate and Algebraic Rules" on page 3-13.

```
Algebraic Rule 18  V25i*(TotalCdc25 - Cdc25p) + V25a*Cdc25p - Kcdc25
```

Algebraic rule when simulating with reactions. `Kcdc25` is the independent variable. For a model using this rule with reactions, see "SimBiology Model with Reactions and Algebraic Rules" on page 3-21.

```
Algebraic Rule 2 (V25i*Cdc25) + (V25a*Cdc25p) - Kcdc25
```

Equation 19, Rate Parameter Kwee1

Algebraic equation to define the rate parameter [Marlovits 1998]. Active Wee1 (`Wee1`) is phosphorylated by MPF to inactive Wee1 (`Wee1p`).

$$k_{wee1} = V_{wee1}'[Wee1p] + V_{wee1}''[Wee1]$$

Algebraic rule in SimBiology format for rate parameter equation 19. For a model using this rule, see "SimBiology Model with Rate and Algebraic Rules" on page 3-13.

```
Algebraic Rule 19  Vwee1i*Wee1p + Vwee1a*(TotalWee1 - Wee1p) - Kwee1
```

Algebraic rule when simulating with reactions. `Kwee1` is the independent variable. For a model using this rule with reactions, see "SimBiology Model with Reactions and Algebraic Rules" on page 3-21.

```
Algebraic Rule 3 (Vwee1i*Wee1p) + (Vwee1a*Wee1) - Kwee1
```

SimBiology Model with Rate and Algebraic Rules

In this section...

Overview

There is one rate rule for each equation defining a species and one algebraic rule for each variable parameter in the M-phase control model [Marlovits 1998]. For a list and description of the equations, see "M-Phase Control Equations" on page 3-4.

A basic model includes rate rules 1 to 5 and 11 to 14 with algebraic rules 17, 18, and 19.

Writing Differential Rate Equations as Rate Rules

Writing differential rate equations in an unambiguous format that a software program can understand is a simple process when you follow the syntax rules for programming languages.

- Use an asterisk to indicate multiplication. For example, `k[A]` is written `k*A` or `k*[A]`. The brackets around the species `A` do not indicate concentration.
- SimBiology uses square brackets around species and parameter name to allow names that are not valid MATLAB variable names. For example, you could have a species named `glucose-6-phosphate dehydrogenase` but you need to add brackets around the name in reaction rate and rule equations.

  ```
  [glucose-6-phosphate dehydrogenase]
  ```

- Use parentheses to clarify the order of evaluation for mathematical operations. For example, do not write Henri-Michaelis-Menten reaction rates as `Vm*C/Kd + C`, because `Vm*C` is divided by `Kd` before adding `C` to the result. Instead, write this reaction rate as `(Vm*C)/(Kd + C)`.

Species

The following table lists species in the model with their initial amounts. There are three variable parameters modeled as species (`K2`, `Kcdc25`, and `KWee1`). You could also model the variable parameters as parameters with the property **ConstantAmount** cleared.

Name	InitialAmount	InitialAmountUnits	ConstantAmount
CycB	0.0		☐
MPF	0.0		☐
pMPF	0.0		☐
pMPFp	0.0		☐
MPFp	0.0		☐
Cdc25p	0.0		☐
Wee1p	0.0		☐
IEp	0.0		☐
APCa	0.0		☐
Kcdc25	0.0		☐
Kwee1	0.0		☐
K2	0.0		☐
Wee1	0.0		☐
TotalCdc25	1.0		☑
TotalWee1	1.0		☑
TotalAPC	1.0		☑
TotalIE	1.0		☐
PPase	1.0		☑
AntiAPC	1.0		☑
Cdc2	100.0		☑

Parameters

The following table lists parameters in the model with their initial values. The property **ConstantValue** is selected for all of the parameters.

Name	Value	ValueUnits	ConstantValue
Vweea	1.0		☑
k1	1.0		☑
Kmwr	1.0		☑
Kmapr	1.0		☑
Km25r	1.0		☑
kcak	0.64		☑
V2a	0.25		☑
V25a	0.17		☑
kier	0.15		☑
kapr	0.13		☑
kap	0.13		☑
kwr	0.1		☑
k25r	0.1		☑
Kmw	0.1		☑
Km25	0.1		☑
kie	0.02		☑
kw	0.02		☑
k25	0.02		☑
V25i	0.017		☑
Vweei	0.01		☑
Kmier	0.01		☑
Kmie	0.01		☑
Kmap	0.01		☑
V2i	0.0050		☑
k3	0.0050		☑
kpp	0.0040		☑

Rate Rule 1, Cyclin B (CycB)

The rate rule is from "Equation 1, Cyclin B" on page 3-6.

```
rate rule: CycB = k1 - K2*CycB - k3*Cdc2*CycB
```

```
    species: CycB = 0 nM
             Cdc2 = 100 nM, [x]constant
parameters: k1 = 1 nM/minute
            K2 = 0 1/minute, []constant
            k3 = 0.005 1/(nM*minute)
```

`K2` is a variable rate parameter whose value is defined by an algebraic rule. See "Algebraic Rule 17, Rate Parameter K2" on page 3-19. Its value varies from `0.005` to `0.25 1/minute`.

Rate Rule 2, M-Phase Promoting Factor (MPF)

The rate rule is from "Equation 2, M-Phase Promoting Factor" on page 3-6.

```
 rate rule: MPF = kpp*MPFp - (Kwee1 + kcak + K2)*MPF + Kcdc25*pMPF
                    + k3*Cdc2*CycB
   species: MPF = 0 nM
            MPFp = 0 nM
            pMPF = 0 nM
parameters: kpp = 0.004 1/minute
            kcak = 0.64 1/minute
            k3 = 0.005 1/(nM*minute)
            K2 = 0 1/minute
            Kcdc25 = 0 1/minute
            Kwee1 = 0 1/minute
```

`K2`, `Kcdc25`, and `Kwee1` are variable rate parameters whose values are defined by algebraic rules. See "Algebraic Rule 17, Rate Parameter K2" on page 3-19, "Algebraic Rule 18, Rate Parameter Kcdc25" on page 3-19, and "Algebraic Rule 19, Rate Parameter Kwee1" on page 3-19.

Rate Rule 3, Inhibited M-Phase Promoting Factor (pMPF)

The rate rule is from "Equation 3, Inhibited M-Phase Promoting Factor" on page 3-7.

```
rate rule: pMPF = Kwee1*MPF - (Kcdc25 + kcak + K2)*pMPF + kpp*pMPFp
```

Rate Rule 4, Activated but Inhibited M-Phase Promoting Factor (pMPFp)

The rate rule is from "Equation 4, Inhibited and Activated M-Phase Promoting Factor" on page 3-8.

```
rate rule: pMPFp = Kwee1*MPFp - (kpp + Kcdc25 + K2)*pMPFp + kcak*pMPF
```

Rate Rule 5, Activated M-Phase Promoting Factor (MPFp)

The rate rule is from "Equation 5, Activated M-Phase Promoting Factor" on page 3-8.

```
rate rule: MPFp = kcak*MPF - (kpp + Kwee1 + K2)*MPFp + Kcdc25*pMPFp
```

Rate Rule 11, Activated Cdc25 (Cdc25p)

The rate rule is from "Equation 11, Cell Division Control 25" on page 3-9.

```
rate rule: Cdc25p = (k25*MPFp*(TotalCdc25 - Cdc25p))/(Km25 + (TotalCdc25 - Cdc25p))
                      - (k25r*PPase*Cdc25p)/(Km25r + Cdc25p)
```

Rate Rule 12, Inhibited Wee1 (Wee1p)

The rate rule is from "Equation 12, Wee1 Activation/Deactivation" on page 3-10.

```
rate rule: Wee1p = (kw*MPFp*(TotalWee1 - Wee1p))/(Kmw + (TotalWee1 - Wee1p))
                     - (kwr*PPase*Wee1p)/(Kmwr + Wee1p)
```

Rate Rule 13, Activated Intermediate Enzyme (IEp)

The rate rule is from "Equation 13, Intermediate Enzyme Activation/Deactivation" on page 3-10.

```
rate rule: IEp = (kie*MPFp*(TotalIE - IEp))/(Kmie + (TotalIE - IEp))
                       - (kier*PPase*IEp)/(Kmier + IEp)
```

Rate Rule 14, Activated APC (APCa)

The rate rule is from "Equation 14, APC Activation/Deactivation" on page 3-11.

```
rate rule: APCa = (kap*IEp*(TotalAPC - APCa))/(Kmap + (TotalAPC - APCa))
                    - (kapr*AntiAPC*APCa)/(Kmapr + APCa)
```

Algebraic Rule 17, Rate Parameter K2

`K2` is a variable rate parameter whose value is determined by the amount of active and inactive `APC`. The algebraic rule is from "Equation 17, Rate Parameter K2" on page 3-11.

```
algebraic rule: V2i*(TotalAPC - APCa) + V2a*APCa - K2
       species: APCi = 1 nM
                APCa = 0 nM
                TotalAPC = 1 nM [x]constant
    parameters: K2  = 0 or 0.25 1/minute, []constant
                V2i = 0.005 1/(nM*minute)
                V2a = 0.25  1/(nM*minute)
```

Algebraic Rule 18, Rate Parameter Kcdc25

`Kcdc25` is a variable rate parameter whose value is determined by the amount of active and inactive `Cdc25`. The algebraic rule is from "Algebraic Rule 18, Rate Parameter Kcdc25" on page 3-19.

```
algebraic rule: V25i*(TotalCdc25 - Cdc25p) + V25a*Cdc25p - Kcdc25
```

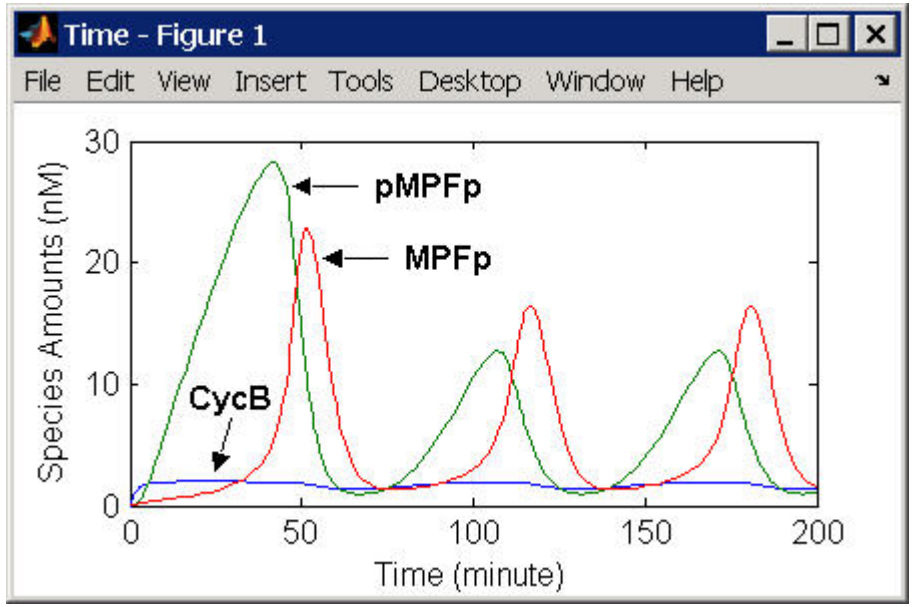
Algebraic Rule 19, Rate Parameter Kwee1

`Kwee1` is a variable rate parameter whose value is determined by the amount of active and inactive `Wee1`. The algebraic rule is from "Equation 19, Rate Parameter Kwee1" on page 3-12.

```
algebraic rule: Vweei*Wee1p + Vweea*(TotalWee1 - Wee1p) - Kwee1
```

Simulation of the M-Phase Control Model with Rules

This is a simulation of the M-phase control model using rate and algebraic rules. Simulate with the `ode15s` solver and plot species `CycB`, `pMPFp`, and `MPFp`. For a description of the model, see "SimBiology Model with Rate and Algebraic Rules" on page 3-13.



If you want to run the simulation, you can open the model within the SimBiology desktop.

1. Open the SimBiology desktop using the function `sbiodesktop`.
2. From the **File** menu, select **Open Project**.
3. Browse to the file `m_phase_xenopus.sbproj` in the following directory:

   matlabroot`/toolbox/simbio/simbiodemos/m_phase_xenopus.sbproj`

For a model using reactions and algebraic rules, see "SimBiology Model with Reactions and Algebraic Rules" on page 3-21.

SimBiology Model with Reactions and Algebraic Rules

In this section...

In this section...

Overview

There can be one or more reactions for an equation defining a species and one algebraic rule for each variable parameter in the M-phase control model [Marlovits 1998]. For a list and description of the equations, see "M-Phase Control Equations" on page 3-4.

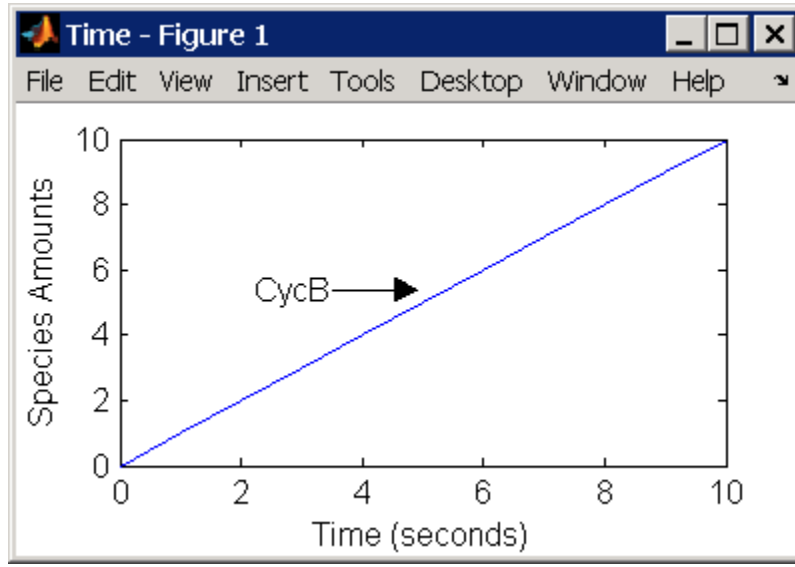
A basic model includes reactions 1 to 8, 11 to 13, 15 to 17, 19, and 36 to 43 with algebraic rules from equations 17, 18, and 19.

Reaction 1, Synthesis of Cyclin B

Cyclin B is synthesized at a constant rate.

```
       reaction: AA -> CycB
  reaction rate: k1 nM/minute
      parameter: k1 = 1 nM/minute
        species: CycB = 0 nM
                 AA = 100 nM [x]constant [x]boundary
```

Simulate reaction 1 with the `ode15s` solver.



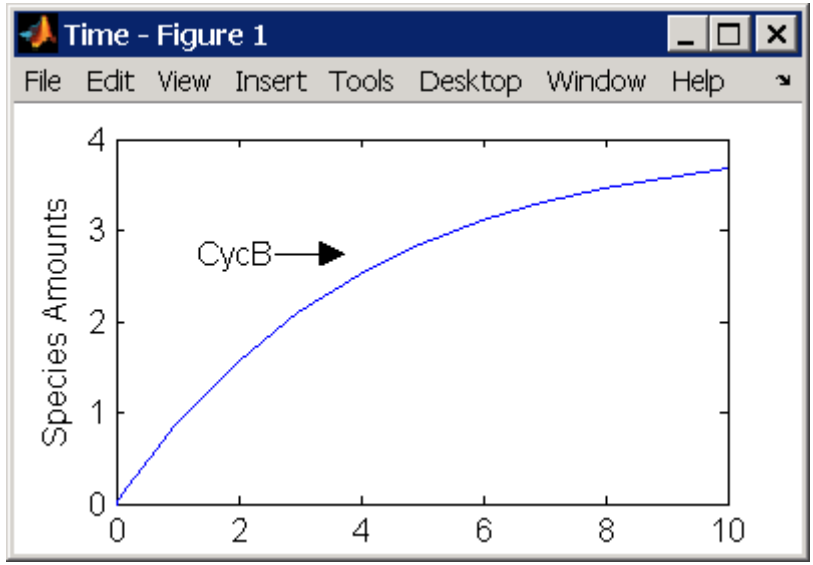
Reaction 2, Degradation of Cyclin B

Cyclin B is degraded at the end of the M-phase.

```
        reaction: CycB -> AA
   reaction rate: K2*CycB nM/minute
      parameters: K2  = 0 1/minute, []constant, variable by rule
                  V2i = 0.005 1/nM*minute
                  V2a = 0.25  1/nM*minute
         species: CycB = 0 nM
                  APCi = 1 nM
                  APCa = 0 nM
                  AA = 100 nM [x]constant [x]boundary
  algebraic rule: (V2i*APCi) + (V2a*APCa) - K2
```

Initially, Cyclin B degradation is low. This implies the amount of active APC (APCa) = 0 and inactive APC (APCi) = APCtotal = 1 nM.

Test the algebraic rule by simulating reactions 1 and 2 with `APCi = 0` and `APCa = 1`.

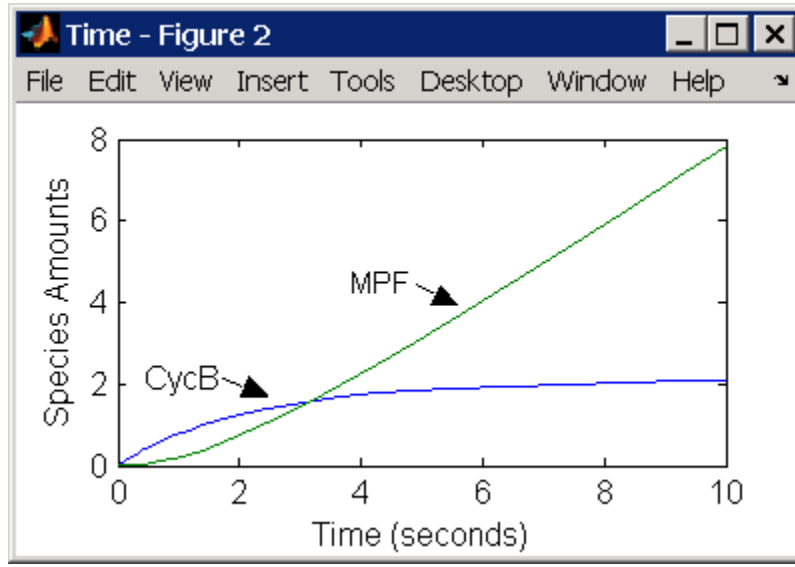


Reaction 3, Dimerization of Cyclin B with Cdc2 Kinase

Cyclin B dimerizes with Cdc2 kinase to form M-phase promoting factor (MPF).

```
      reaction: Cdc2 + CycB -> MPF
 reaction rate: k3*Cdc2*CycB nM/minute
    parameters: k3 =   0.005 1/(nM*minute)
       species: Cdc2 = 100     nM
                CycB =   0     nM
                MPF  =   0     nM
```

Test the model by simulating with `K2 = 0.25`.

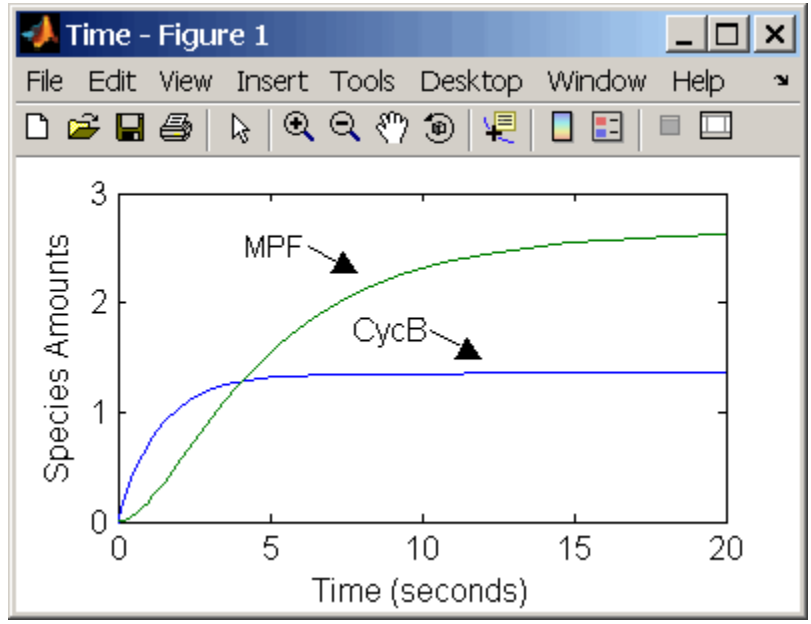


Reaction 4, Degradation of Cyclin B on MPF

Cyclin B is tagged with ubiquitin groups and degrades while bound to Cdc2.

```
       reaction: MPF -> Cdc2 + AA
  reaction rate: K2*[MPF]
     parameters: K2 = 0 or 0.25 1/minute, variable by rule
                 v2i = 0.005 1/(nM*minute)
                 v2a = 0.25 1/(nM*minute)
        species: MPF = 0 nM
                 APCi = 1 nM
                 APCa = 0 nM
                 AA = 100 nM [x]constant [x]boundary
algebraic rule: (v2i*APCi) + (v2a*APCa) - K2
```

Test the simulation with `APCa = 1` and `APCi = 0`. Because the amount of APCa (active) is high, K2 increases and the degradation starts to balance the synthesis of MPF.



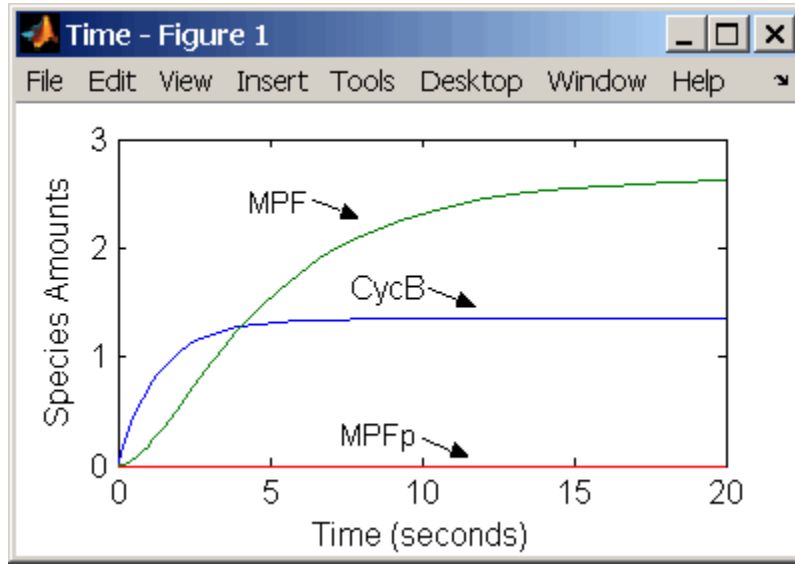
Reaction 5, Deactivation of Active MPF

Active MPF (MPFp) is dephosphorylated on Thr-161 by an unknown phosphatase (PP) to inactive MPF (MPF).

```
       reaction: MPFp -> MPF
  reaction rate: kpp*[MPFp]
     parameters: kpp = 0.004 1/minute
        species: MPFp = 0 nM
                 MPF = 0 nM
```

`kcakr = 0.004` 1/minute [Marlovits 1998, p. 175], but is renamed to `kpp` [Borisuk 1998].

Test simulation with APCa = 1 and APCi = 0. MPF increases without reaching steady state.



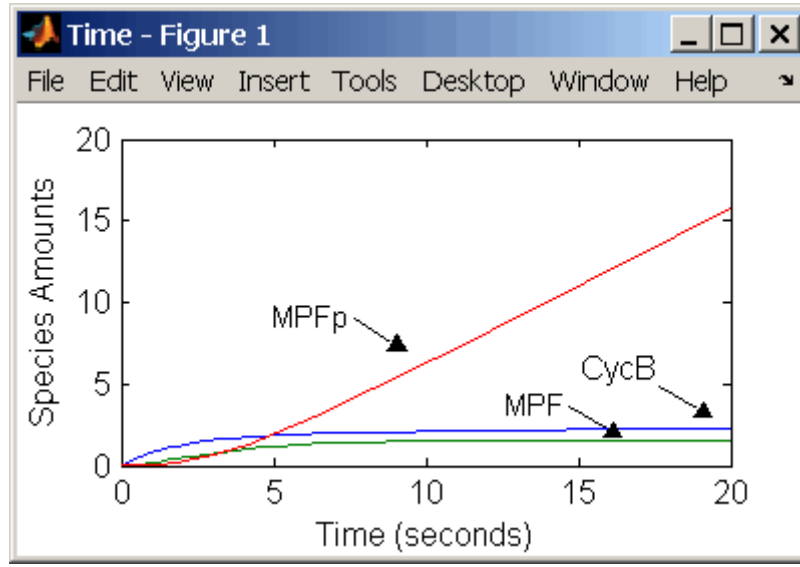
Reaction 6, Activation of MPF

Inactive MPF (MPF) is phosphorylated on Thr-161 by an unknown cyclin activating kinase (CAK).

```
      reaction: MPF -> MPFp
reaction rate: kcak*[MPF]
   parameters: kcak = 0.64 1/minute
      species: MPF = 0 nM
               MPFp = 0 nM
```

The kinase reaction that phosphorylates MPF to the active form is 160 times faster than the phosphatase reaction that dephosphorylates active MPF.

Simulate the model with reactions 1 to 6. Notice that after adding reaction 6, most of the product goes to active MPF (MPFp).



Reaction 7, Remove Inhibiting Phosphate from Inhibited MPF

Cdc25 phosphatase removes the inhibiting phosphate groups at the threonine 14 and tyrosine 15 residues on Cdc2 kinase.

```
      reaction: pMPF -> MPF
 reaction rate: Kcdc25*[pMPF]
    parameters: Kcdc25 = 0.0 1/minute or 0.017 1/minute, variable by
                                                        algebraic rule
                V25i = 0.017 1/(mM*minute)
                V25a = 0.17 1/mM*minute
       species: pMPF = 0 nM
                MPF = 0 nM
                Cdc25 = 1 nM (inactive)
                Cdc25p = 0 nM (active)
algebraic rule: (V25i*Cdc25) + (V25a*Cdc25p) - Kcdc25
```

Initially, all of the Cdc25 phosphatase is in the inactive form (Cdc25).

Enter the initial value for Kcdc25 as 0.0 and let the first time step calculate the value from the rule, or enter an initial value using the rule.

Initially, set **ConstantAmount** for Cdc25 and Cdc25p to test reactions 1 through 7. Then after you can add the reactions to regulate the Cdc25 phosphatase by clearing the **ConstantAmount** property.

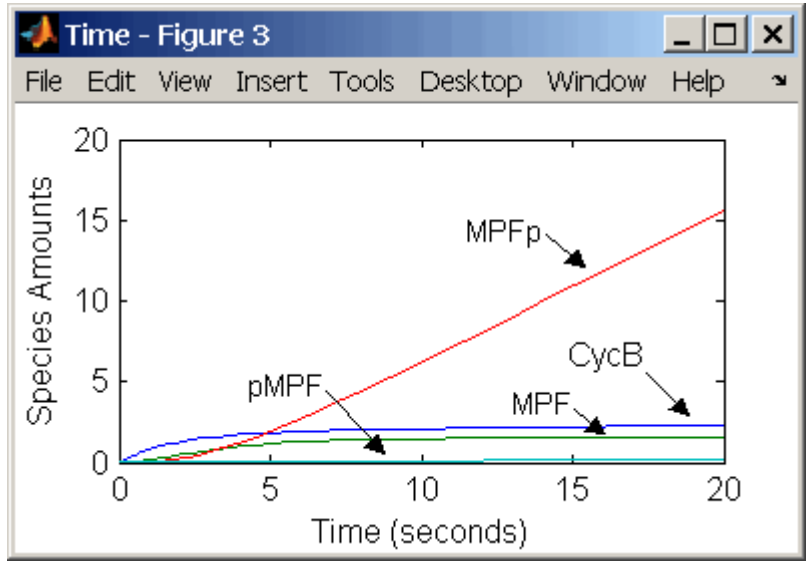
Reaction 8, Inhibition of MPF by Phosphorylation

Addition of inhibiting phosphate groups by Wee1 kinase to inhibit active M-phase promoting factor (MPF). Myt1 kinase is also involved with the phosphorylation, but its contribution is grouped with Wee1.

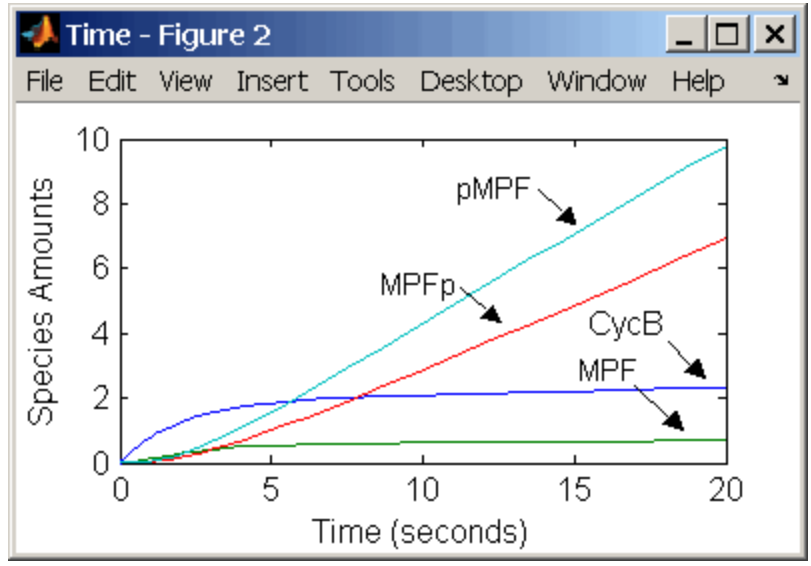
```
       reaction: MPF -> pMPF
  reaction rate: Kwee1*[MPF]
     parameters: Kwee1 = 0.0 1/minute or 0.01 1/minute, variable by
                                                        algebraic rule
                 Vwee1i = 0.01 1/(nM*minute)
                 Vwee1a = 1.0 1/(nM*minute)
        species: MPF = 0 nM
                 pMPF = 0 nM
                 Wee1p = 1 nM (inactive)
                 Wee1 = 0 nM  (active)
  algebraic rule: (Vwee1i*Wee1p) + (Vwee1a*Wee1) - Kwee1
```

The initial capitalization for the parameter Kwee1 is a convention to indicate that this value changes during the simulation.

Test the simulation for reactions 1 through 8 with `Wee1p` (inactive) = `1` and `Wee1` (active) = `0`.



Test the simulation with `Wee1p` (inactive) = `0` and `Wee1` (active) = `1`.



Reaction 11, Degradation of Cyclin B on Inhibited MPF

Degradation of cyclin B (`CycB`) on inhibited MPF (`pMPF`). Cyclin B is tagged with ubiquitin groups and degrades while bound to Cdc2.

```
        reaction: pMPF -> Cdc2 + AA
   reaction rate: K2*[pMPF] nM/minute
      parameters: K2 = 0 or 0.25 1/minute, variable by rule
                  V2i = 0.005 1/nM*minute
                  V2a = 0.25 1/nM*minute
         species: MPF = 0 nM
                  APCi = 1 nM
                  APCa = 0 nM
                  AA = 100 nM [x]constant [x]boundary
                  Cdc2 = 100 nm
  algebraic rule: (V2i*APCi) + (V2a*APCa) - K2
```

Test the simulation with Wee1 active (`Wee1 = 1`) and APC active (`APCi = 1`).

Reaction 12, Deactivation of MPF to Inhibited MPF

Inhibited/active MPF (pMPFp) is dephosphorylated on Thr-161 by an unknown phosphatase (PP) to inhibited MPF (pMPF). Compare reaction 12 with reaction 5.

```
       reaction: pMPFp -> pMPF
  reaction rate: kpp*[pMPFp]
     parameters: kpp = 0.004 1/minute
        species: pMPFp = 0 nM
                 pMPF  = 0 nM
```

Reaction 13, Activation of Inhibited MPF

Inhibited MPF (pMPF) is phosphorylated on Thr-161 by an unknown cyclin-activating kinase (CAK). Compare reaction 13 with reaction 6.

```
       reaction: pMPF -> pMPFp
  reaction rate: kcak*[pMPF] nM/minute
     parameters: kcak = 0.64 1/minute
        species: pMPF = 0 nM
```

```
                pMPFp = 0 nM
```

Test the simulation with Wee1p = 1 (inactive)/ Wee1 = 0 and then test with Wee1p = 0 (inactive)/ Wee1 = 1.

Reaction 15, Degradation of Cyclin B on Active but Inhibited MPF

Degradation of cyclin B (`CycB`) on inhibited MPF (`pMPF`). Cyclin B is tagged with ubiquitin groups and degrades while bound to cdc2 kinase.

```
       reaction: pMPFp -> Cdc2 + AA
  reaction rate: K2*[pMPFp] nM/minute
     parameters: K2 = 0 or 0.25 1/minute, variable by rule
                 v2i = 0.005 1/nM*minute
                 v2a = 0.25 1/nM*minute
        species: MPF = 0 nM
                 APCi = 1 nM
                 APCa = 0 nM
                 AA = 100 nM [x]constant [x]boundary
                 Cdc2 = 100 nm
algebraic rule: (V2i*APCi) + (V2a*APCa) - K2
```

Reaction 16, Inhibit MPF by Phosphorylation

Addition of inhibiting phosphate groups by Wee1 kinase to inhibit active M-phase promoting factor (MPF). Myt1 kinase is also involved with the phosphorylation, but its contribution is grouped with Wee1.

```
       reaction: MPFp -> pMPFp
  reaction rate: Kwee1*[MPFp] nM/minute
     parameters: Kwee1 = 1/minute []constant, variable by rule
                 Vweei = 0.01 1/nM*minute
                 Vweea = 1 1/nM*minute
        species: MPFp = 0 nM
                 pMPFp = 0 nM
                 Wee1p = 1 nM (inactive)
                 Wee1 = 0 nM (active)
algebraic rule: (Vwee1i*Wee1p) + (Vwee1a*Wee1) - Kwee1
```

Reaction 17, Remove Inhibiting Phosphate from Activated MPF

Remove the inhibiting phosphate group from pMPFp with cdc25 phosphatase.

```
      reaction: pMPFp -> MPFp
reaction rate: Kcdc25*[pMPFp]
   parameters: Kcdc25 = 0 1/minue, []constant, variable by rule
               V25i = 0.017 1/nM*minute
               V25a = 0.17 1/nM*minute
      species: pMPFp = 0 nM
               MPFp = 0 nM
algebraic rule: (V25i*Cdc25) + (V25a*Cdc25p) - Kcdc25
```

Reaction 19, Degradation of Cyclin B on Activated MPF

Degradation of cyclin B (`CycB`) on inhibited MPF (`pMPF`). Cyclin B is tagged with ubiquitin groups and degrades while bound to cdc2 kinase.

```
      reaction: MPFp -> MPF + AA
 reaction rate: K2*[MPFp] nM/minute
   parameters: K2 = 0 or 0.25 1/minute, variable by rule
               V2i = 0.005 1/nM*minute
               V2a = 0.25 1/nM*minute
      species: MPF = 0 nM
               MPFp = 0 nM
               APCi = 1 nM
               APCa = 0 nM
               AA = 100 nM [x]constant [x]boundary
               Cdc2 = 100 nm
algebraic rule: (V2i*APCi) + (V2a*APCa) - K2
```

Reaction 36, Activation of Cdc25 by Activated MPF

Activation of cdc25 phosphatase by phosphorylation with active M-phase promoting factor (`MPFp`).

```
      reaction: Cdc25 + (MPFp) -> Cdc25p + (MPFp)
reaction rate: (k25*[MPFp]*[Cdc25])/(Km25 + [Cdc25])
   parameters: k25 = 0.02 1/minute
```

```
              Km25 = 0.1 nM
        species: Cdc25 = 1 nM (inactive)
              Cdc25p = 0 nM (active)
```

Initially MPF is inhibited (MPF* reacts to pMPF*).

Reaction 37, Deactivation of Cdc25

Deactivation of cdc25 phosphatase by dephosphorylation with an unknown phosphatase.

```
      reaction: Cdc25p -> Cdc25
reaction rate: (k25r*[Cdc25p])/(Km25r + [Cdc25p])
   parameters: k25r = 0.1 nM/minute
              Km25r = 1 nM
       species: Cdc25 = 1 nM (inactive)
              Cdc25p = 0 nM (active)
```

Reaction 38, Deactivation of Wee1 by Active MPF

Deactivation of Wee1 kinase by phosphorylation with active M-phase promoting factor (MPFp).

```
      reaction: Wee1 + (MPFp) -> Wee1p + (MPFp)
reaction rate: (kw*[MPFp]*[Wee1])/(Kmw + [Wee1]) nM/minute
   parameters: kw = 0.02 1/minute
              Kmw = 0.1 nM
       species: Wee1p = 1 nM (inactive)
              Wee1 = 0 nM (active)
```

Initially MPF is inhibited (MPF* reacts to pMPF*).

Reaction 39, Activation of Wee1

Activation of Wee1 kinase by dephosphorylation with an unknown kinase.

```
      reaction: Wee1p -> Wee1
reaction rate: (kwr*[Wee1p])/(Kmwr + [Wee1p]) nM/minute
   parameters: kwr = 0.1 nM/minute
              Kmwr = 1 nM
       species: Wee1p = 1 nM (inactive)
```

```
                Wee1 = 0 nM (active)
```

Reaction 40, Activation of Intermediate Enzyme by Active MPF

The inactive intermediate enzyme (IE) is activated by phosphorylation with active M-phase promoting factor (MPFp).

```
       reaction: IE + (MPFp) -> IEp + (MPFp)
  reaction rate: (kie*[MPFp]*[IE])/(Kmie + [IE])
     parameters: kie = 0.02 1/minute
                 Kmie = 0.01nM
        species: IE = 1 nM (inactive)
                 IEp = 0 nM (active)
```

Reaction 41, Deactivation of IE

The active intermediate enzyme (IE) is deactivated by dephosphorylation.

```
       reaction: IEp -> IE
  reaction rate: (kier*[IEp])/(Kmier + [IEp])
     parameters: kier = 0.15 nM/minute
                 Kmier = 0.01 nM
        species: IE = 1 nM (inactive)
                 IEp = 0 nM (active)
```

Reaction 42, APC Activation by IEp

Anaphase-promoting complex (APC) is activated by an active intermediate enzyme (IEp).

```
       reaction: APCi + IEp -> APCa + IEp
  reaction rate: (kap*[IEp]*[APCi])/(Kmap + [APCi])
     parameters: kap = 0.13 1/minute
                 Kmap = 0.01 nM
       species : APCi = 1 nM
                 APCa = 0 nM
```

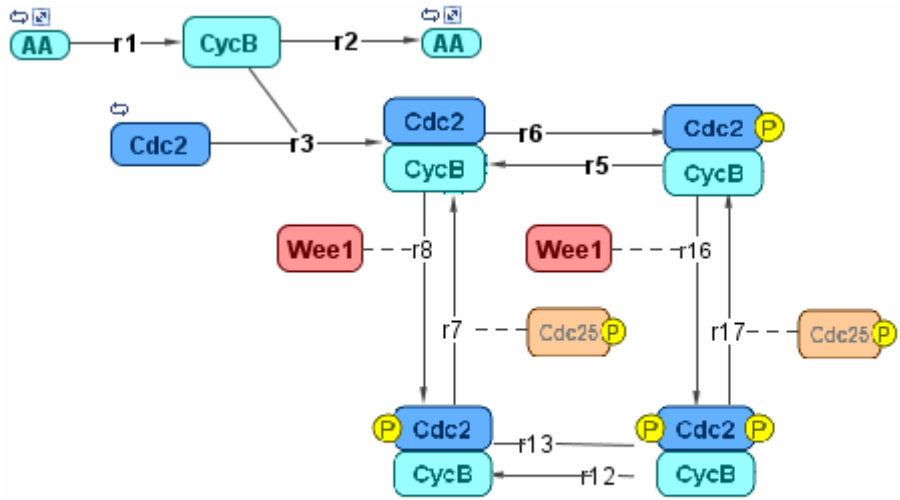
Reaction 43, APC Deactivation

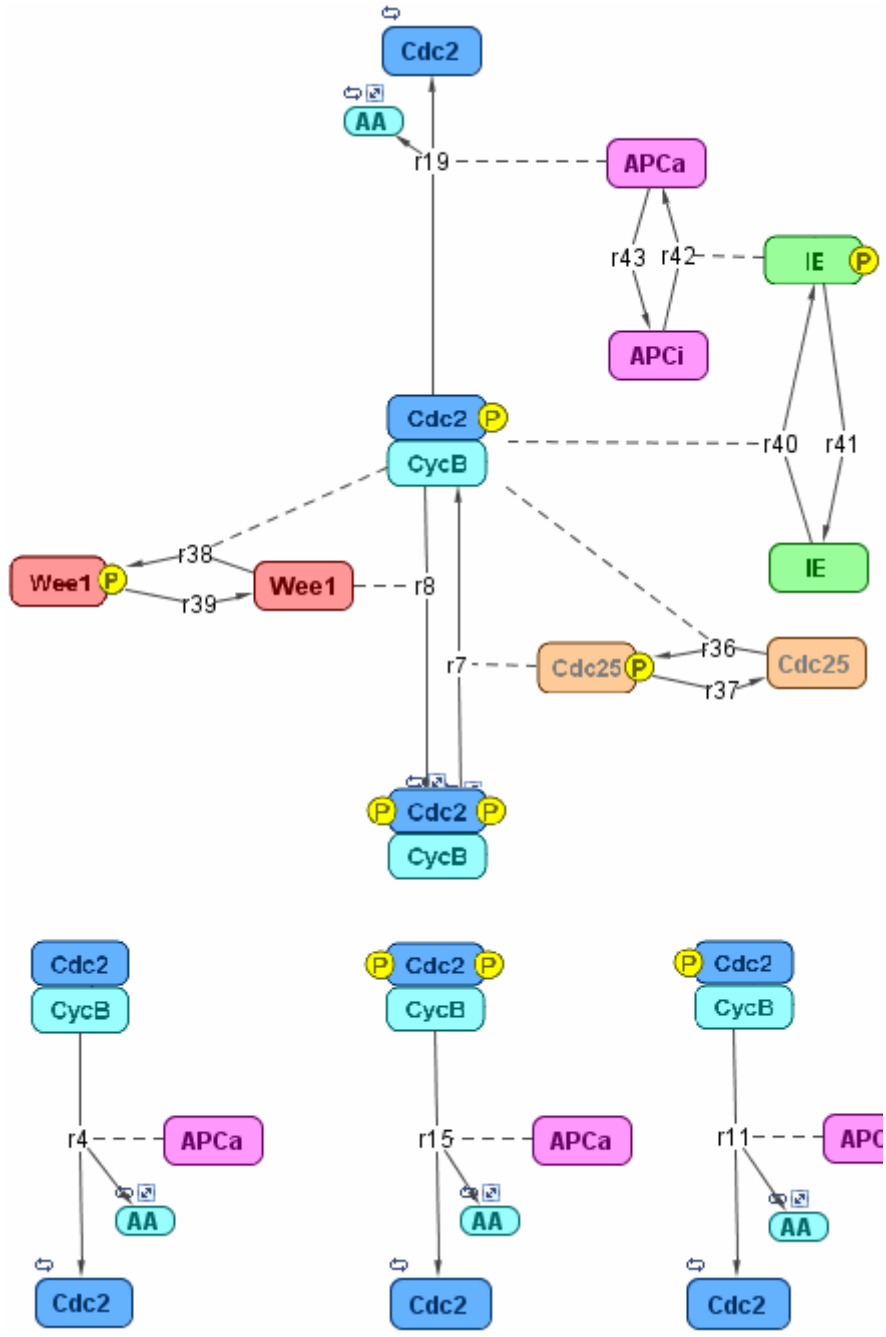
Anaphase-promoting complex (APC) is deactivated.

```
       reaction: APCa -> APCi
```

```
reaction rate: (kapr*[APCa])/(Kmapr + [APCa])
   parameters: kapr = 0.13 nM/minute
               Kmapr = 1 nM
     species : APCi = 1 nM
               APCa = 0 nM
```

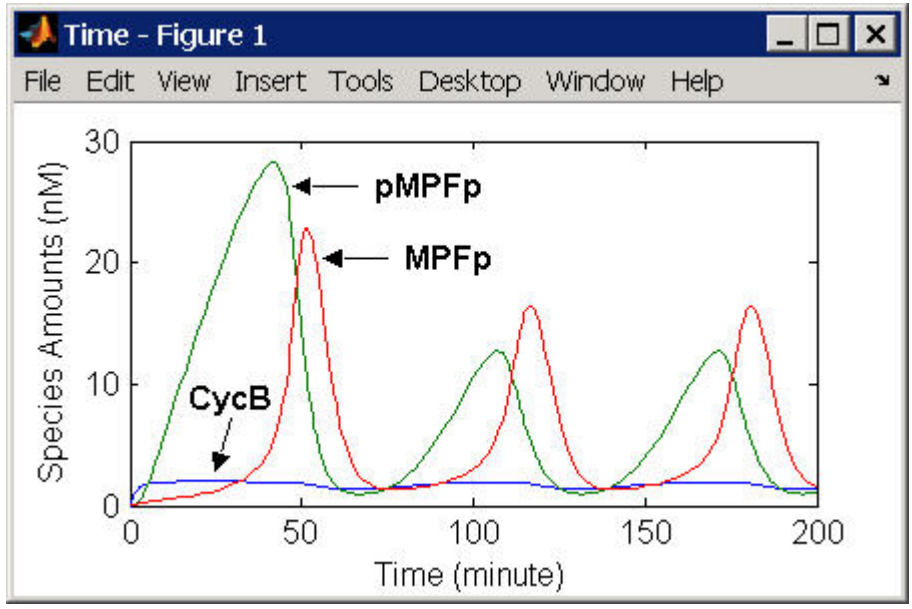
Block Diagram of the M-Phase Control Model with Reactions




Cdc2
AA
r19
APCa
r43
r42
IE
P
APCi
Cdc2
P
CycB
r40
r41
r38
Wee1
P
r39
Wee1
r8
IE
r36
r7
Cdc25
P
Cdc25
r37
P
Cdc2
P
CycB
Cdc2
CycB
r4
APCa
AA
Cdc2
P
Cdc2
P
CycB
r15
APCa
AA
Cdc2
P
Cdc2
P
CycB
r11
APC
AA
Cdc2

Simulation of the M-Phase Control Model with Reactions

This is a simulation of the M-phase control model using reactions and algebraic rules. Simulate with the `ode15s` solver and plot species `CycB`, `pMPFp`, and `MPFp`. For a description of the model, see "SimBiology Model with Reactions and Algebraic Rules" on page 3-21.



If you want to run the simulation, you can open the model within the SimBiology desktop.

1. Open the desktop using the function `sbiodesktop`.
2. From the **File** menu, select **Open Project**.
3. Browse to the file `m_phase_xenopus.sbproj` in the following directory:

   matlabroot`/toolbox/simbio/simbiodemos/m_phase_xenopus.sbproj`

For a model using reactions and algebraic rules, see "SimBiology Model with Reactions and Algebraic Rules" on page 3-21.

References

[1] Borisuk M, Tyson J (1998), "Bifurcation analysis of a model of mitotic control in frog eggs," Journal of Theoretical Biology, 195(1):69–85, PubMed 9802951.

[2] Marlovits G, Tyson C, Novak B, Tyson J (1998), "Modeling M-phase control in Xenopus oocyte extracts: the surveillance mechanism for unreplicated DNA," Biophysical Chemistry, 72(1-2):169–184, PubMed 9652093.

[3] Novák B, Tyson J (1993), "Numerical analysis of a comprehensive model of M-phase control in Xenopus oocyte extracts and intact embryos," Journal of Cell Science, 106(4):1153–1168, PubMed 8126097.

Index