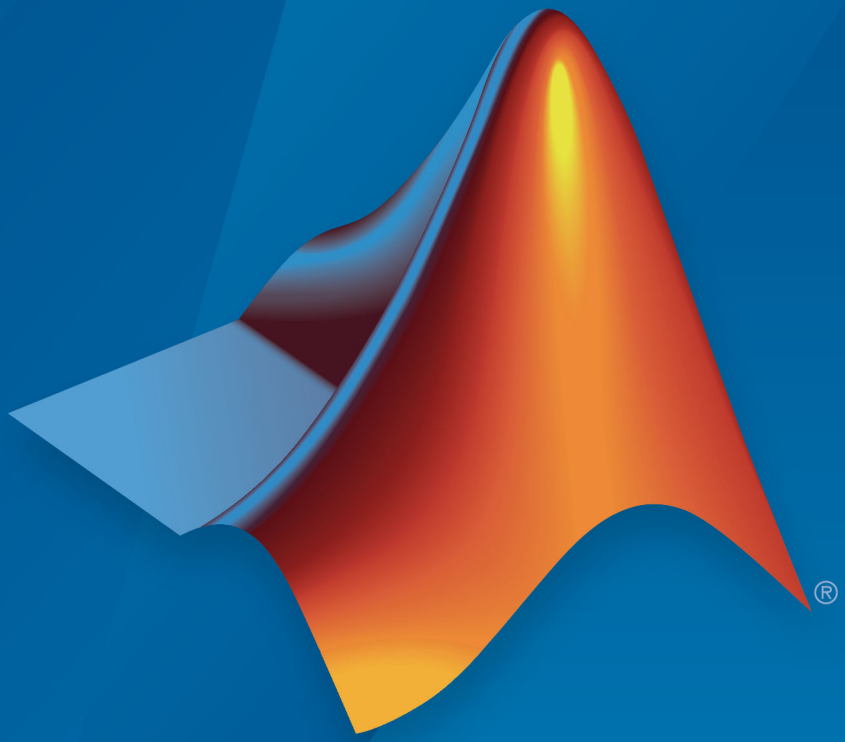
SimBiology®

User's Guide



MATLAB®

R2018a

MathWorks®

How to Contact MathWorks

Latest news: www.mathworks.com

Sales and services: www.mathworks.com/sales_and_services

User community: www.mathworks.com/matlabcentral

Technical support: www.mathworks.com/support/contact_us

Phone: 508-647-7000

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

SimBiology® User's Guide

© COPYRIGHT 2005–2018 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

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

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)
March 2009	Online only	Updated for Version 3.0 (Release 2009a)
September 2009	Online only	Updated for Version 3.1 (Release 2009b)
March 2010	Online only	Updated for Version 3.2 (Release 2010a)
September 2010	Online only	Updated for Version 3.3 (Release 2010b)
April 2011	Online only	Updated for Version 3.4 (Release 2011a)
September 2011	Online only	Updated for Version 4.0 (Release 2011b)
March 2012	Online only	Updated for Version 4.1 (Release 2012a)
September 2012	Online only	Updated for Version 4.2 (Release 2012b)
March 2013	Online only	Updated for Version 4.3 (Release 2013a)
September 2013	Online only	Updated for Version 4.3.1 (Release 2013b)
March 2014	Online only	Updated for Version 5.0 (Release 2014a)
October 2014	Online only	Updated for Version 5.1 (Release 2014b)
March 2015	Online only	Updated for Version 5.2 (Release 2015a)
September 2015	Online only	Updated for Version 5.3 (Release 2015b)
March 2016	Online only	Updated for Version 5.4 (Release 2016a)
September 2016	Online only	Updated for Version 5.5 (Release 2016b)
March 2017	Online only	Updated for Version 5.6 (Release 2017a)
September 2017	Online only	Updated for Version 5.7 (Release 2017b)
March 2018	Online only	Updated for Version 5.8 (Release 2018a)

Contents

1 SimBiology Desktop

General Workflow **1-2**

SimBiology Desktop Navigation **1-5**

SimBiology Desktop Help and Tools **1-10**
- Contextual Help **1-10**
- Message Indicators **1-11**
- Contextual Icons **1-15**
- Graphical Context Menus **1-17**
- Tools **1-18**

Keyboard Shortcuts for SimBiology Desktop **1-20**
- Shortcuts for the Diagram View **1-20**

Modeling Workflow **1-22**

Model Views **1-25**
- Diagram View **1-25**
- Table View **1-47**
- Equations View **1-49**

Setting Preferences **1-54**

Libraries **1-56**
- Kinetic Laws Library **1-57**
- Units Library **1-58**
- Unit Prefixes Library **1-58**
- Blocks Library **1-59**

Analysis Workflow **1-60**

Task Editor **1-63**

Built-in Tasks **1-66**

Configuring Tasks **1-70**
Configuring Model-Related Settings **1-70**
Configuring Simulation-Related Settings **1-71**
Configuring Task-Specific Settings **1-71**

Running Tasks **1-76**
Configuring Live Plots **1-76**
Exploring Models **1-79**

External Data and Task Results **1-83**
External Data **1-84**
Task Results **1-86**

2 Modeling

What is a Model? **2-2**
Model Definition **2-2**
Expressions **2-2**
Quantities **2-3**
Model Hierarchy **2-4**
Representing a Model **2-4**

Species Object **2-6**
How Species Amounts Change During Simulations **2-6**
Keeping a Species Amount Unchanged **2-6**
Changing a Species Amount with a Reaction or Rule **2-7**
Changing a Species Amount with a Rule When Species is Part of a Reaction **2-7**
Keeping a Species Amount Unchanged When Species is Part of a Reaction that Adds or Removes Mass **2-8**

Definitions and Evaluations of Reactions **2-10**
Writing Reaction Expressions **2-10**
Writing Reaction Rate Expressions Explicitly **2-11**
Creating Reaction Rate Expressions Using Kinetic Law Objects **2-12**
Examples of Creating Reaction Rates **2-13**

How Reaction Rates Are Evaluated 2-14
Viewing Equations for Reactions 2-16

Definitions and Evaluations of Rules 2-17
Overview 2-17
Initial Assignment 2-17
Repeated Assignment 2-18
Algebraic Rules 2-18
Repeated Assignment vs. Algebraic Rules 2-19
Rate Rules 2-19
Evaluation Order of Rules 2-19
Conservation of Amounts During Simulation 2-20
Writing Rule Expressions 2-21
Considerations When Imposing Constraints 2-21
Rate Rule Examples 2-22

Events 2-30
Overview 2-30
Event Triggers 2-30
Event Functions 2-31
Specifying Event Triggers 2-31
Specifying Event Functions 2-33
Simulation Solvers for Models Containing Events 2-34
How Events Are Evaluated 2-34
Evaluation of Simultaneous Events 2-36
Evaluation of Multiple Event Functions 2-37
When One Event Triggers Another Event 2-37
Cyclical Events 2-38
Using Events to Address Discontinuities in Rule and Reaction Rate Expressions 2-38

Variants 2-40
Creating Variants Programmatically 2-40
Creating Variants Graphically 2-40

Doses 2-42
Representing Doses 2-42
Creating Doses Programmatically 2-43
Creating Doses Graphically 2-44
Parameterized and Adaptive Doses 2-44
Simulation Solvers for Models Containing Doses 2-46

Simulate Biological Variability of the Yeast G Protein Cycle Using the Wild-Type and Mutant Strains **2-47**

Create and Simulate a Model with a Custom Function **2-49**
Overview **2-49**
Create a Custom Function **2-51**
Load the Example Model **2-51**
Add the Custom Function to the Example Model **2-52**
Define a Rule to Change Parameter Value **2-52**
Add an Event to Reset the Solver at a Discontinuity **2-52**
Simulate the Modified Model **2-53**

View Model Equations **2-57**

Component Usage **2-58**
Species Usage **2-58**
Parameter Usage **2-58**
Compartment Usage **2-59**
Unit and UnitPrefix Usage **2-59**
Abstract Kinetic Law Usage **2-59**

Evaluation of Model Component Names in Expressions **2-61**
Guidelines for Naming Model Components **2-61**
Guidelines for Referencing Names in Expressions **2-61**
Precedence Rules for Evaluating Quantity Names **2-61**

3 Structural Analysis

Model Verification **3-2**
What is Model Verification? **3-2**
When to Verify a Model **3-2**
Verifying That a Model Has No Warnings or Errors **3-3**
Model Verification Example **3-3**

Conserved Moiety Determination **3-4**
Introduction to Moiety Conservation **3-4**
Algorithms for Conserved Cycle Calculations **3-4**
More About **3-6**

Determining Conserved Moieties **3-7**

Determining the Adjacency Matrix for a Model **3-10**
What Is an Adjacency Matrix? **3-10**
Retrieving an Adjacency Matrix for a Model **3-10**

Determining the Stoichiometry Matrix for a Model **3-12**
What Is a Stoichiometry Matrix? **3-12**
Retrieving a Stoichiometry Matrix for a Model **3-13**

Selecting Absolute Tolerance and Relative Tolerance for Simulation **3-15**
Algorithm **3-15**
Absolute Tolerance Scaling **3-16**

Troubleshooting Simulation Problems **3-18**
Tips for Solving Simulation Problems **3-18**
How to Change Solver Options and Simulation Options **3-20**

Simulate Model of Glucose-Insulin Response with Different Initial Conditions **3-22**

4 Simulation and Analysis

Model Simulation **4-2**

Deriving ODEs from Reactions **4-4**

Choosing a Simulation Solver **4-7**

SUNDIALS Solvers **4-9**

Stochastic Solvers **4-10**
When to Use Stochastic Solvers **4-10**
Model Prerequisites for Simulating with a Stochastic Solver **4-10**
What Happens During a Stochastic Simulation? **4-11**
Stochastic Simulation Algorithm (SSA) **4-11**
Explicit Tau-Leaping Algorithm **4-11**

Implicit Tau-Leaping Algorithm 4-12
References 4-13

Ensemble Runs of Stochastic Simulations 4-15
Running Ensemble Simulations 4-15

Configuring Simulation Settings 4-16

Create and Simulate a Simple Model 4-17

Simulate the Yeast Heterotrimeric G Protein Cycle 4-22

Sensitivity Calculation 4-27
About Calculating Sensitivities 4-27
Model Requirements for Calculating Sensitivities 4-27
Calculate Sensitivities using sbiosimulate or SimFunctionSensitivity Object 4-28
References 4-30

Calculate Sensitivities 4-31
Overview 4-31
Load and Configure the Model for Sensitivity Analysis 4-32
Perform Sensitivity Analysis 4-33
Extract and Plot Sensitivity Data 4-33

Identify Important Network Components from an Apoptosis Model Using Sensitivity Analysis 4-35

Perform a Parameter Scan 4-40

Nonlinear Mixed-Effects Modeling 4-42
What Is a Nonlinear Mixed-Effects Model? 4-42
Nonlinear Mixed-Effects Modeling Workflow 4-44
Specify a Covariate Model 4-45
Specify an Error Model 4-47
Maximum Likelihood Estimation 4-47
Obtain the Fitting Status 4-48

Nonlinear Regression 4-50
What is Nonlinear Regression? 4-50
Fitting Options in SimBiology 4-51
Maximum Likelihood Estimation 4-53
Fitting Workflow for sbiofit 4-55

Supported Methods for Parameter Estimation **4-57**

Error Models **4-60**

Progress Plot **4-61**
Progress Plot for Nonlinear Mixed-Effects Methods **4-61**
Progress Plot for Nonlinear Regression Methods **4-63**

Fit a One-Compartment Model to an Individual's PK Profil e **4-70**

Estimate Category-Specific PK Parameters for Multiple Individuals **4-78**

Perform Hybrid Optimization Using sbiofit **4-91**

Fit a Two-Compartment Model to PK Profiles of Multiple Individuals **4-96**

Accelerating Model Simulations and Analyses **4-106**
What Is Acceleration? **4-106**
When to Accelerate **4-106**
Prerequisites for Accelerating Simulations and Analyses . . . **4-106**
Accelerate Simulations Programmatically **4-107**
Accelerate Simulations using SimBiology Desktop **4-109**
Troubleshooting Accelerated Simulations **4-109**

Noncompartmental Analysis **4-111**
Data **4-111**
Dosing **4-111**
Calculating NCA Parameters **4-118**

5 Pharmacokinetic Modeling

Pharmacokinetic Modeling Functionality **5-2**
Overview **5-2**
How SimBiology Supports Pharmacokinetic Modeling **5-2**
Pharmacokinetic Modeling Examples **5-4**
Acknowledgements: Tobramycin Data Set **5-4**

Importing Data — Supported Files and Data Types **5-6**
Supported Files and Data Types **5-6**
Support for Importing NONMEM Formatted Files **5-6**
Creating a Data File with SimBiology Definitions **5-11**

Importing Data **5-12**
Import Data from Files **5-12**
Importing Data from NONMEM-Formatted Files **5-13**
Other Resources for Importing Data **5-14**

Import Data from a NONMEM-Formatted File Using the SimBiology Desktop **5-15**

Create Pharmacokinetic Models **5-24**
Ways to Create or Import Pharmacokinetic Model **5-24**
How SimBiology Models Represent Pharmacokinetic Models **5-24**
Create a Pharmacokinetic Model Using the Command Line . . **5-26**
Dosing Types **5-28**
Elimination Types **5-31**
Intercompartmental Clearance **5-33**
Unit Conversion for Imported Data **5-34**

Perform Data Fitting with PKPD Models **5-36**

A Creating Reaction Rates

Define Reaction Rates with Mass Action Kinetics **A-2**
Definition of Mass Action Kinetics **A-2**
Zero-Order Reactions **A-2**
First-Order Reactions **A-3**
Second-Order Reactions **A-4**
Reversible Mass Action **A-6**

Define Reaction Rates with Enzyme Kinetics **A-8**
Simple Model for Single Substrate Catalyzed Reactions **A-8**
Enzyme Reactions with Differential Rate Equations **A-8**
Enzyme Reactions with Mass Action Kinetics **A-10**

Enzyme Reactions with Irreversible Henri-Michaelis-Menten Kinetics A-11

B Models Used in Examples

Minimal Cascade Model for a Mitotic Oscillator **B-2**
Goldbeter Model **B-2**
SimBiology Model with Rate Rules **B-5**
SimBiology Model with Reactions **B-7**
References **B-17**

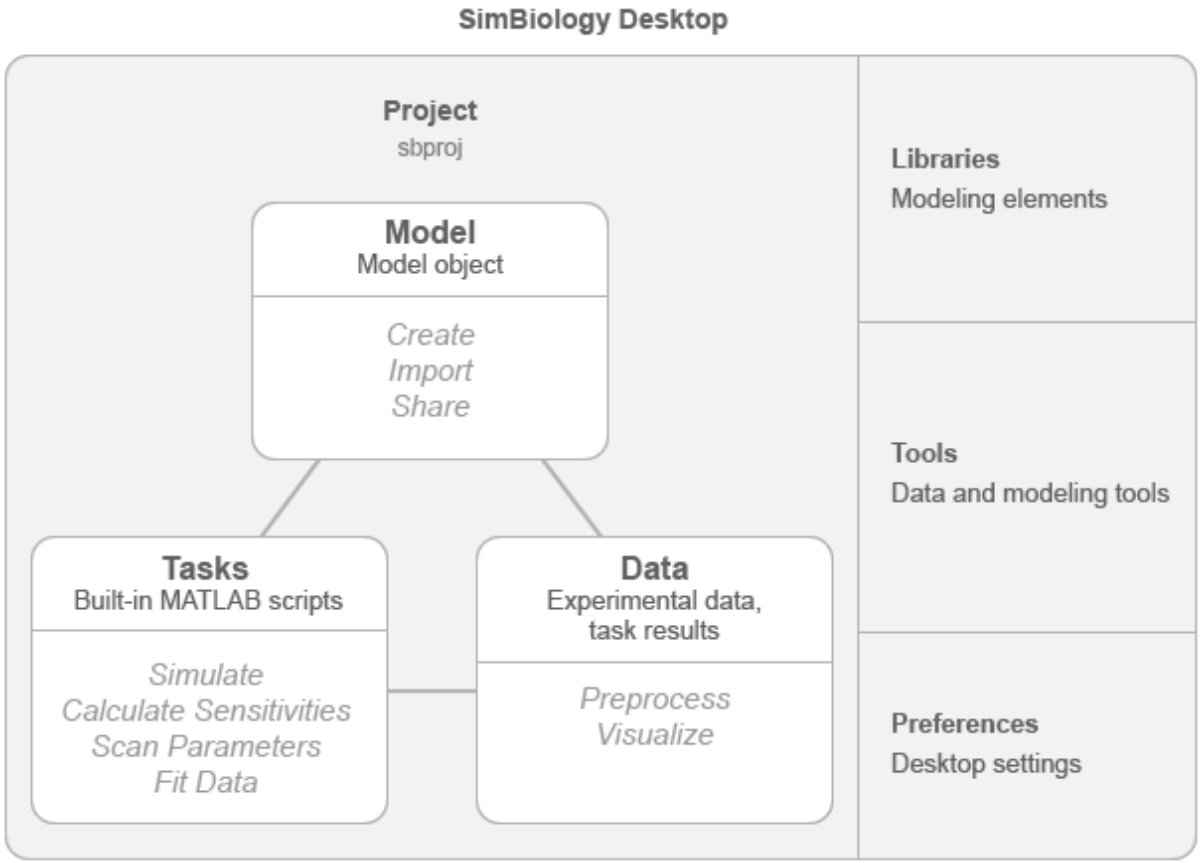
Model of the Yeast Heterotrimeric G Protein Cycle **B-18**
Background on G Protein Cycles **B-18**
Modeling a G Protein Cycle **B-19**
References **B-22**

Model of M-Phase Control in Xenopus Oocyte Extracts **B-23**
M-Phase Control Model **B-23**
M-Phase Control Equations **B-25**
SimBiology Model with Rate and Algebraic Rules **B-33**
SimBiology Model with Reactions and Algebraic Rules **B-39**
References **B-56**

SimBiology Desktop

- “General Workflow” on page 1-2
- “SimBiology Desktop Navigation” on page 1-5
- “SimBiology Desktop Help and Tools” on page 1-10
- “Keyboard Shortcuts for SimBiology Desktop” on page 1-20
- “Modeling Workflow” on page 1-22
- “Model Views” on page 1-25
- “Setting Preferences” on page 1-54
- “Libraries” on page 1-56
- “Analysis Workflow” on page 1-60
- “Task Editor” on page 1-63
- “Built-in Tasks” on page 1-66
- “Configuring Tasks” on page 1-70
- “Running Tasks” on page 1-76
- “External Data and Task Results” on page 1-83

General Workflow



The SimBiology desktop is a user interface with a set of integrated tools that are designed to facilitate building, simulating, and analyzing models of dynamic systems. It provides an interactive interface to build models and perform model analyses such as simulation, sensitivity calculation, and parameter estimation. It lets you save your modeling work as a SimBiology project that contains models, experimental data, model analysis tasks, and task results.

A SimBiology model is a set of quantities and mathematical expressions that represent a dynamic system. Expressions describe the mathematical relationships among quantities. For details about SimBiology models, see “What is a Model?” on page 2-2.

The desktop lets you build such a dynamic model interactively. One approach is connecting graphical blocks that represent modeling elements such as quantities and expressions. Alternatively, you can build a model by entering expressions as strings such as `A –> B`, where `A` and `B` are quantities participating in a transformation process.

Once you have a model, you can perform analyses on it. For instance, you can simulate the model to see its dynamic behavior over a time course or explore biological variability by simulating alternate scenarios. You can also perform parameter scans and sensitivity analysis to investigate the influence of model parameters and initial conditions on model behavior. To help you perform these analyses, the desktop provides built-in scripts with a user interface called tasks.

You can import time-course data to the desktop. It provides tools for filtering and visualizing the data. Once imported, you can use the data to estimate model parameters or compare to simulation results.

As you build and analyze models, you can use built-in components from libraries. For instance, the units library provides the unit functionality and has a collection of predefined units that you can use in your model. To help you build models graphically, the desktop provides the blocks library with blocks representing different modeling elements. Custom elements can be added to any of these libraries. For details, see “Libraries” on page 1-56.

As you work on models and analyses, the desktop continuously checks if there are any errors or warnings and indicates them using message indicators on page 1-11. You can hover over each indicator to see the corresponding warning or error message.

Preferences on page 1-54 let you customize model building and display settings, task settings, and search options.

See Also

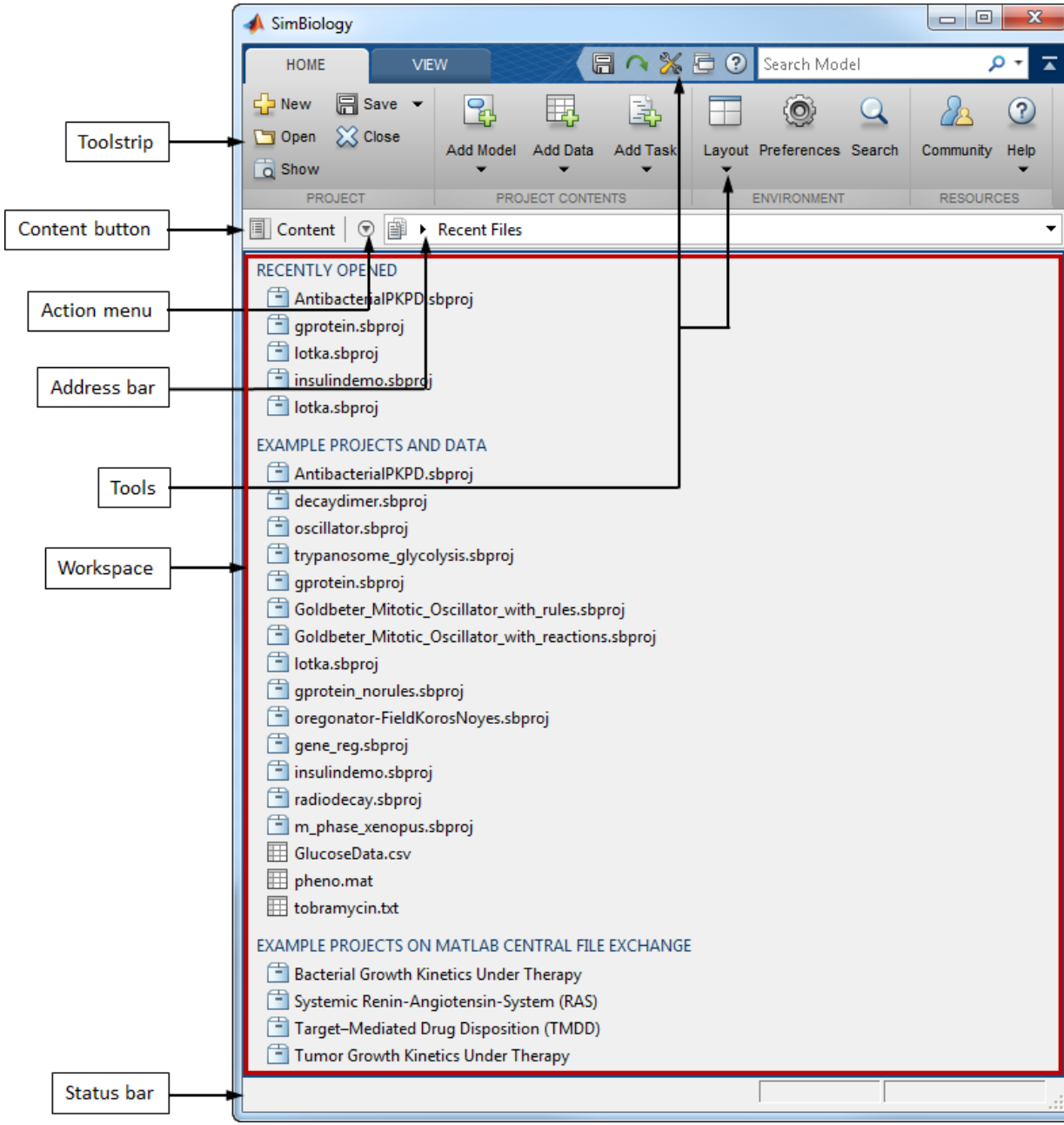
More About

- “SimBiology Desktop Navigation” on page 1-5

- “SimBiology Desktop Help and Tools” on page 1-10
- “Setting Preferences” on page 1-54
- “Libraries” on page 1-56

SimBiology Desktop Navigation

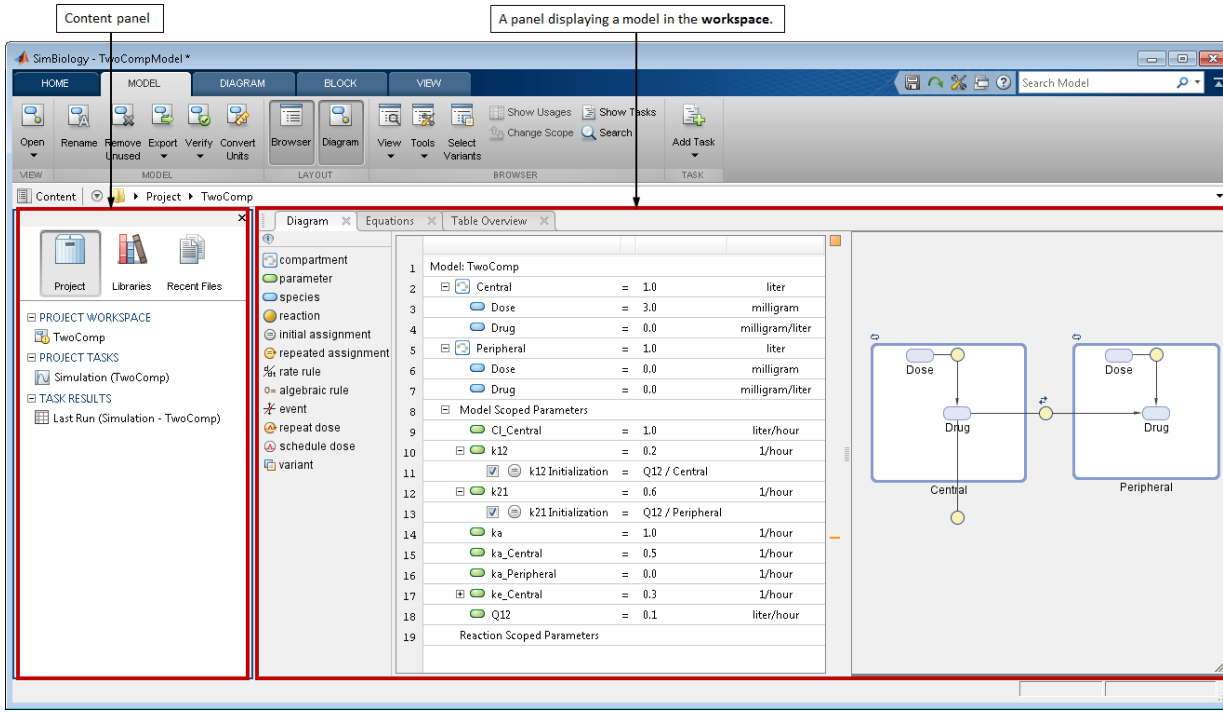
To open the SimBiology desktop, enter `simbiology` at the MATLAB® command prompt or select **SimBiology** on the **Apps** tab.


Toolstrip
Content button
Action menu
Address bar
Tools
Workspace
Status bar
SimBiology
HOME
VIEW
Search Model
New
Save
Open
Close
Show
Add Model
Add Data
Add Task
Layout
Preferences
Search
Community
Help
PROJECT
PROJECT CONTENTS
ENVIRONMENT
RESOURCES
Content
Recent Files
RECENTLY OPENED
AntibacterialPKPD.sbproj
gprotein.sbproj
lotka.sbproj
insulindemo.sbproj
lotka.sbproj
EXAMPLE PROJECTS AND DATA
AntibacterialPKPD.sbproj
decaydimer.sbproj
oscillator.sbproj
trypanosome_glycolysis.sbproj
gprotein.sbproj
Goldbeter_Mitotic_Oscillator_with_rules.sbproj
Goldbeter_Mitotic_Oscillator_with_reactions.sbproj
lotka.sbproj
gprotein_norules.sbproj
oregonator-FieldKorosNoyes.sbproj
gene_reg.sbproj
insulindemo.sbproj
radiodecay.sbproj
m_phase_xenopus.sbproj
GlucoseData.csv
pheno.mat
tobramycin.txt
EXAMPLE PROJECTS ON MATLAB CENTRAL FILE EXCHANGE
Bacterial Growth Kinetics Under Therapy
Systemic Renin-Angiotensin-System (RAS)
Target–Mediated Drug Disposition (TMDD)
Tumor Growth Kinetics Under Therapy

The desktop provides options to help you build models, load data, explore data with plots, and add tasks for model analyses. You can find these options in the toolstrip, content panel, action menu, and address bar.

The toolstrip has the **Home** and **View** tabs. Depending on what is open, other context-sensitive tabs appear. The content panel shows the contents of a project, available libraries, and files that are recently open. Click the **Content** button to open the content panel. The action menu contains additional functionality related to an open panel. For instance, after opening a model, you can rename it or export it as an SBML file using the corresponding options from the action menu. The address bar shows which panel is open. You can use it to see the contents of a project and open them. Click an arrow in the address to open other panels.

When you open data, a model, or a library, the desktop shows the related items in a panel displayed in the workspace. For instance, the next figure shows a model open in the Diagram view on page 1-25. Its graphical representation is displayed in a panel as shown. If there are multiple panels such as other views of the model, each panel opens in its own tab.



In the content panel, **Project** contains a list of models, tasks, task results, and imported data. To open an item in a project, double-click it. Models, task results, or data are displayed in the workspace when opened. Opening a task displays it in the task editor on page 1-63. **Libraries** on page 1-56 lists available units, unit prefixes, and blocks that you can use while building a model.

Recent Files has a list of files that you recently opened, projects and data from SimBiology examples, and projects from MATLAB Central File Exchange. If you open any model or data file listed in the recent files, the desktop imports the model or data and adds to the currently open project or to a new project.

See Also

More About

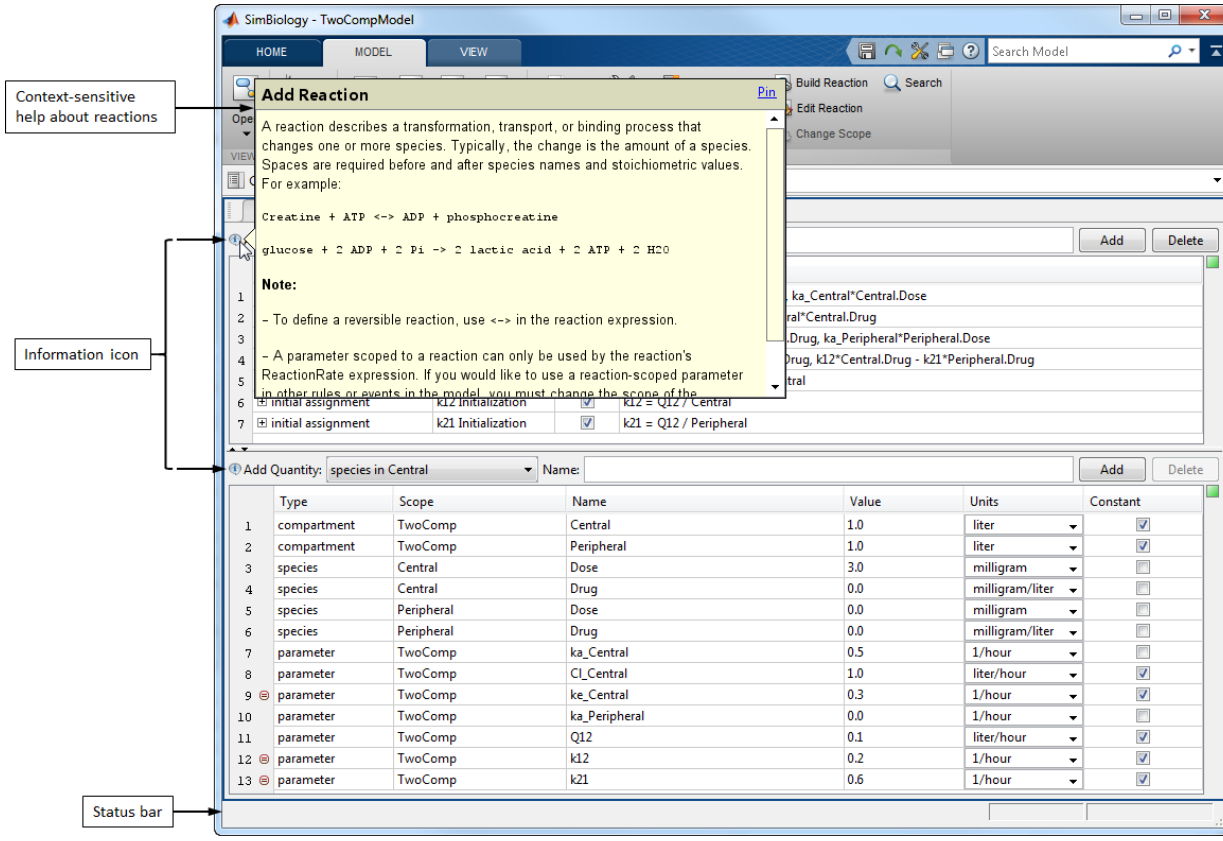
- “General Workflow” on page 1-2
- “SimBiology Desktop Help and Tools” on page 1-10
- “Setting Preferences” on page 1-54

SimBiology Desktop Help and Tools

While building models and configuring tasks, you can obtain information about modeling elements and tasks from context-sensitive help. The desktop identifies errors and warnings of a project using message indicators on page 1-11. In addition, you can check the status bar to find out if the desktop is running a task or performing actions in the background such as loading a project. As you construct models using building blocks on page 1-59, you can use graphical context menus to edit block properties. The desktop also provides various tools on page 1-18 for model building, error checking, and searching.

Contextual Help

The desktop provides context-sensitive help with additional information about models and tasks. For instance, while building models, you can get more information about quantities and expressions and how to use them. Look for the information icon, , and hover over it to see the help. To open the help as a separate dialog, click **Pin** on the top right corner.



Message Indicators

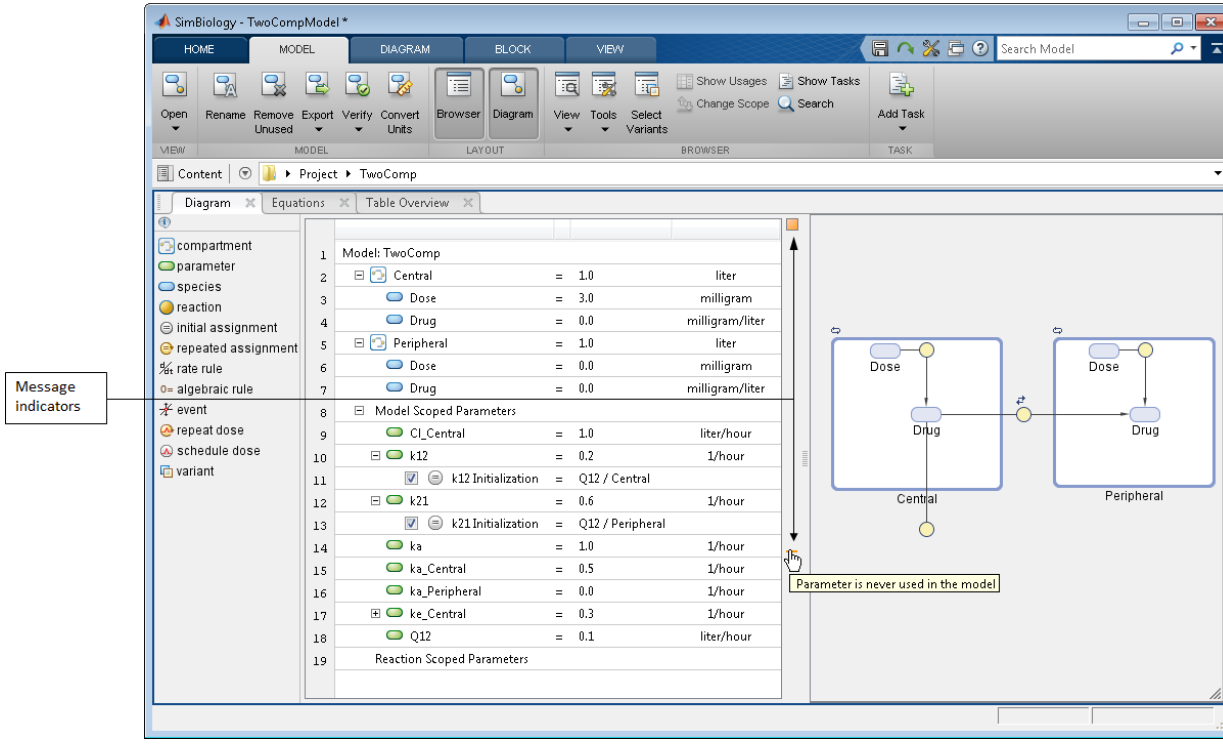
As you build models and configure tasks, the desktop identifies errors and warnings with message indicators. The indicators are color-coded as follows.

- **Red** – one or more errors (and warnings) were detected.
- **Orange** – warnings, but no errors, were detected.
- **Green** – no errors or warnings were detected.

Click a square indicator, such as , to iterate through errors or warnings. Hover over a dash indicator, such as , to see the specific error or warning message. Clicking a dash

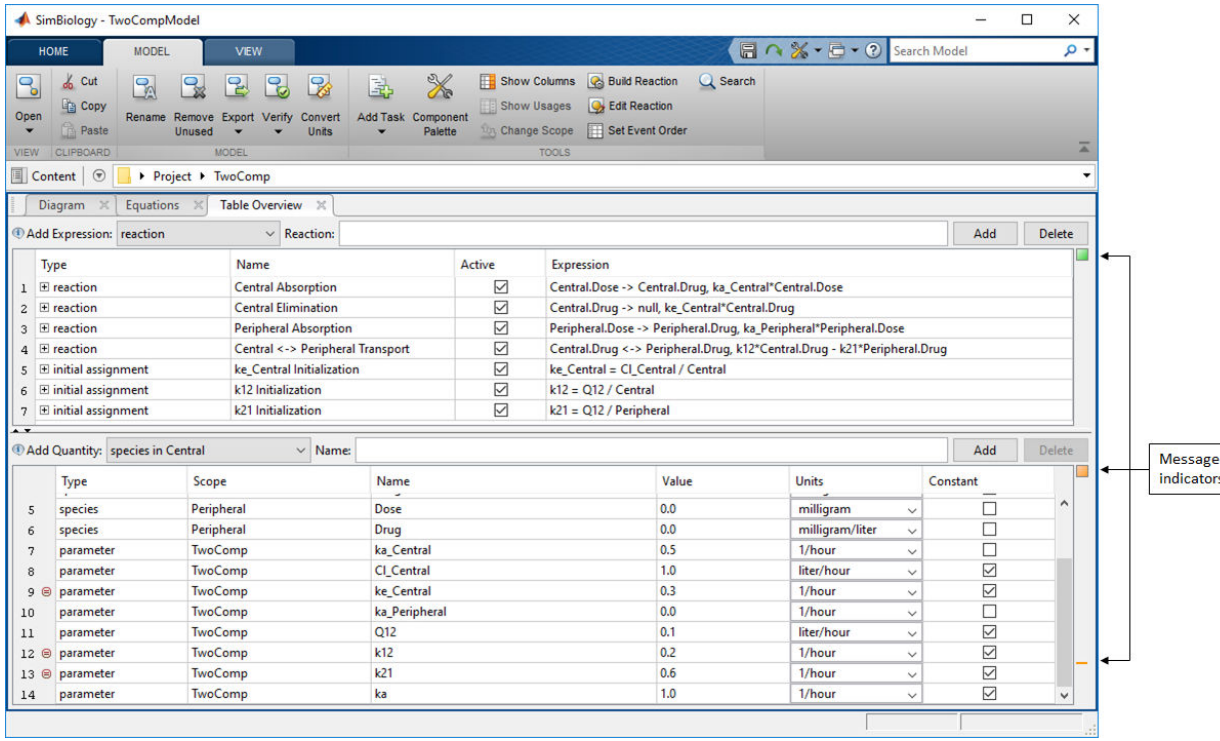
indicator highlights the relevant row in a table. The following figure shows the locations of indicators in the desktop.

Message Indicators in Diagram View



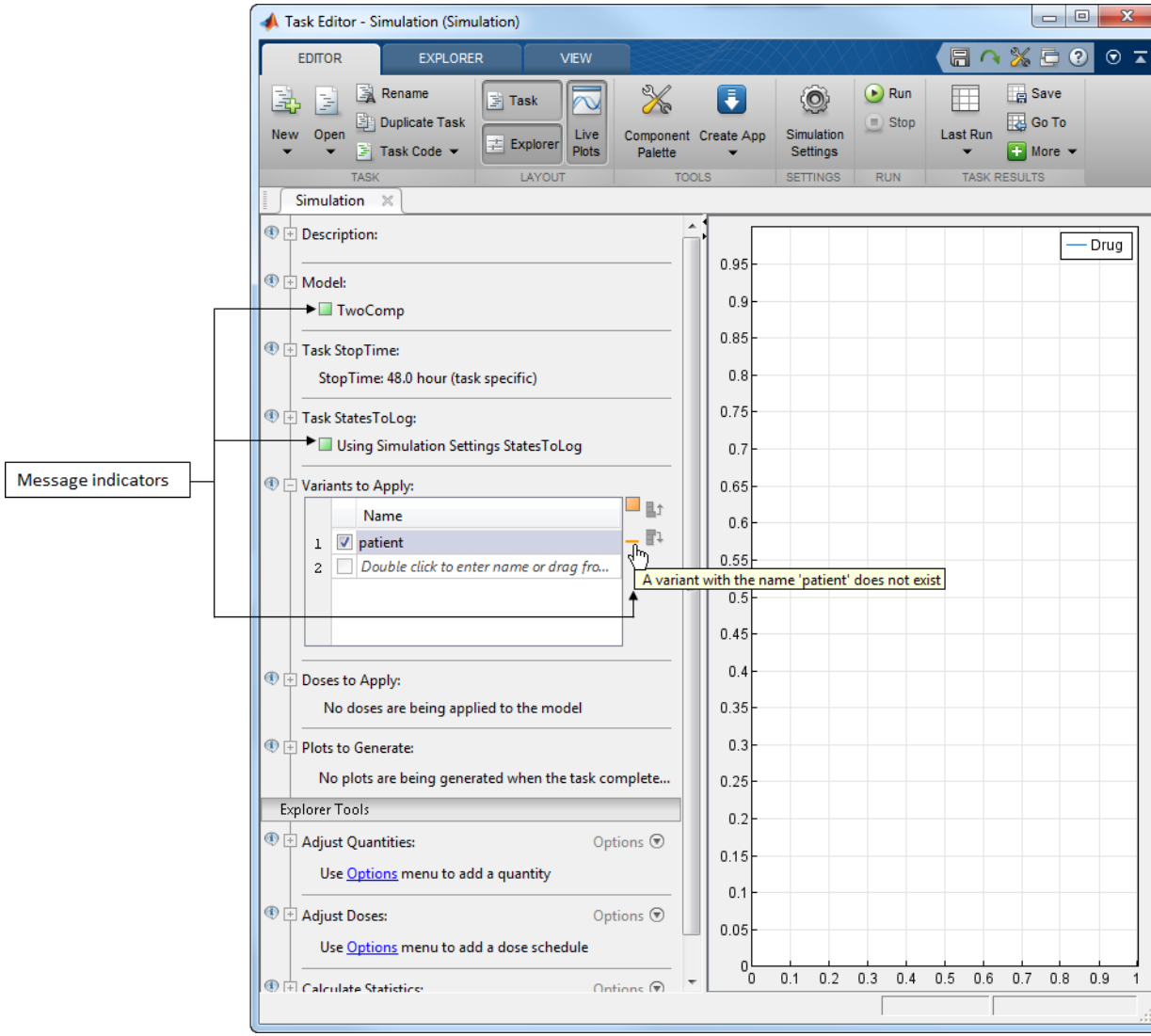
For details about the diagram view and other views of a model, see “Model Views” on page 1-25.

Message Indicators in Table View



For details about the table view and other views of a model, see “Model Views” on page 1-25.

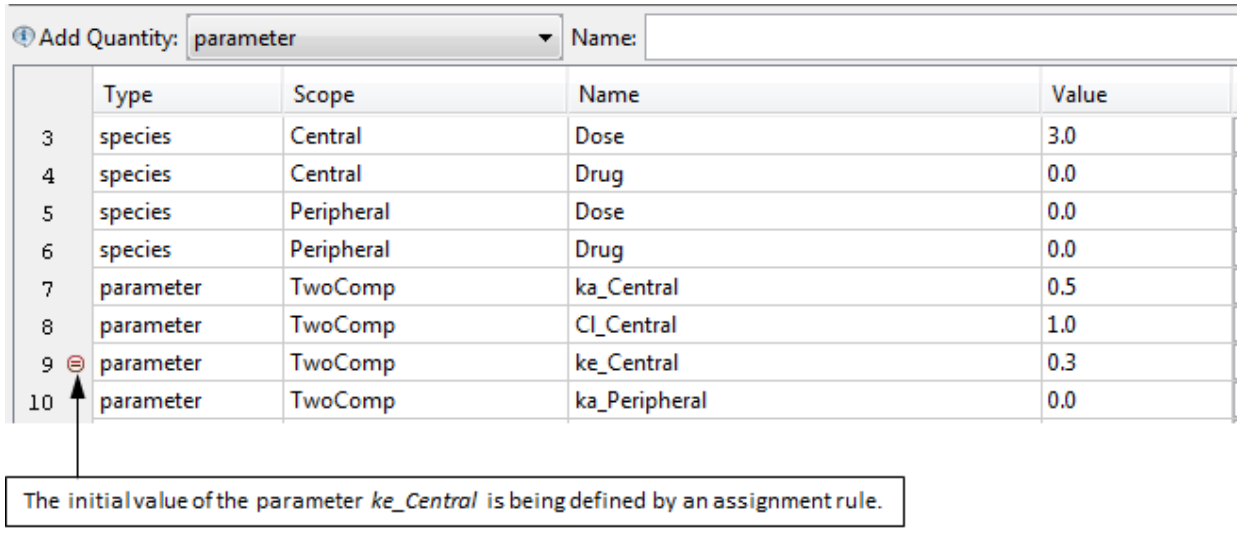
Message Indicators in Task Editor



For details about the task editor, see “Task Editor” on page 1-63.

Contextual Icons

The desktop uses contextual icons to provide more information about quantities and blocks. For instance, in the Diagram view, an icon is displayed above a block if it has an error: . In the Table view, an icon is shown when a species is being dosed or a quantity is being updated by an assignment rule.



For more information, hover the mouse over an icon. The following table has the complete list of contextual icons and corresponding model views where they are displayed.

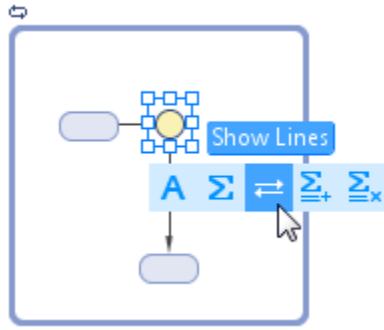
Icon	Description	Diagram View	Table View	Equations View
	Block has an error.	✓	χ	χ

Icon	Description	Diagram View	Table View	Equations View
	Block has a warning or is not being used in a model.	✓	χ	χ
	Block is pinned to its current location in the Diagram view.	✓	χ	χ
	Species amount, parameter value, or compartment volume is set to be constant. In other words, the ConstantAmount, ConstantValue, or ConstantCapacity property of the species, parameter, or compartment block is set to true.	✓	χ	χ
	Species block has the BoundaryCondition property set to true.	✓	χ	χ
	Species or event block is cloned.	✓	χ	χ
	Reaction block has the Reversible property set to true.	✓	χ	χ
	Reaction, rule, or event block has the Active property set to false, meaning it does not participate in the model simulation.	✓	χ	χ

Icon	Description	Diagram View	Table View	Equations View
	Reaction-scoped parameter shadows (i.e., takes precedence over) a model-scoped parameter. The up arrow icon indicates the parameter that shadows. The down arrow icon indicates the parameter that is being shadowed.	✓	✓	✓
	Species amount, parameter value, or compartment volume (capacity) is defined by an assignment rule.	✓	✓	✓
	Species amount is being increased by one or more doses. You must select the desired dose(s) in a task to see the dosing effects.	✓	✓	✓

Graphical Context Menus

While building models using blocks on page 1-43, you can configure block properties using graphical context menus. Select a block and then hover over the graphical icons for the context menu options.



Tools

The desktop provides the following tools to help you build and analyze models.

Tool	Purpose
MATLAB Code Capture Tool	Displays the equivalent MATLAB code of the desktop actions.
Diagram Overview	Shows the graphical representation of an entire model. Use this to pan through a model and zoom in on a particular area.
Component Palette	Lists of modeling elements such as quantities. You can drag and drop these elements to a task when configuring it.
Task Data Comparison	Compare results from fit tasks. For instance, you may want to run a fit task multiple times using different optimization methods and compare the final results using this tool. To compare results, first open the tool. Then run a fit task, and save the task result. Then make changes to the task and rerun it. The tool displays the parameter estimation quality measures, such as log-likelihood, AIC, and BIC, for the last run and each saved result.
Usages	Shows all expressions and tasks that reference a parameter, species, compartment, variant, dose, kinetic law, unit, unit prefix, or plot type. Use the context menu of the corresponding entry in a table to show the usages on page 2-58.

Tool	Purpose
Errors and Warnings	Displays the error and warnings of a model. To populate the list, select **Verify** on the **Model** tab. When you run a task, SimBiology automatically checks if there are any errors or warnings. Double-click an error or warning to go to where it occurs.
Search Results	Shows all instances of word or phrase in a model, kinetic law library, unit and unit prefix library. To search, type in the search box above the toolstrip. **Tip** To search for any keyword or phrase on any open panel, click the **Search** button on the **Home** or **Model** toolstrip. You can define specific search criteria. For instance, you can specifically search for any parameters whose names start with `'k1'`.

Tip You can dock each tool to the desktop or undock as a separate dialog via the action menu button on the top right corner of the tool. When docked, you can drag and reposition the tool anywhere within the desktop.

See Also

More About

- “General Workflow” on page 1-2
- “SimBiology Desktop Navigation” on page 1-5
- “Setting Preferences” on page 1-54
- “Keyboard Shortcuts for SimBiology Desktop” on page 1-20

Keyboard Shortcuts for SimBiology Desktop

On Macintosh platforms, use the **command** key instead of **Ctrl**.

Shortcuts for the Diagram View

Action	Shortcut
Select all blocks	**Ctrl** + **A**
Highlight block in the Table view on page 1-47 or in the Browser on page 1-31 if open	**Ctrl** + **B**
Copy block	**Ctrl** + **C**
Duplicate block	**Ctrl** + **D**
Paste block	**Ctrl** + **V**
Print the diagram on page 1-26	**Ctrl** + **P**
Delete block	**Delete**
Move block up by 5 pixels	Up arrow
Move block down by 5 pixels	Down arrow
Move block left by 5 pixels	Left arrow
Move block right by 5 pixels	Right arrow
Move block by 1 pixel	**Ctrl** + any arrow key
Move block by 10 pixels	**Ctrl** + **Shift** + any arrow key
Pan the diagram	**Shift** + any arrow key
Zoom in	**Ctrl** + **+**
Zoom out	**Ctrl** + **-**
Toggle selection	**Ctrl** + **I**
Select the next cloned block	**N**
Select the previous cloned block	**P**
Move one or more blocks in a clockwise fashion	**Shift** + **U**

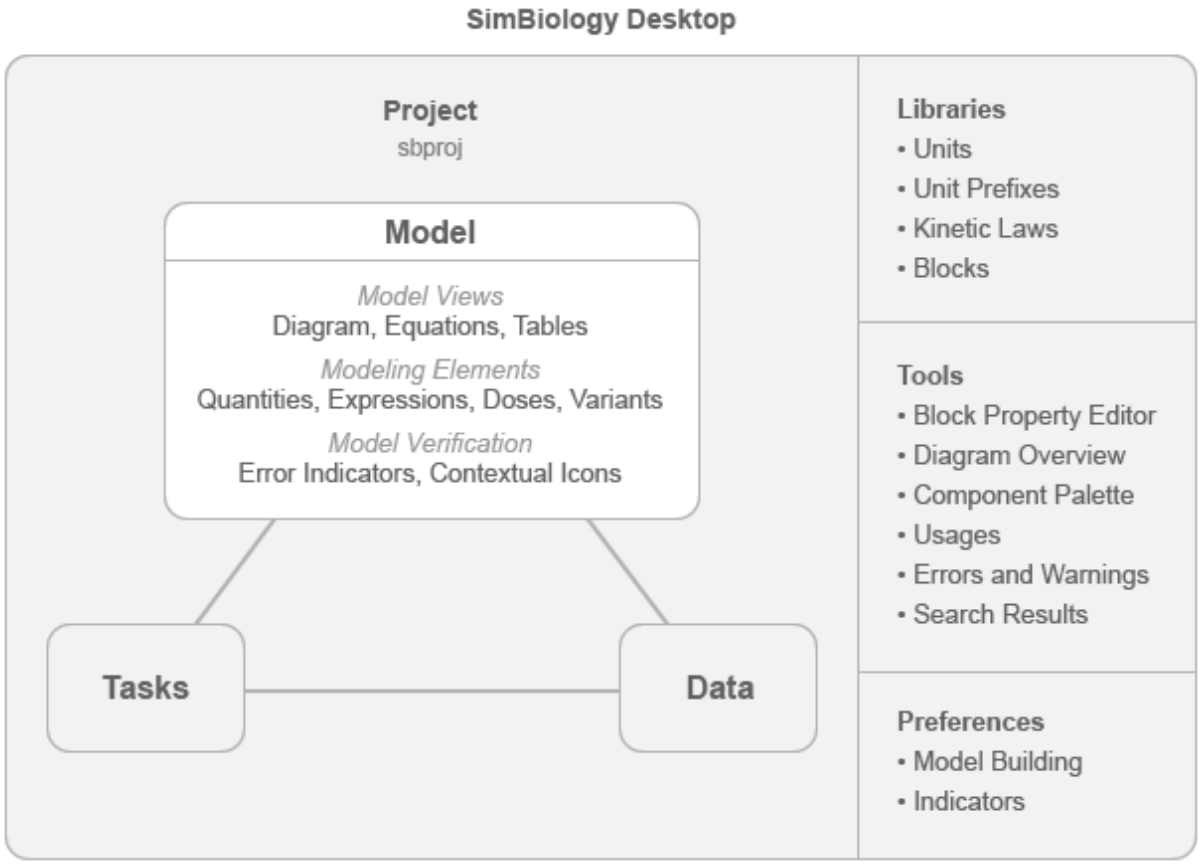
Action	Shortcut
Move one or more blocks in an anti-clockwise fashion	**Ctrl** + **Shift** + **U**
Expand one or more blocks by increasing the distance between them	**Shift** + **E**
Collapse one or more blocks by decreasing the distance between them	**Ctrl** + **E**
Show usages on page 1-18 of a quantity	**Ctrl** + **Shift** + **F**
Show search bar	**Ctrl** + **F**

See Also

More About

- “SimBiology Desktop Help and Tools” on page 1-10
- “Blocks Library” on page 1-59
- “Diagram View” on page 1-25
- “Table View” on page 1-47

Modeling Workflow



A SimBiology model is a dynamic system described by a set of quantities and mathematical expressions. There are three types of quantities: species, parameter, and compartment. Three types of expressions describe the mathematical relationships among quantities. The first type of expression is a reaction which describes a process such as a transformation, transport, binding or unbinding of reactants and products. Another type of expression is a class of assignment equations (rules) which define how quantity values are initialized or updated. The third type is an event that describes a change in a quantity

value during simulation. For details about SimBiology models, see “What is a Model?” on page 2-2.

Biological variability can be modeled using a modeling element called variant. A variant is a collection of quantities with alternate values. Variants do not change the original quantity values permanently. For instance, you can have a set of values for immunological parameters of a healthy person represented by a variant and a different set of values for a cancer patient represented by another variant. You can then simulate the model with each variant to see the predictions for both cases.

An increase in a species amount or concentration due to an external stimulus such as an oral or intravenous administration of a drug can be modeled using an element called dose. You can use an array of doses to explore different dosing regimens and determine the optimal dosing strategy. For details, see “Doses” on page 2-42.

A model can be built and visualized in three different views: the Diagram view, Table view, and Equations view. Each view is a different representation of the same model. For example, the Diagram view shows a graphical representation of the model and lets you build models interactively. For details on all views, see “Model Views” on page 1-25.

As you build a model, the desktop checks if there are any errors or warnings and identifies them using message indicators on page 1-11. You can hover over each indicator to see the corresponding warning or error message. During model verification, the desktop examines many aspects of the model including the model structure and validity of mathematical expressions. It also verifies the consistency of units and dimensions, and flags any issues.

You can save multiple models in one project, which uses a native file format ending in `*.sbproj`. Alternatively, a model can be saved to an SBML–formatted file, but some SimBiology features not supported by SBML are not saved with the model. For details, see “SBML Support”.

Given a model, you can perform several model analyses such as simulation, parameter estimation, and sensitivity calculation. For details, see “Analysis Workflow” on page 1-60.

See Also

More About

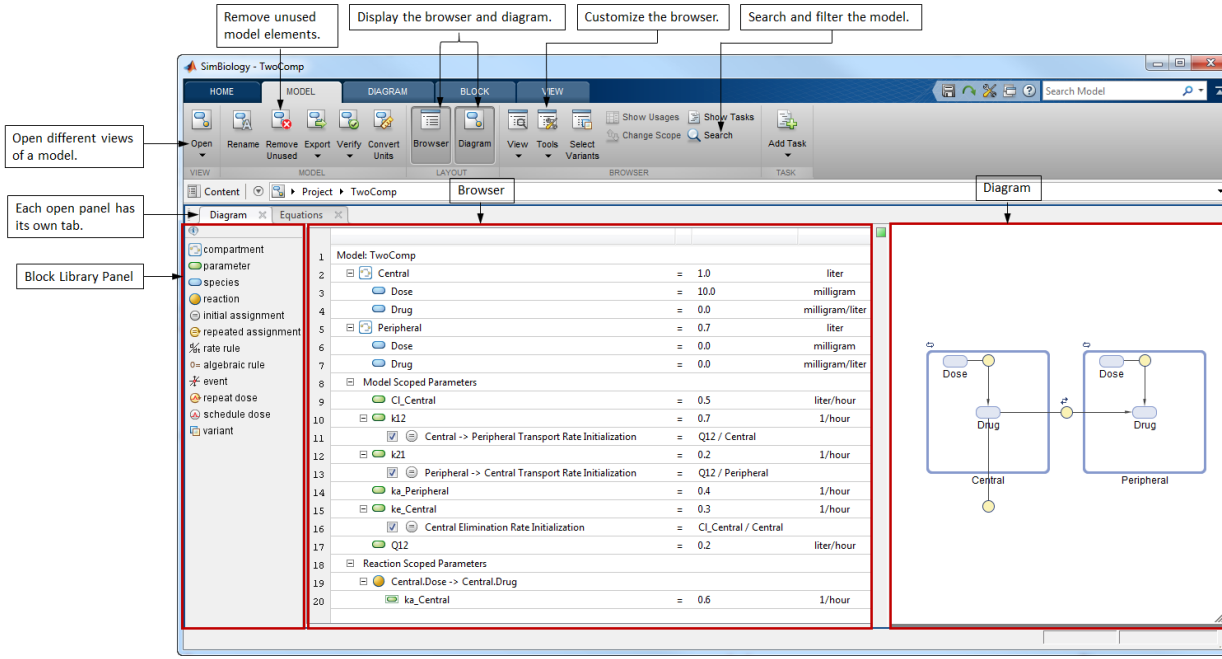
- “Model Views” on page 1-25
- “Model Definition” on page 2-2
- “SimBiology Desktop Help and Tools” on page 1-10
- “What Is SBML?”

Model Views

The SimBiology desktop provides three different views of a model on page 2-2: the Diagram view, Table view, and Equations view. The Diagram view shows a graphical representation of a model. It describes the model using a set of connected blocks. Blocks represent different modeling elements such as quantities and expressions, and the view shows the relationships between model elements graphically. The Table view displays modeling elements and their properties in a tabular form. You can use either view to build a model. The Equations view shows the differential equations and other expressions such as assignment rules of a model. This view does not let you edit the model. If you make a change to the model using the Diagram or Table view, the change is reflected in all views. To open any view, select **Open** from the **Model** tab.

Diagram View

The Diagram view shows the structure of a model graphically. It provides information about how model elements, such as species, interact with one another using a block diagram. To supplement the diagram on page 1-26, this view also contains the browser on page 1-31 that shows the relationships between quantities and expressions. You can build a model by dragging and dropping blocks on page 1-43 from the block library panel to the browser or diagram. The browser and diagram are synchronized, and any update or change is reflected in both. The next figure shows a model open in the Diagram view.



Diagram

The diagram section contains a set of connected blocks that shows the structure of a model and the relationships between model quantities and expressions graphically.

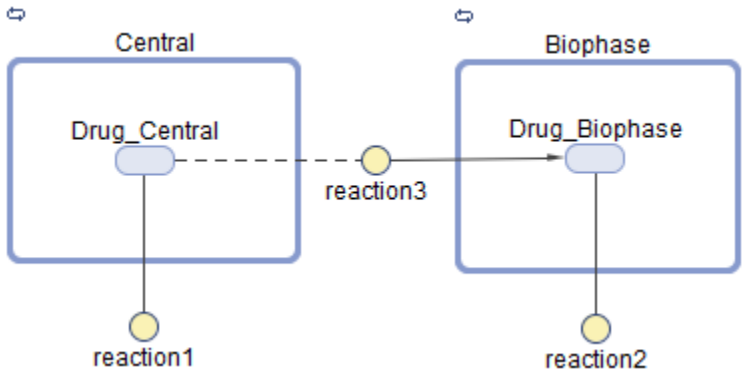
You can add new quantities and expressions by dragging and dropping blocks from the block library panel to the block diagram. To connect two blocks, **Ctrl** + click (**Option** + click for a Mac) the first block and drag to the second block. Double-click a block to edit its properties. Right-click a block to see the context menu with more options, such as hiding the block to avoid clutter. Double-click the name of a block to rename.

The desktop uses contextual icons to provide more information about a block. The icons appear above each block. For instance, if a species is being dosed, a dose icon appears above the species block. Hover the mouse over the icon for more information. For details, see contextual icons on page 1-15.

Adding and configuring reactions

Suppose you want to model the pharmacokinetics of antibacterial drugs in a time-kill curve experiment [1]. Such an experiment involves exposing an in vitro bacterial

inoculum to a fixed antibiotic dose and monitoring bacterial activity over time. You can model the in vitro drug kinetics using a one-compartment model with linear elimination to account for drug degradation due to compound instability. This is represented by the first reaction `Drug_Central -> null`. Include a second compartment (`Biophase`) to incorporate potential pharmacologic delay, which correspond to the second and third reactions: `Drug_Central -> Drug_Central + Drug_Biophase`, `Drug_Biophase -> null`.

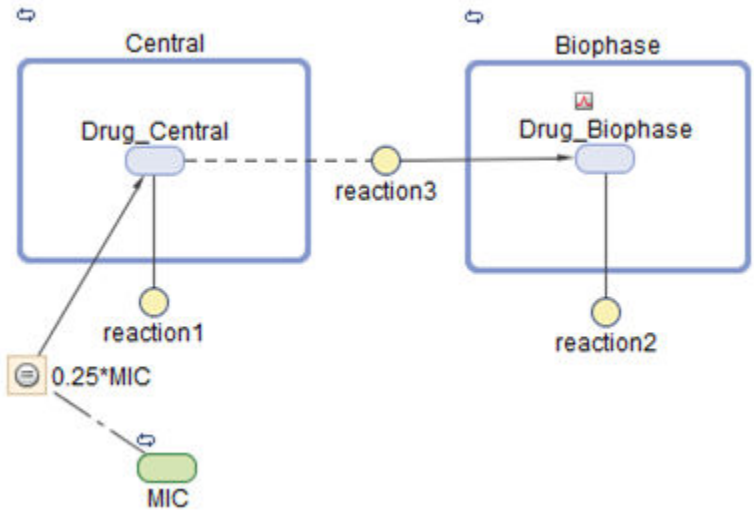


Drag and drop two compartment blocks onto the diagram. Then drop a species block inside each compartment. To build the first reaction, *reaction1*, drag and drop a reaction block onto the diagram. Then draw a line (**Ctrl**+Click or **Options**+Click on a Mac) from the *Drug_Central* species to *reaction1*. Double-click the reaction block and the **Block Property Editor** opens. In the **Quantities Used by Reaction** table, enter *kdeg* in the **Name** column as the **Forward Rate Parameter**. This automatically adds a parameter named *kdeg*, which is used as the forward rate constant for the reaction rate, that is, `kdeg*Drug_Central`. Similarly, configure the second reaction *reaction2*. Enter *ke* as the forward rate parameter for the reaction. To build the third reaction, *reaction3*, draw a line from *Drug_Central* to *Drug_Biophase*. A reaction block is automatically added between two species blocks. The input and output of the *Biophase* compartment are modeled as first-order kinetic processes. *Drug_Central* is both a reactant and product in the input process, assuming that the presence of the *Biophase* compartment does not affect the mass balance. SimBiology indicates such species using a dashed line. The dashed line can be achieved by drawing another line from *reaction2* to *Drug_Central*. You can also update a reaction by dragging a reaction line or arrow (**Ctrl**+Click or **Option**+Click in a Mac) to another species.

Defining or updating a quantity value using mathematical equations

The equations can take the form of initial assignments, assignments during the course of a simulation (repeated assignments), algebraic relationships, or differential equations (rate rules). Each equation is represented by a unique block. For details about rule blocks, see the block library on page 1-43. For more information about rules, see “Definitions and Evaluations of Rules” on page 2-17.

Suppose, in the above example, you want to set the initial concentration of *Drug_Central* to a factor of minimal inhibitory concentration (MIC). You can do so by using an assignment rule that initializes the concentration of *Drug_Central* at simulation time = 0, that is, $Drug_Central_0 = 0.25*MIC$, where *MIC* is a parameter. The following figure shows the graphical representation of the assignment rule.



Drag and drop the initial assignment rule block from the block library onto the diagram. To define the left-hand-side (LHS) of the equation, draw a line from the rule block to *Drug_Central*. An arrow appears pointing at *Drug_Central*. To display the right-hand-side (RHS) of the equation, right-click the rule block and select **Show Only Expressions**. By default, the RHS is set to 1. Double-click it and enter the equation: $Drug_Central = 0.25*MIC$. Drag and drop a parameter block and rename it as *MIC*. A dash–dot line automatically appears connecting *MIC* to the rule block indicating *MIC* is referenced by the RHS. You can also change the LHS by dragging the arrow (**Ctrl**+Click or **Option** +Click in a Mac) to another quantity.

Incorporating sudden changes in model behavior

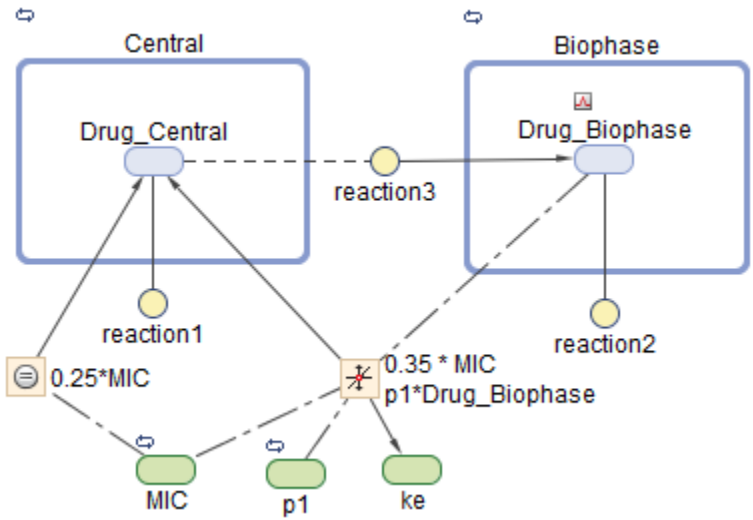
You can model sudden changes in model behavior based on a specified condition. For example, you can reset a parameter value at a certain time point or when a certain concentration threshold is crossed. In SimBiology, you can model such changes using a modeling element called event. An event lets you specify discrete transitions in quantity values that occur when a custom condition becomes true. Such condition is called an event trigger. Once the condition becomes true, one or more event functions are executed. For details, see events on page 2-30.

Suppose you want to set the concentration of *Drug_Central* at time = 5 to another factor of *MIC* as well as the parameter value of the elimination rate constant *ke*. Specifically,

```
if (time >= 5)
    Drug_Central  = 0.35*MIC;
    ke = p1*Drug_Biophase;
```

where *p1* is another parameter.

The next figure shows the graphical representation of the event.

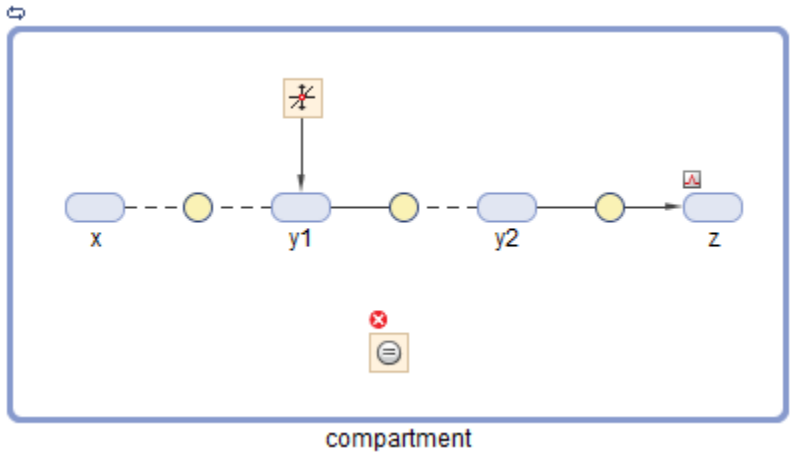


By default, parameter blocks are hidden from the diagram. To see hidden parameter blocks such as *ke*, right-click the diagram, and select **Show All Hidden Blocks and Lines**. Drag and drop an event block onto the diagram. To define the quantities that are

being updated by the event, draw a line from the event block to *Drug_Central* and another line to *ke*. Two arrows appear pointing at *Drug_Central* and *ke*, which are the LHS of event functions. To show the RHS of event functions, right-click the event block and select **Show Only Expressions**. Double-click the expressions to edit the event functions as shown. To see the dash–dot lines indicating quantities referenced in the RHS of an event function, right-click the event block and select **Show Expressions and Lines (Alt +Click)**. In this example, *MIC*, *p1*, and *Drug_Biophase* are the referenced quantities as indicated by dash–dot lines. You can also change the LHS by dragging the arrow (**Ctrl** +Click or **Option**+Click in a Mac) to another species.

Interpreting a model from its diagram

The next figure shows an example of a block diagram of another model and interpretations from looking at the graphical semantics.



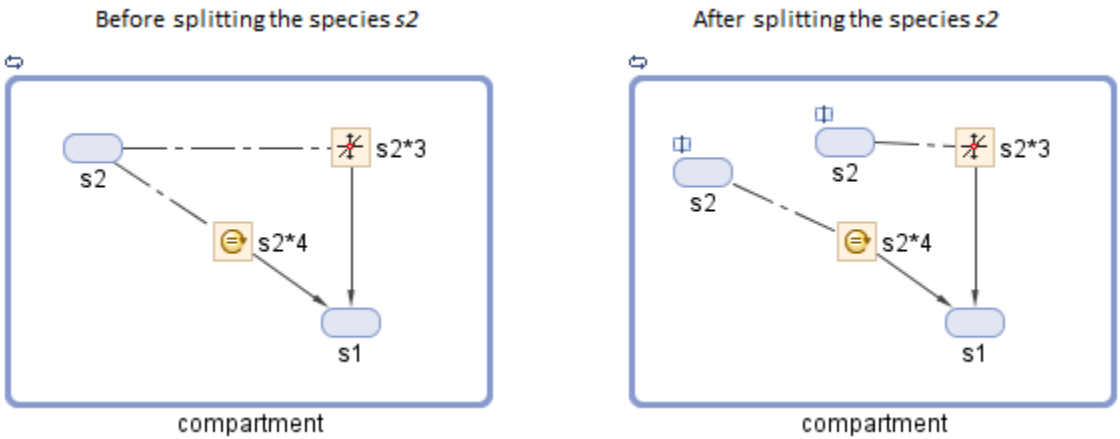
There are three reactions in this model. Species *x*, *y1*, and *y2* are catalysts, that is, both reactants and products, and *z* is a product. The amount of species *y1* is being modified by an event. Different contextual icons above some blocks indicate:

- The amount of species *z* is being increased by a dose.
- There is an error with an initial assignment rule.
- The *compartment* has constant volume.

Making the diagram clearer

When there are multiple references to the same quantity, multiple lines are connected to the quantity block. This can cause the diagram to look cluttered. To make the diagram clearer, you can split the block, that is, create copies of the same block, so that each reference is connected to a different copy of the block. You can also clone a block to add another use for it. For instance, you can first clone a species block that you know will be referenced in multiple expressions. Then use each clone in each expression as you build the model.

SimBiology lets you clone or split a species block. To do so, first select the block. Then select **Split** or **Clone** from the drop-down menu of the **Split** button on the **Block** tab. Event blocks can only be split but not cloned. The next figure shows an example where a species (*s2*) is referenced by a repeated assignment rule and event, and splitting the block creates a copy for each reference. Copied blocks are then marked by a contextual icon to indicate that they have been cloned.



You can also hide blocks to avoid clutter. To hide a block, right-click it, and select **Hide Block** from the context menu. By default, blocks that represent constant parameters are hidden. Hidden blocks can be shown by selecting **Hidden Blocks** on the **Block** tab.

Browser

The browser supplements the diagram by providing a table of model elements that shows the relationships between quantities and expressions. You can edit these elements in the browser or add new quantities or expressions by dragging and dropping blocks from the

block library panel onto the browser. If you make a change in the browser, the diagram is automatically updated.

You can view the model elements shown in the browser by quantities or expressions. In the quantities view, the browser shows all quantities of the model. If a quantity is being modified by any expressions and doses, the view also lists them. Alternatively, the expressions view lists all expressions of the model. It also shows quantities that are referenced by each expression. To switch between the two views, select **View > By Quantities** or **View > By Expressions** on the **Model** tab.

View By Quantities

The quantities view shows a table of quantities and expressions that are modifying them. Use this view to check the values of quantities and if they are being modified by expressions such as events or assignment rules. You can also see the right-hand-side, of each modifying expression. To display quantity units, select **Tools > Show Quantity Units** on the **Model** tab. The check box next to each expression indicates whether the expression is active and used during simulation.

You can model biological variability using a modeling element called variant. A variant is a collection of quantities with alternative values. For instance, in the above example, you can have one set of parameter values such as the elimination rate (*ke*) and degradation rate (*kdeg*) for each antibacterial drug.

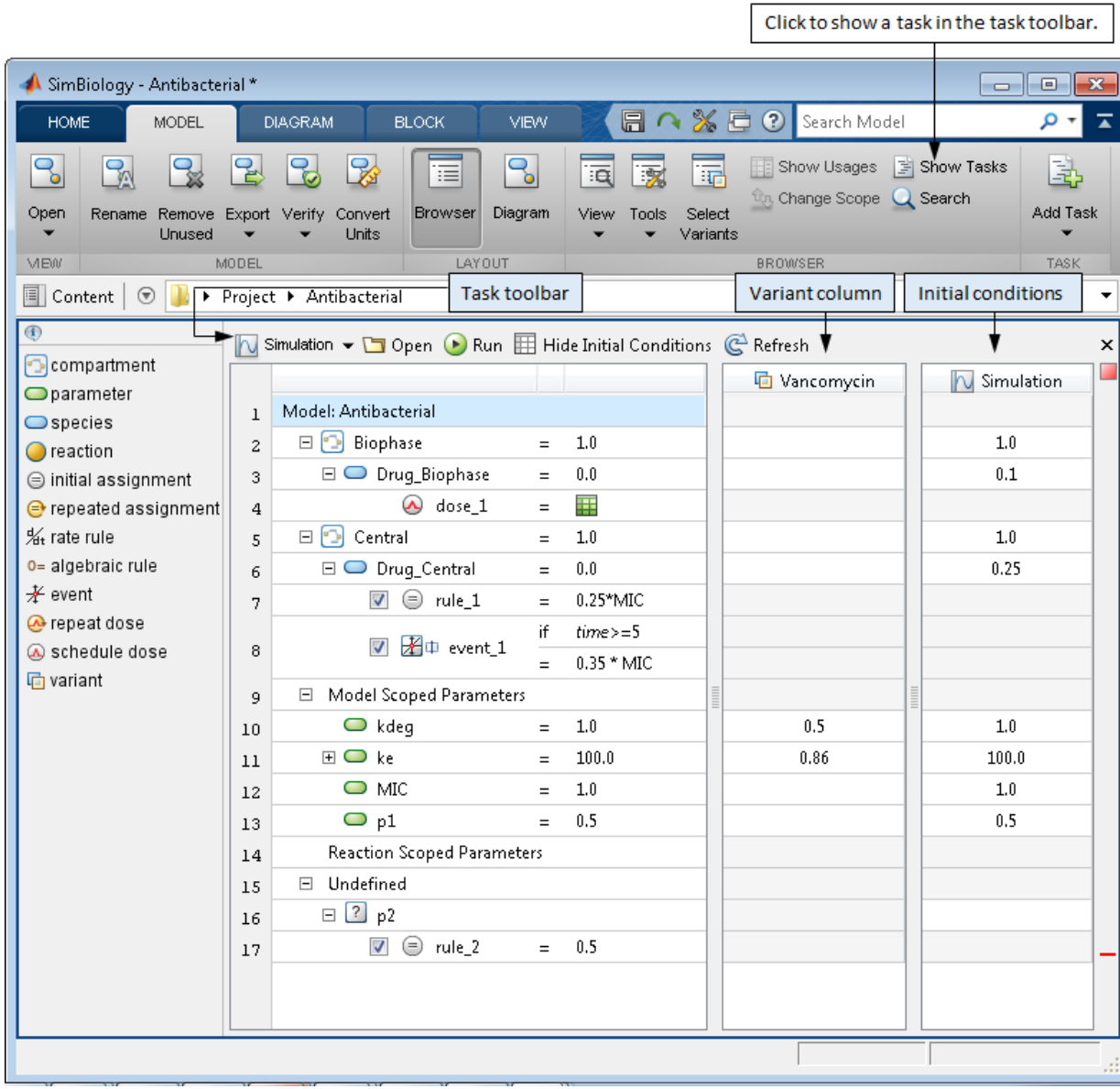
The browser shows a variant column for each variant of a model. For instance, a variant named *Vancomycin* is shown in the next figure. You can edit each quantity value by double-clicking it. When there are multiple variants, you can display a subset by clicking **Select Variants** on the **Model** tab. To add a variant, drag and drop a variant block from the block panel onto the browser.

If there are expressions for which the left-hand-sides are not defined, these expressions are listed under the section named **Undefined**. For example, in the next figure, the initial assignment *rule_2* is modifying a quantity *p2* that is not defined yet in the model. You can right-click the undefined quantity and define it as a species, parameter, or compartment.

Before running any analysis task, SimBiology prepares a model for simulation and updates the quantity values according to the variant values, assignment rules, and doses. For details, see “Model Simulation” on page 4-2. You can check if the quantity values are initialized as you expect by checking the initial conditions of the task. To see the initial conditions, first, select **Show Tasks** on the **Model** tab. Then select a task in the task toolbar. For instance, in the following figure, the initial conditions of a simulation task is shown. In this task, a variant called *Vancomycin* and a dose called *dose_1* have

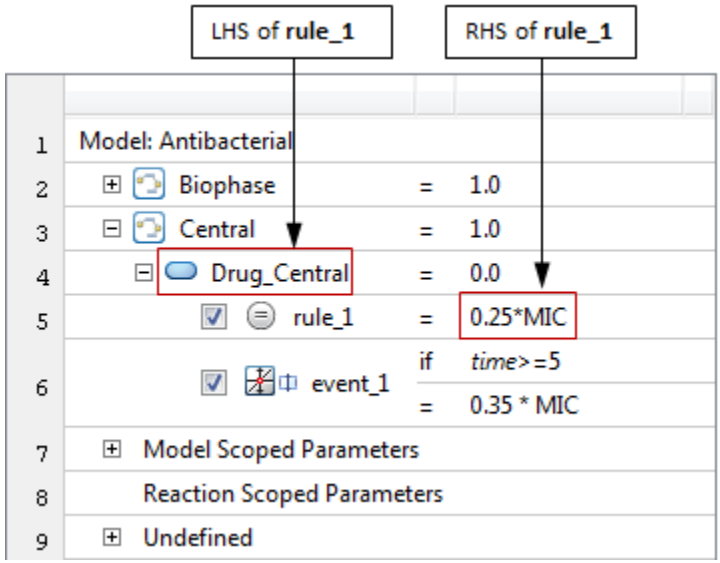
been selected to apply to the model during simulation. The concentration of *Drug_Central* at the beginning of the simulation is 0.25, compared to the model value 0.0. The value is updated because, at simulation time = 0, SimBiology evaluates the initial assignment *rule_1* that initializes *Drug_Central* to `0.25*MIC`. Since the variant *Vancomycin* has been selected, the alternate values stored in it, namely `0.5` and `0.86` for *kdeg* and *ke*, are used instead of model values.

Since a dose is applied to species *Drug_Biophase* at time = 0, the browser displays the total amount of the species, that is, the initial condition value of the species plus the dose amount. For instance, in the next figure, the total amount of species *Drug_Biophase* after applying the dose is 0.1. Alternatively, you can view the initial value and dose amount separately by selecting **Tools > Task Initial Conditions Options > Show the dose amount separate from the initial condition value** from the **Model** tab.



Defining or updating a quantity value using mathematical equations. The equations can take the form of initial assignments, assignments during the course of a

simulation (repeated assignments), algebraic relationships, or differential equations (rate rules). For details, see rules on page 2-17. In the following example, the concentration of *Drug_Central* at time = 0 is defined by an initial assignment equation, that is, $Drug_Central_0 = 0.25*MIC$. The next figure shows how the browser displays the assignment rule in relation to the species *Drug_Central*.



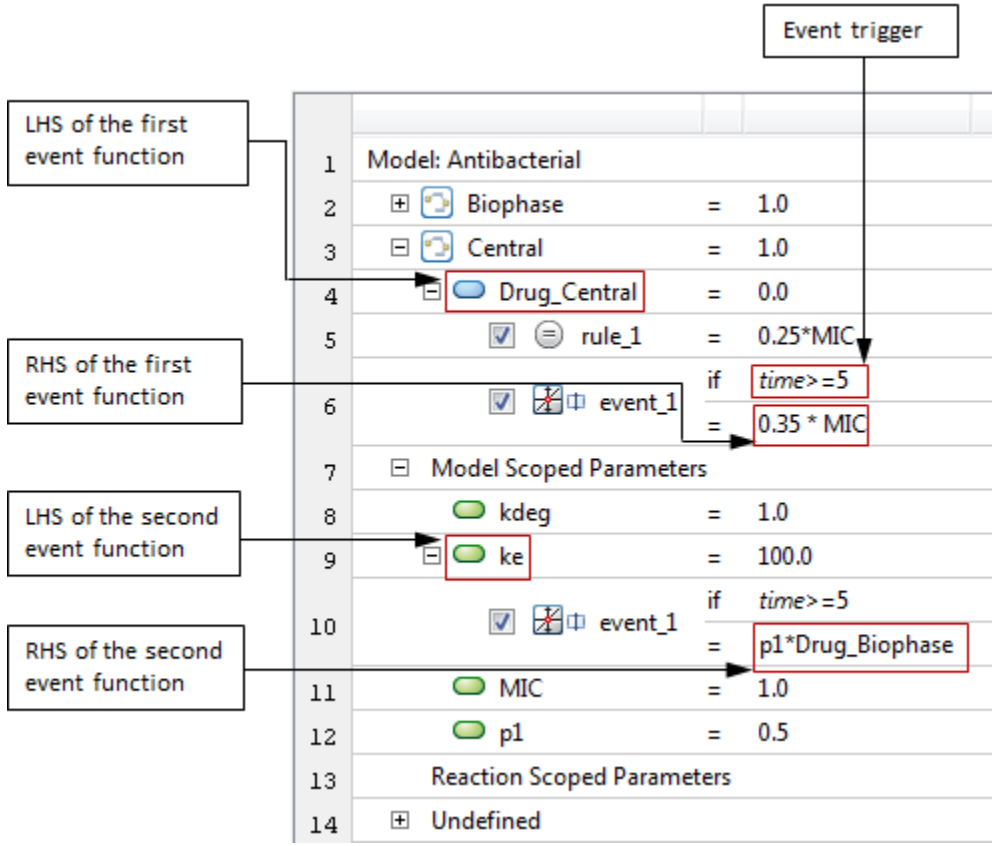
Drag and drop the initial assignment rule block onto the species *Drug_Central*. The rule block appears as a separate row below the species. Double-click the RHS of the equation to edit it.

Incorporating sudden changes in model behavior. You can model sudden changes in model behavior based on a specified condition by using a modeling element called event. In the following example, you want to set the concentration of *Drug_Central* at time = 5 to another factor of *MIC* as well as the parameter value of the elimination rate constant *ke*. Specifically,

```
if (time >= 5)
    Drug_Central  = 0.35*MIC;
    ke = p1*Drug_Biophase;
```

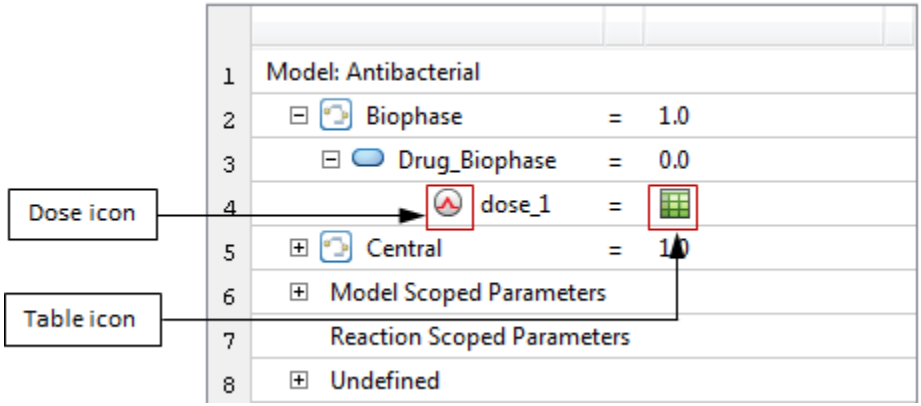
where *p1* is another parameter.

The next figure shows how the browser displays the event.



Drag and drop the event block onto the species *Drug_Central*. An event appears as a separate row below the species. To add the second event function, right-click the event and select **Add EventFcn**. Double-click the event trigger or RHS of an event function to edit.

Increasing a species amount using doses. You can model the increase in the amount of a species due to a stimulus such as an oral or intravenous administration of a drug. To model such an increase in a species amount, use the modeling element called dose on page 2-42. In the following example, the concentration of species *Drug_Biophase* is being increased by a dose.



Drag and drop a dose block onto the target species. A dose appears as a separate row below the species. To edit the dose amount, double-click the table icon. In this example, the concentration of species *Drug_Biophase* is being increased by a schedule dose *dose_1*. For more details about dose objects, see “Doses” on page 2-42.

View by Expressions

In the expressions view, you can see all expressions of a model grouped by their types such as reactions, assignment rules, and events. It shows how each expression is defined and the relationship between each expression and quantities. The check box next to each expression indicates whether the expression is active and used during simulation.

You can expand each reaction in the browser to configure its properties, such as reaction rate, kinetic law, quantities referenced by the reactions, and quantity values. You can define your own reaction rate or use a predefined rate that follows some particular reaction kinetics such as mass action.

A built-in kinetic law in SimBiology specifies a rate law that defines the reaction rate. Specifically each law has a predefined reaction rate containing parameters and species that need to be mapped to the corresponding model quantities to determine the final reaction rate. The browser displays the mapping information in parentheses next to a quantity. For instance, if a parameter `kf` is used as the forward rate parameter in the reaction rate, the browser displays `kf(Forward)`. In case you map incorrectly, you can reset the mapping by right-clicking the quantity and selecting **Remove Kinetic Law Mapping**.

The kinetic law for a newly added reaction is configured to **MassAction** by default, and the desktop automatically creates and maps the species and parameters needed by the

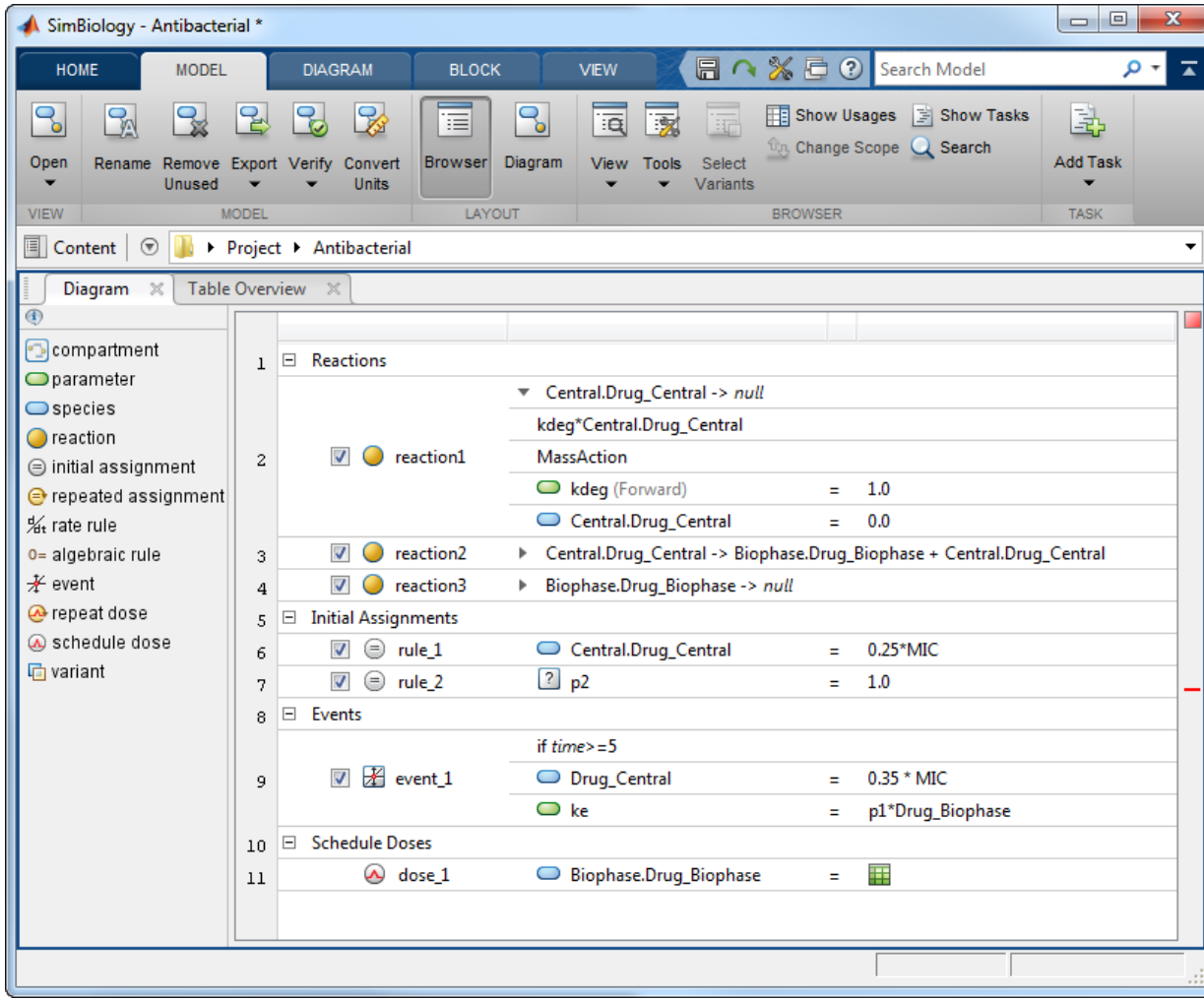
reaction rate. For other kinetic laws, only parameters are created and mapped. You need to create and map the species manually. To change the default kinetic law and reaction building settings, select **Tools > Reaction Building Preferences** on the **Model** tab.

This view also displays the assignment rules grouped according to their types such as initial assignment, repeated assignment, rate rule, and algebraic rule. For each rule, you can see its left-hand-side (LHS) and right-hand-side (RHS). The LHS is the quantity that is being modified by a rule, except for the algebraic rule. The algebraic rule takes the form `0 = Expression`, and the rule is specified as the `Expression`. For details, see “Definitions and Evaluations of Rules” on page 2-17.

This view also shows events. For each event, its trigger and event functions are grouped together. Each trigger is started with the word `if` followed by an expression such as `if time>=3`. You can also configure the LHS and RHS of each event function.

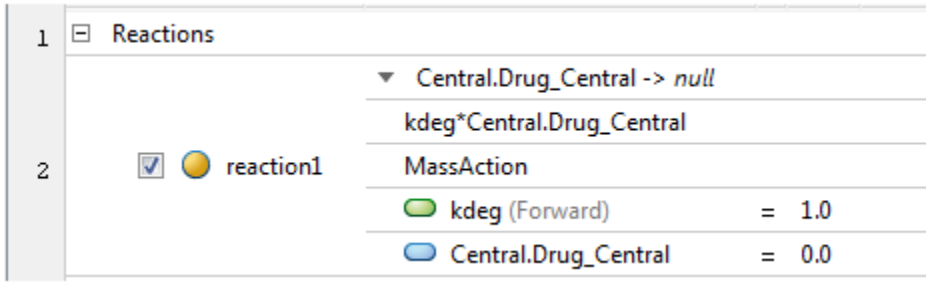
In addition to all the expressions, you can also see doses on page 2-42 that are grouped into either schedule doses or repeat doses. For each dose, the corresponding dosed species, that is, dose target, is shown right next to it. You can double-click the dose icon to edit all dose properties or the table icon to edit just the dose schedule.

The following figure shows a model open in the expressions view of the browser.



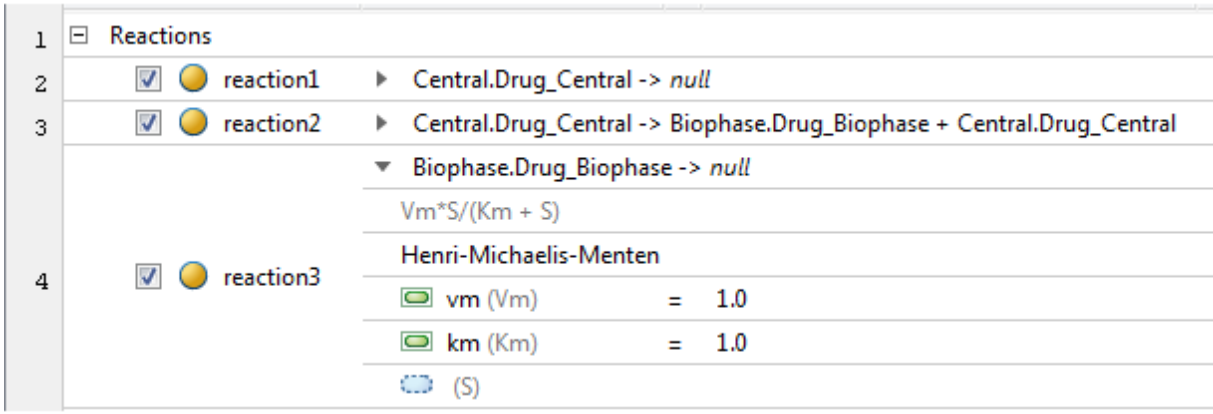
Adding and configuring reactions. There are three reactions in this example. Suppose the first and second reaction follow the mass action kinetics and the third reaction follows the Michaelis-Menten kinetics. For illustration purposes, the configuration of *reaction1* and *reaction3* is described below.

The following figure shows the details of *reaction1*. It follows the mass action kinetics with the reaction rate `kdeg*Central.Drug_Central`.



To add a reaction, drag and drop a reaction block onto the browser. You can then enter the reaction string by double-clicking the default `null -> null` string. Since there are species in two compartments, qualify the species name with the compartment name. For example, *Central.Drug_Central* indicates that the *Drug_Central* species is inside the *Central* compartment. By default, the reaction's kinetic law is configured to **MassAction**. The parameter *kf* is automatically created and mapped to the forward rate constant of the reaction rate. If you want to use a different parameter, you can change it by double-clicking *kf* and entering the name of another parameter such as *kdeg*. You can then delete the parameter *kf* which is no longer used in the reaction rate.

The following figure shows the details of *reaction3* that is configured to follow Michaelis-Menten enzyme kinetics with the reaction rate `Vm*S/(Km+S)`.

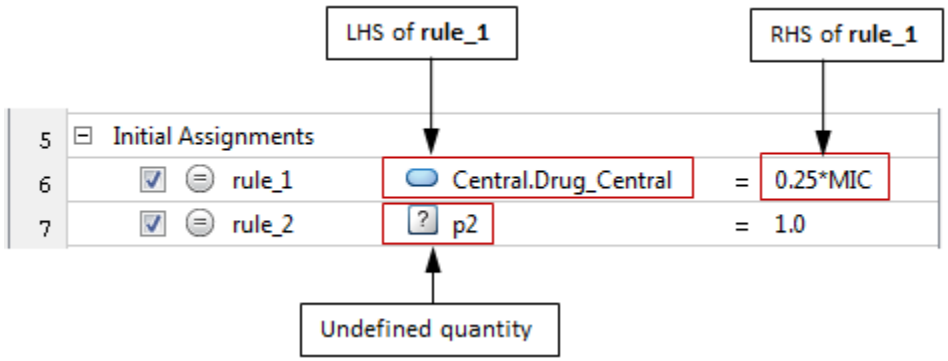


Drag and drop a reaction block onto the browser. By default, it uses the mass action kinetic law. The parameter *kf* is automatically created and mapped to the forward rate constant. To specify the reaction to follow Michaelis-Menten kinetics, double-click the row

labeled **MassAction** and select **Henri-Michaelis-Menten**. SimBiology automatically updates the reaction rate to `Vm*S/(Km+S)`. The parameters *vm* and *km* are also automatically added and mapped to *Vm* and *Km*, which are the parameters referenced in the reaction rate. However, the *S* species referenced in the reaction rate is not mapped yet as indicated by a dotted species block. Double-click the *(S)* row to enter a model species that corresponds to the species *S*. The reaction rate is then automatically updated using the selected species. To avoid confusion, you may delete the parameter *kf*, which is no longer used in the reaction rate.

Adding and configuring assignment rules. In this example, there are two initial assignment rules on page 2-17. The first initial assignment *rule_1* initializes the concentration of species *Drug_Central*. The second initial assignment *rule_2* initializes an undefined quantity *p2*.

The next figure shows how the expressions view displays such assignment rules.



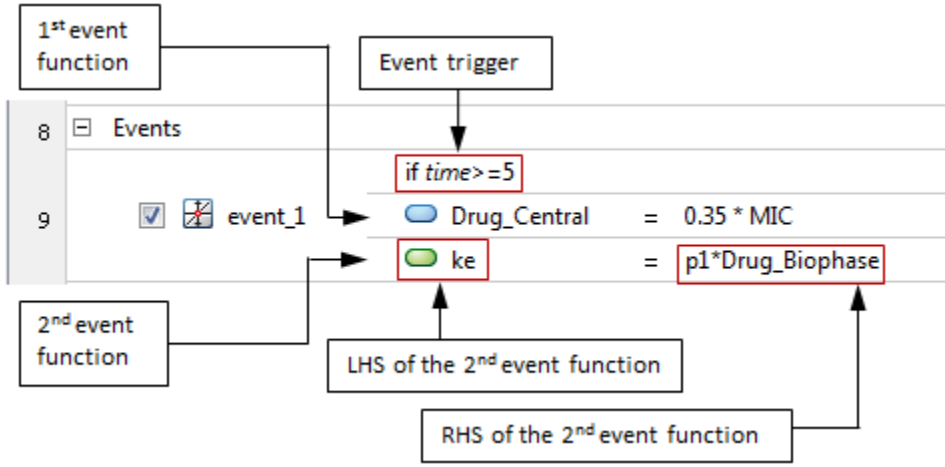
Drag and drop the initial assignment rule block onto the browser. Define the LHS of the rule by double-clicking the string `null` and entering the name of a quantity that the rule is modifying. Double-click the RHS of the equation to edit it. If the LHS of the rule is referring to a quantity that is not yet defined in the model, such as *p2* in this example, right-click the undefined quantity and select **Define Quantities** to define it as a species, parameter, or compartment.

Adding and configuring events. In this example, there is one event *event_1* with two event functions changing the concentration of species *Drug_Central* and parameter *ke* respectively. Specifically,

```
if (time >= 5)
    Drug_Central  = 0.35*MIC;
    ke = p1*Drug_Biophase;
```

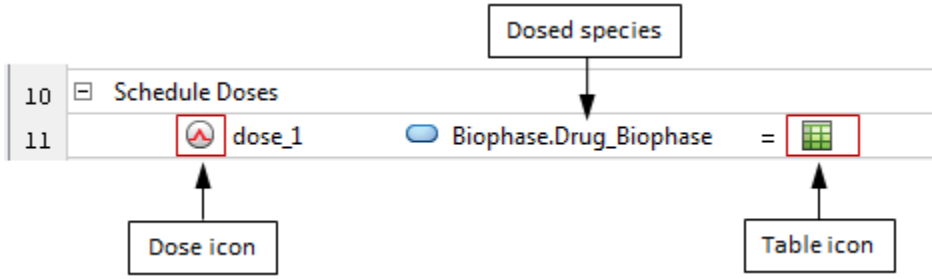
where *p1* is a parameter.

The next figure shows how the expressions view displays the event.



Drag and drop the event block onto the browser. Double-click the default trigger `if time>=1` to edit it. The default event function, `null = 0.0`, is listed below the trigger. Double-click its LHS and RHS to edit. To add more event functions, right-click the event and select **Add EventFcn**.

Adding and configuring doses. You can increase the amount or concentration of a species using a dose during simulation. In this example, the concentration of species *Drug_Biophase* is being increased by a schedule dose *dose_1*. The next figure shows how the expressions view displays the dose.



Drag and drop a dose block onto the browser. Define the dosed species by double-clicking the LHS of the table icon. Double-click the dose icon to edit all dose properties or the table icon to edit just the dose schedule.

Block Library Panel

The block library panel contains blocks from one or more block libraries. SimBiology provides a default block library which contains blocks that represent all modeling elements that you can use to build models. You can also create a library of custom blocks with different block appearance settings. For instance, you can customize a receptor protein to have a different block shape or color than other species whenever you use it in the diagram. For details on the SimBiology libraries, see “Libraries” on page 1-56.

You can drag and drop most blocks from the panel to both the browser and diagram, and there are some blocks that can be dropped only in the browser, not in the diagram. Some blocks do not appear as blocks in the diagram but as contextual icons on page 1-15. For instance, a dose block appears as a contextual icon above a species block that is being dosed. The following table summarizes the behaviors of different built-in block types.

Block Name	Block Graphics	Drag and Drop onto the Browser	Drag and Drop onto the Diagram	Block appears as	Description
Species		Yes	Yes	Block	Quantity that participates in expressions, such as reactions and represents an amount or concentration.

Block Name	Block Graphics	Drag and Drop onto the Browser	Drag and Drop onto the Diagram	Block appears as	Description
Compartment		Yes	Yes	Block	Physically bounded region that contains species in a model. All species in a model must belong to a compartment. A compartment can also belong to another compartment. For instance, a compartment that represents a cell can contain other cellular components such as mitochondria and nucleus as separate compartments within the cell compartment.
Parameter		Yes	Yes	Block	Quantity that is referenced by expressions. For instance, you can use it to define a rate constant of a reaction. By default, if you drop a parameter block onto the browser or diagram, it is available to all expressions including reactions. This type of parameter is called a model-scoped parameter and listed under the **Model Scoped Parameters** section in the quantities view of the browser. Another type is called a reaction-scoped parameter, where the parameter is only available to one particular reaction. These parameters are listed under the **Reaction Scoped Parameters** section. To add a reaction-scoped parameter, switch to the expressions view of the browser and then drop a parameter block onto a particular reaction. To switch between the two types, right-click a parameter block and select **Change Scope**.

Block Name	Block Graphics	Drag and Drop onto the Browser	Drag and Drop onto the Diagram	Block appears as	Description
Reaction		Yes (only in the expressions view of the browser)	Yes	Block	Process such as a transformation, transport, or the binding and unbinding of reactants and products.
Initial assignment		Yes	Yes	Block	Expression to assign the initial value of a quantity. For example, an initial assignment rule `s4 = 0.75` means that the amount of species *s4* is initialized to 0.75 at simulation time = 0.
Repeated assignment		Yes	Yes	Block	Expression to assign a quantity value repeatedly during simulation. For example, a repeated assignment rule `x = y1 + y2` means that the amount of species *x* is repeatedly assigned the total amount of species *y1* and *y2* throughout the simulation.
Rate rule		Yes	Yes	Block	Differential equation to specify the time derivative of a model quantity. For example, a differential equation `x = k * (y + z)` specifies that the time derivative of species *x* (`dx/dt`) is evaluated continuously during the simulation according to the equation.

Block Name	Block Graphics	Drag and Drop onto the Browser	Drag and Drop onto the Diagram	Block appears as	Description
Algebraic rule		Yes	Yes	Block	Expression to specify a mathematical constraint (for example, a nonlinear equation) on one or more quantities that must hold during simulation. For example, if you have an equation such as `x = log(x+time)`, write the corresponding algebraic rule as `log(x+time)-x`. **Tip** Use algebraic rules only for nonlinear equations that cannot be solved analytically to get closed-form solutions. For details, see "Algebraic Rules" on page 2-18.
Event		Yes	Yes	Block	Discrete transition in a quantity value. This discrete transition occurs when a specified condition becomes true.
Repeat dose		Yes	No	Contextual icon on page 1-15 above a species block that is being dosed	Modeling element that increases the value of a species by a certain amount at predefined time intervals. To add a dose to a species, drag and drop the dose block onto the species in the quantities view of the browser. Double-click the dose icon to edit the dose value.

Block Name	Block Graphics	Drag and Drop onto the Browser	Drag and Drop onto the Diagram	Block appears as	Description
Schedule dose		Yes	No	Contextual icon on page 1-15 above a species block that is being dosed	Modeling element that increases the value of a species by certain amount at specific time points. To add a dose to a species, drag and drop the dose block onto the species in the quantities view of the browser. Double-click the dose icon to edit the dose value.
Variant		Yes (only in the quantities view of the browser)	No	Column in the Browser	Modeling element that is a collection of quantities with alternate values from the original model values.
Text		No	Yes	Block	Text block to annotate the model in the diagram.

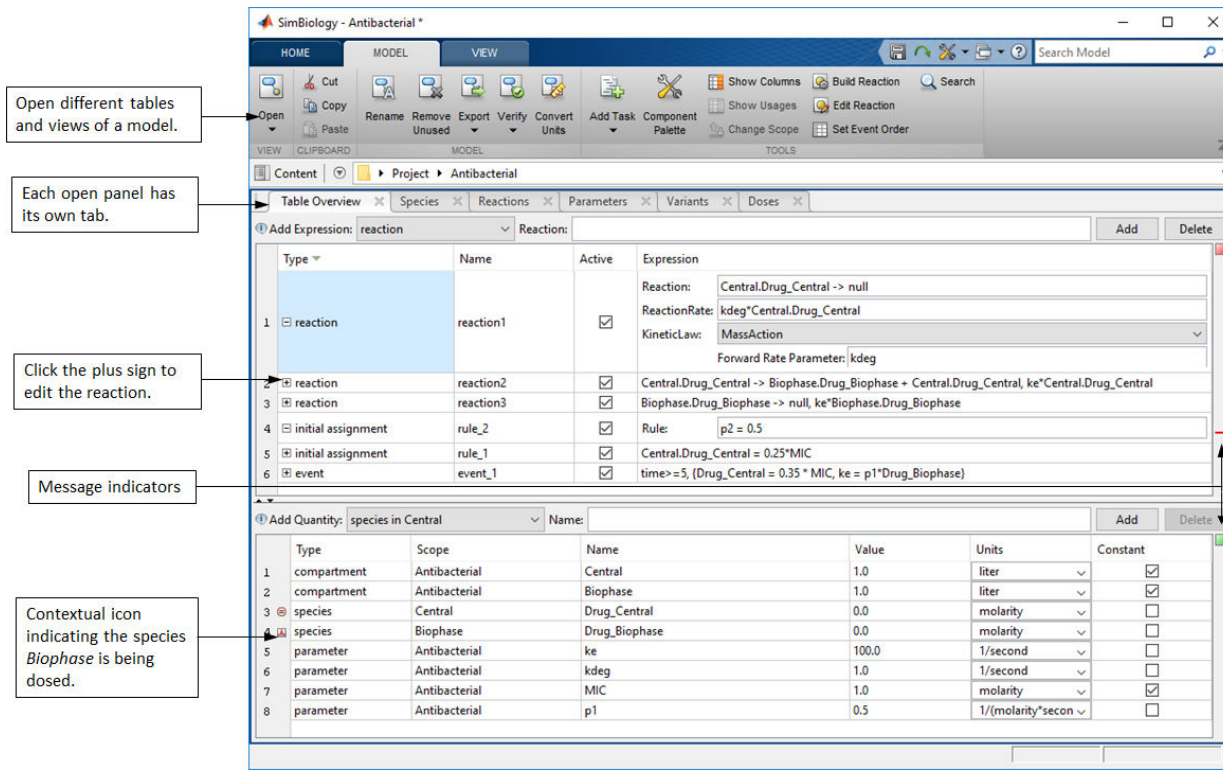
Table View

This view presents a model in a tabular format, and there are many tables that organize the model information differently. For instance, the Table Overview gives an overview of the entire model by showing the complete list of model quantities, expressions, and some

of their properties. There are also tables dedicated to each type of quantity or expression, doses, and variants, and these individual tables provide more information specific for each type.

You can open different tables from the **Open** drop-down list to edit and add elements to each table. For instance, you can enter a reaction string such as `y2 -> z` in the Table Overview or Reactions table.

The context menu, which you can open using a right-click, provides options to customize the table and open other modeling tools. For instance, to check which expressions are referencing a quantity, right-click the quantity and select **Show Usages** from the context menu.



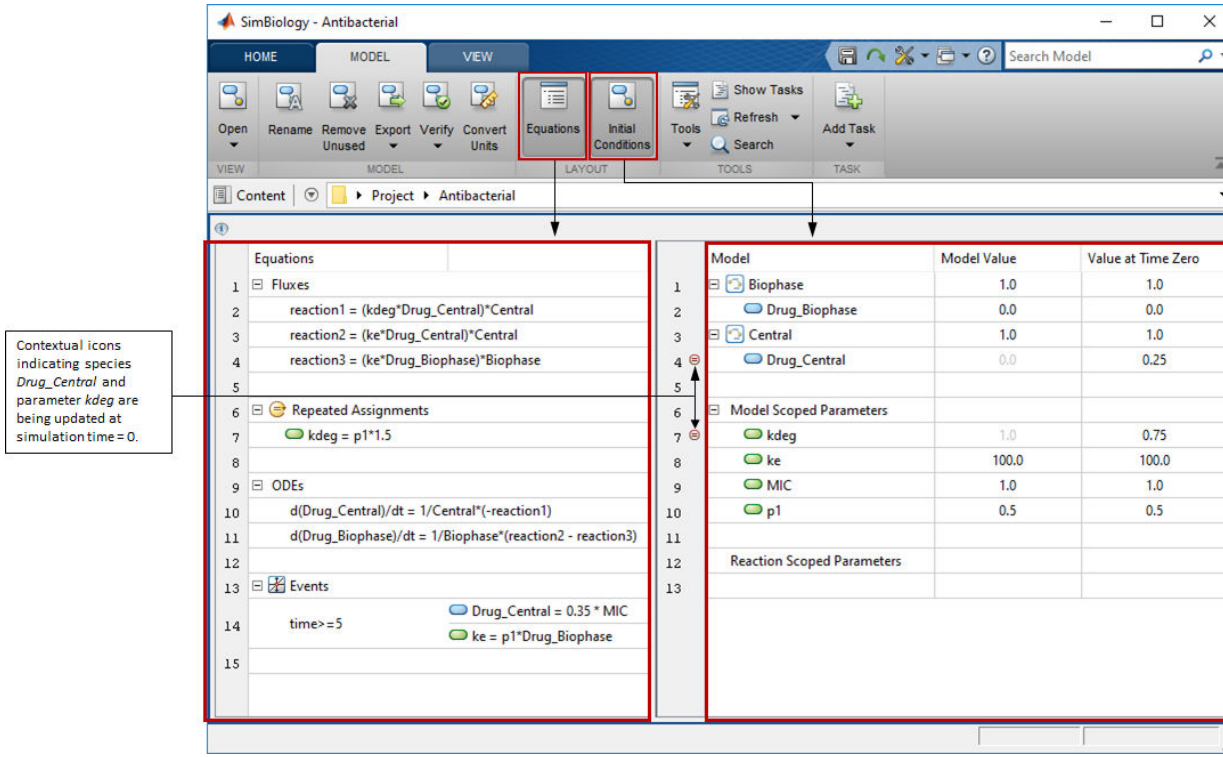
Equations View

This view describes a model in the form of mathematical equations. It shows a system of ordinary differential equations (ODEs) that represent a model. Specifically, the ODEs are derived from model reactions and define what quantities are being integrated during model simulation. For details about the simulation process and how SimBiology constructs ODEs, see “Model Simulation” on page 4-2.

You can use this view to help debug a model. For instance, you can check the initial conditions of ODEs to see if the quantity values are initialized as you expect. You can also see how SimBiology corrects the dimensions of ODEs by dividing the right-hand-sides of equations with compartment volumes. The volume-correction information can help debug unexpected simulation results, especially when you have a multi-compartment model with different compartment volumes. You can also compare the ODEs to the ones from a publication to reproduce the reported results.

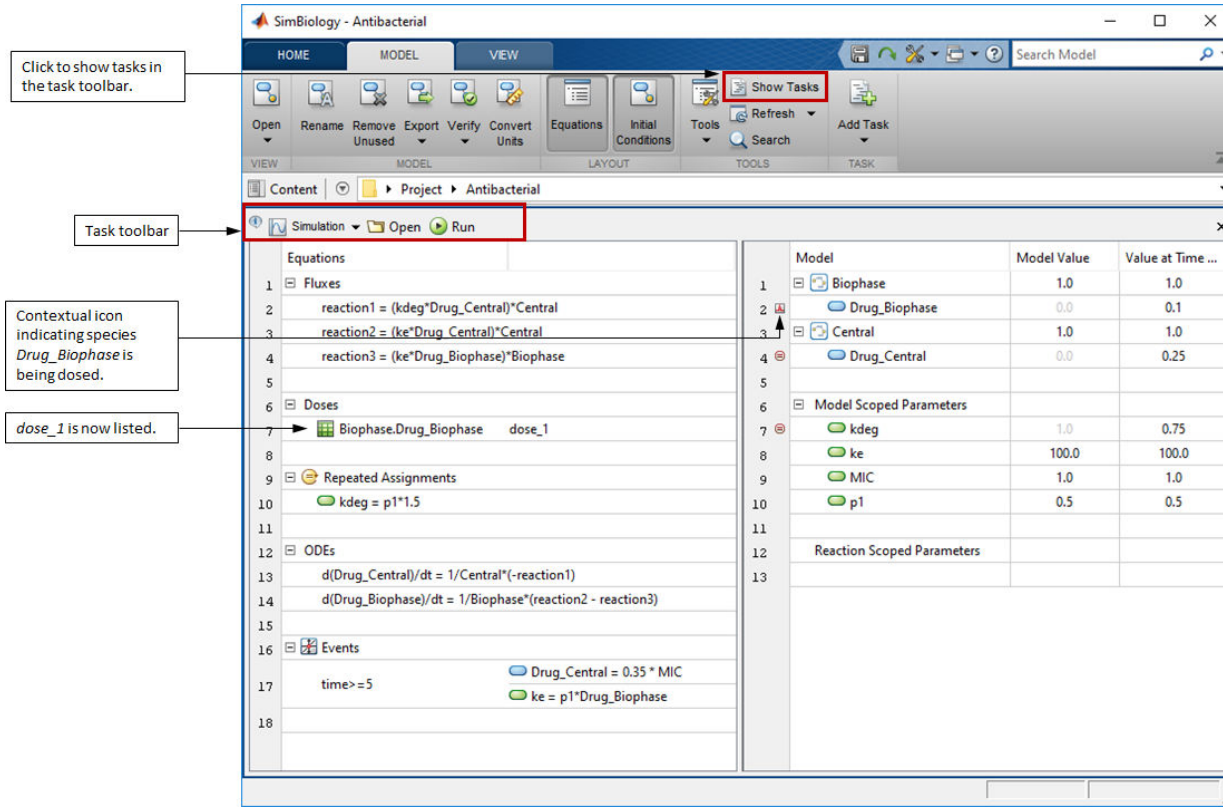
The initial conditions of ODEs are used as starting points to simulate the dynamics of a model. The initial conditions are the quantity values at simulation time = 0. The **Value at Time Zero** column of the view shows these values. You can check this column to see if the quantity values are initialized as you expect based on initial assignment and repeated assignment rules, and debug the rules as necessary. For details about how SimBiology evaluates the assignment rules, see “Model Simulation” on page 4-2.

The next figure shows a model open in the equations view, which has the **Equations** and **Initial Conditions** sections. The **Equations** section contains the expressions and ODEs that SimBiology evaluates during model simulation. The **Initial Conditions** section contains the quantity values at simulation time = 0. Since initial assignments are evaluated just once at time = 0, they are only accessible in the **Initial Conditions** section via a contextual icon. Repeated assignments are evaluated both at time = 0 and during simulation. Hence they are shown in the **Equations** section and also indicated by contextual icons.



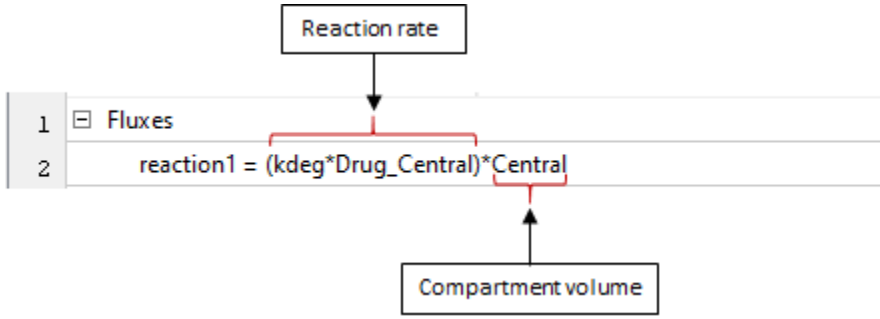
In this model, the concentration of drug in the central compartment *Drug_Central* is initialized to `0.25*MIC`, resulting in the amount of 0.25 as shown in the **Value at Time Zero** column. To see the initial assignment equation, click the contextual icon next to *Drug_Central*. In addition, a repeated assignment equation sets the value of parameter *kdeg* to `p1*1.5`, resulting in the value of 0.75. You can see the equation listed under the **Repeated Assignments** section. The desktop also shows a contextual icon next to the parameter *kdeg* to indicate the application of the assignment rule.

To perform an analysis on a model such as simulation or sensitivity calculation, you need to run a task. You can specify which doses and variants to use for the analysis in the task configuration on page 1-70. When you select a task, the desktop applies the doses and variants specified in the task, hence updating the corresponding quantity values in the **Value at Time Zero** column. The dose is also added to the **Equations** section. The next figure shows the equations view of the same model with a task selected. In this example, the task is a simulation task that uses a dose *dose_1* to increase the concentration of drug in the *Biophase* compartment *Drug_Biophase* at simulation time = 0.

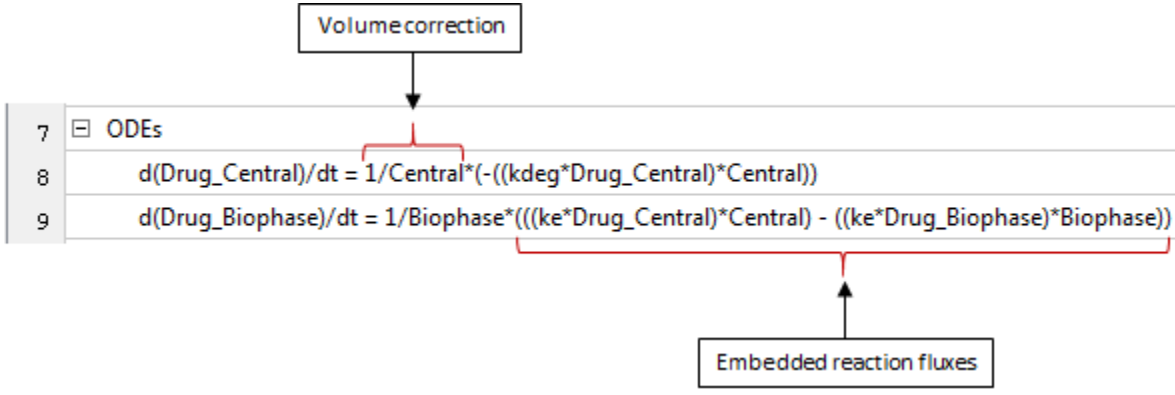


The grayed-out numbers in the **Model Value** column indicates the quantity values that have been changed due to the task configuration. The model value and the value at time zero are considered different if the relative tolerance between the values is greater than 10^{-12}, that is, `|x-y| > 1e-12*min(|x|,|y|)` where x is the model value and y is the value at time zero.

The **Fluxes** section of the equations view contains reaction fluxes for all reactions in the order they appear in the model. Reaction fluxes are equivalent to reaction rates except that the dimensions of fluxes are always `amount/time`. Hence, if the dimension of a reaction rate is `concentration/time`, the expression for the reaction flux is equal to the reaction rate multiplied by a compartment volume. SimBiology uses the name of a compartment to represent its volume in reaction fluxes and ordinary differential equations (ODEs). For details, see “Deriving ODEs from Reactions” on page 4-4. The next figure illustrates an example where *reaction1* is volume-corrected.



While preparing a model for simulation and other analyses, SimBiology performs dimensional analysis to make sure the dimensions of the left-hand-side (LHS) and right-hand-side (RHS) of each ODE are consistent. The LHS of each ODE is the time-derivative of the amount or concentration of a species, and the RHS is defined using reaction fluxes. If you specify no units, the default dimension for a species is `concentration`, and the default dimension for a flux is `amount/time`. For such cases, SimBiology divides the RHS by a compartment volume to make the dimensions of LHS and RHS consistent. For details, see “Deriving ODEs from Reactions” on page 4-4. You can see all the ODEs in the **ODEs** section of the view as shown in the next figure. In this example, SimBiology uses the default dimensions for species and fluxes. Hence each RHS of ODE is divided by the corresponding compartment volume *Central* or *Biophase* for volume-correction. By default, the flux names are used in ODEs. To use the expressions explicitly instead of using the flux names, select **Tools > Embed fluxes** on the **Model** tab. The following figure shows ODEs after fluxes have been embedded.



References

[1] Nielsen, E. I., Viberg, A., Lowdin, E., Cars, O., Karlsson, M.O., and Sandstrom, M. (2007) Semimechanistic pharmacokinetic/pharmacodynamic model for assessment of activity of antibacterial agents from time-kill curve experiments. Antimicrobial Agents and Chemotherapy. 51:128-136.

See Also

More About

- “General Workflow” on page 1-2
- “Modeling Workflow” on page 1-22
- “Analysis Workflow” on page 1-60

Setting Preferences

Preferences are the desktop settings that apply to all projects. They remain persistent across sessions of MATLAB and the desktop. To access and set preferences, select **Preferences** on the **Home** tab.

Desktop display — has options to choose which columns to display for tables shown in the desktop.

Model Building — lets you select which model view to show by default when you open a model. The Diagram view on page 1-25 is the default. Other options are Table Overview on page 1-47 and Equations view on page 1-49. You can also change the default options related to configuring reactions. For instance, you can change the default kinetic law or disable the option to create parameters automatically. For details about configuring reactions, see “View by Expressions” on page 1-37.

Indicators — lets you choose whether to show indicators above the blocks or next to a quantity in a table. For details, see “Message Indicators” on page 1-11 and “Contextual Icons” on page 1-15.

Tasks — has options to choose the layout of the Task Editor on page 1-63 and change task settings.

Report Generator — lets you select the output format of the report, type of images, and location of images after you have run the Generate report task.

Search — lets you choose in which model view a search item is displayed when you double-click it from the search results. The diagram view on page 1-25 is the default.

History — lets you choose the number of most recently used models, projects, and data files to show when you click **Recent Files** in the content panel on page 1-5. You can also specify the number of recent searches that show up when you select **View Search History** from the search drop-down menu.

Confirmation Dialogs — lets you specify whether or not the desktop displays confirmation dialog boxes before some desktop actions. For instance, a warning dialog is displayed by default when you try to delete data from a project.

See Also

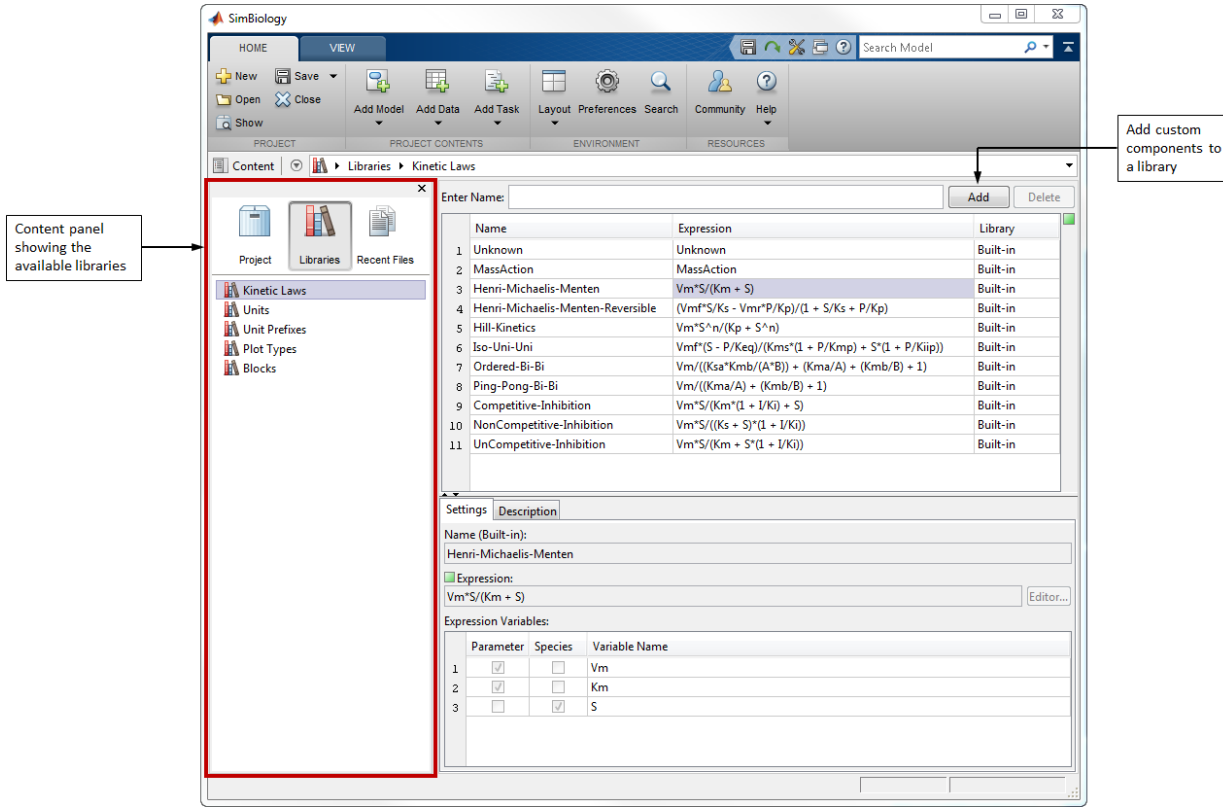
More About

- “General Workflow” on page 1-2
- “Modeling Workflow” on page 1-22
- “Libraries” on page 1-56

Libraries

The SimBiology libraries are collections of built-in components that you can use to build and analyze models. For instance, you can use built-in units such as `mole` or `molecule` as amount units for species. As you perform model analyses, you can use built-in plots such as box plot or residuals plot to display the analysis results. You can also add custom components to any library. For instance, you can define a custom unit or block and use it as you build models. Similar to preferences on page 1-54, libraries are saved across sessions of MATLAB and the SimBiology desktop. Libraries are available for all projects and are not part of any one project. You can also export libraries and share with others by selecting **Export All Libraries** from the action menu.

To open a library in the SimBiology desktop, select **Content > Libraries** and double-click a library. For instance, the following figure shows the kinetic laws library with all the available built-in kinetic laws.



Kinetic Laws Library

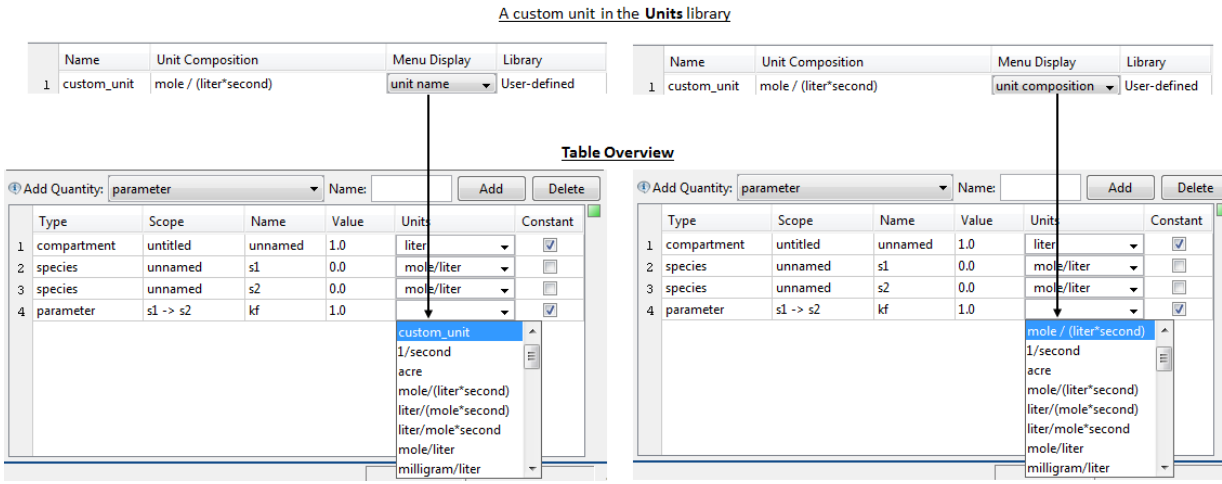
As you configure model reactions, you can use predefined reaction rates that follow particular kinetics such as mass action or Michaelis-Menten. SimBiology provides a list of such predefined rates in the **Kinetic Laws** library.

The kinetic law for a newly added reaction is configured to **MassAction** by default, and SimBiology automatically creates and maps the species and parameters needed by the reaction rate. For other kinetic laws, only parameters are created and mapped. You need to create and map the species manually. For details about configuring reaction rates in the SimBiology desktop, see “View by Expressions” on page 1-37. Use the **Unknown** kinetic law to define a custom reaction rate with its own parameters. You must define and add the species and parameters needed by the custom rate.

Note The **MassAction** and **Unknown** kinetic laws can have different simulation results even when the reaction rate is the same. This can happen when you have a reversible reaction with species in different compartments. The difference in simulation results is because of the volume-scaling performed by SimBiology during the dimensional analysis. For details, see “Deriving ODEs from Reactions” on page 4-4. Specifically, for **MassAction**, SimBiology uses corresponding compartment volumes to multiply the forward and reverse rates. However, for **Unknown** and other built-in kinetic laws, SimBiology multiplies the entire rate by only one compartment which contains the reactants. To see exactly what compartment volumes are used for scaling, open the “Equations View” on page 1-49 and check the **ODEs** section.

Units Library

It provides a collection of units that you can use. The library displays the unit composition for each unit. The **Menu Display** column controls how each unit is displayed when selecting quantity units in the Table on page 1-47 or Diagram on page 1-25 views. For instance, suppose you have a custom unit with the unit composition `mole/(liter*second)`. Depending on what you select in the **Menu Display**, SimBiology displays it differently in the **Units** column of Table Overview.



Unit Prefixes Library

It provides a list of all the available unit prefixes.

Blocks Library

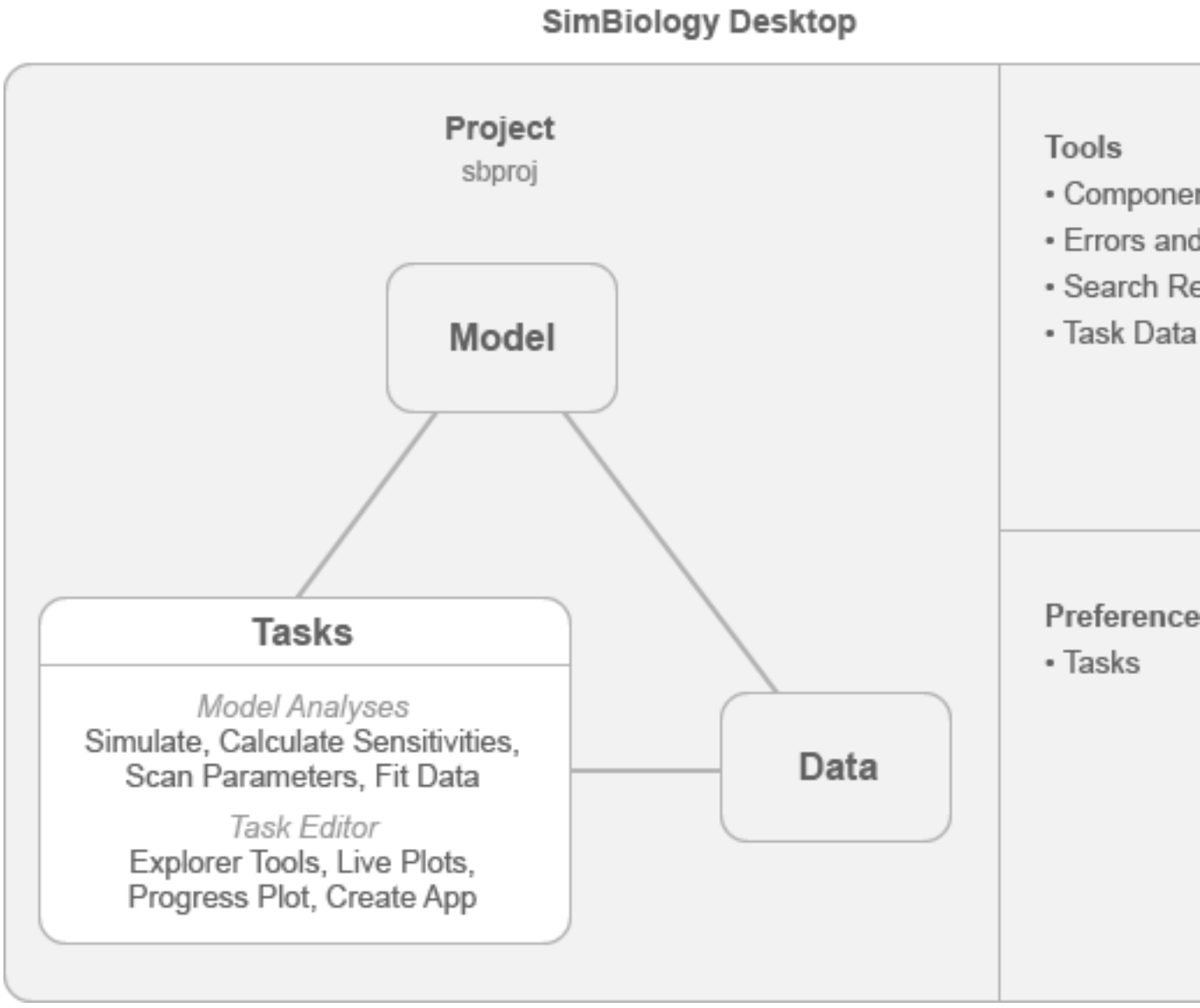
It contains blocks that you can use to build models interactively. SimBiology provides a default block library called **ModelBuilding** that contains built-in blocks representing all modeling elements. In addition to built-in blocks, you can create custom blocks with different appearances. For instance, you can customize a receptor protein to have a different block shape or color than other species. You can use these blocks to build a model graphically in the diagram view on page 1-25. For details about different built-in block types, see Built-in Blocks on page 1-43.

See Also

More About

- “General Workflow” on page 1-2
- “Modeling Workflow” on page 1-22
- “Analysis Workflow” on page 1-60
- “Setting Preferences” on page 1-54

Analysis Workflow



SimBiology lets you analyze models of dynamic systems. For instance, you can simulate various biological systems such as signaling pathways and explore what-if hypotheses. You can investigate system dynamics and guide experimentation using parameter sweeps and sensitivity analysis. Various dosing regimens can be evaluated to assess different combination therapies. In addition, you can use experimental time course data to estimate model parameters using nonlinear regression or mixed-effects techniques.

To perform these analyses, the SimBiology desktop provides built-in MATLAB scripts with a user interface called *tasks*. Each task is open and displayed in the task editor, where you can configure the task's settings such as specifying how long the simulation runs. The desktop automatically checks for errors and warnings as you configure the task and flags any issues using message indicators.

As a task is running, it displays a plot that shows the values of model quantities over time. The plot is updated live as the model is simulated. You can also add experimental data to the plot to compare with simulation results.

The desktop provides interactive model exploration tools that let you vary parameter values, initial conditions, and dosing schedules. For instance, you can simulate the model and interactively vary the value of a parameter of interest. The live plot automatically updates for the change, and you can check the plot to see if the simulation result using the new parameter value is close to experimental data. Then you can use the new value as an initial estimate when you perform parameter estimation.

Once the task finishes, you can visualize the results using different MATLAB plots. The desktop provides a default set of plots and selects the plots that are appropriate for the specific task being run. You can add additional plots to gain more insight into your results. You can export the results to MATLAB workspace or share with others by saving the results in a MAT or Excel® file.

The desktop provides the equivalent MATLAB script for each task. You can use these scripts as templates and modify them as necessary. For instance, you can modify the parameter scan task for more flexible sampling options using other MATLAB functions. You can write such scripts from within the task editor by creating a custom task, which is saved with the project.

You can also create standalone applications for model distribution and simulation. Suppose you want to share your model and analysis with non-modelers using a simple standalone application. You, as a modeler, can hide the details of the model and decide what model information to expose to other non-modelers. With the desktop you can build such an application that end users can use to vary model parameters, modify dosing

schedules, visualize the dynamics of key response variables, and save the results. For details, see “Deployment”.

See Also

More About

- “General Workflow” on page 1-2
- “Modeling Workflow” on page 1-22
- “Task Editor” on page 1-63
- “Configuring Tasks” on page 1-70

Task Editor

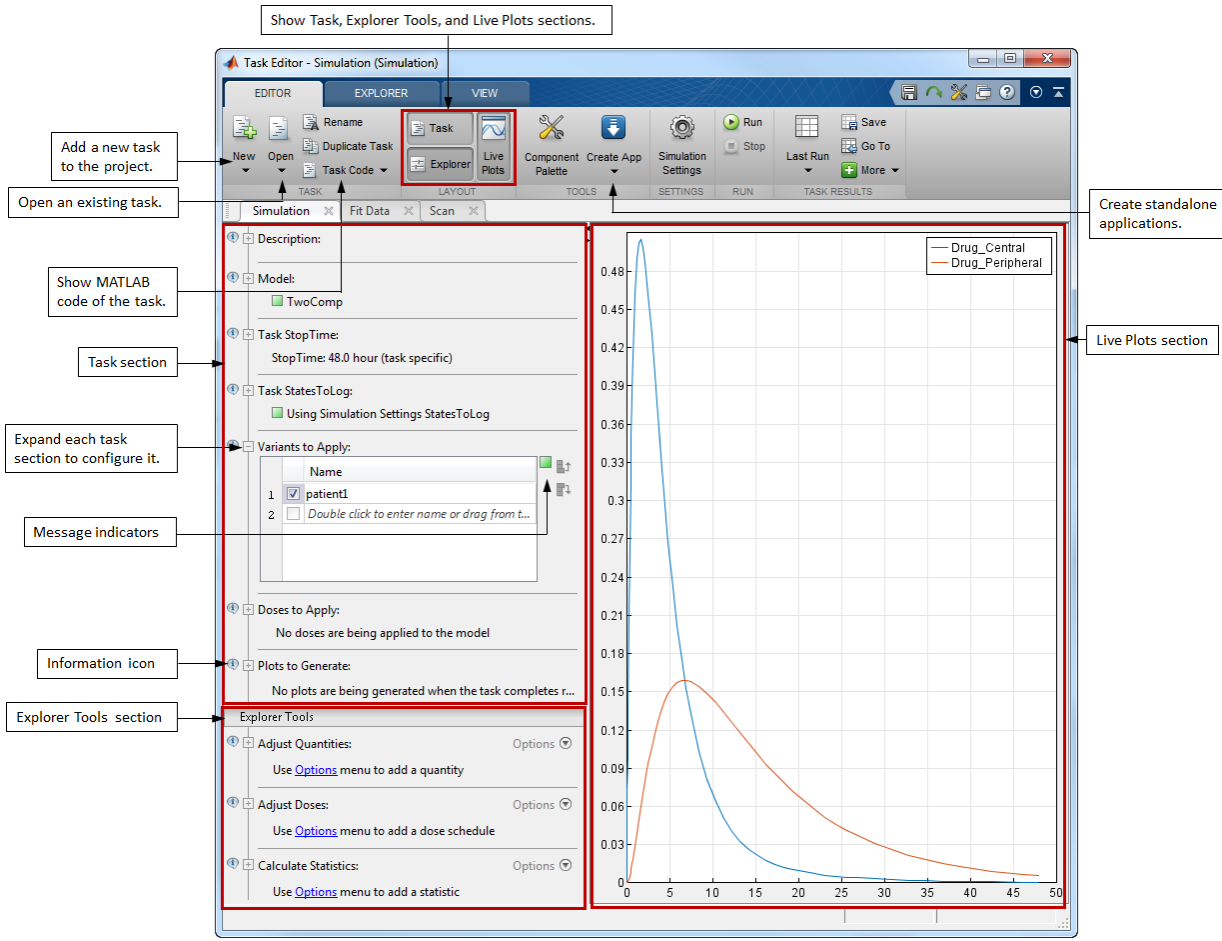
Each task on page 1-66 in the SimBiology desktop opens within the task editor, where you can edit the task settings, explore model parameters, and visualize live simulation results. If there are multiple tasks, each task opens in its own tab within the editor. The editor is composed of the **Task**, **Explorer Tools**, and **Live Plots**. You can display each of them by selecting the appropriate toggle button in the **Editor** tab. The tab also contains other options to add new tasks, view equivalent MATLAB code of the task, create standalone applications, and run the task.

The **Task** gives you a graphical way to define what is executed when you run a task. Suppose you have a variant that has an alternate set of parameter values for a cancer patient and you want to simulate the model using those values. You can specify the variant in the **Variants to Apply** section of a simulation task and run it. For more details about each section of a task, you can hover over the information icon to see its context-sensitive help. As you configure the task, the desktop checks for any errors or warnings and flags them using message indicators. For details, see “Configuring Tasks” on page 1-70.

The **Explorer Tools** let you explore parameter values, initial conditions, and dose schedules with sliders. These interactive tools let you quickly iterate through values without modifying the base model. For instance, you can vary a model parameter value and find out an initial value that generates simulation results closer to experimental observations. Then you can use the value as an initial estimate when you perform parameter estimation. You can also calculate statistics on simulation results using mathematical expressions to get better insight into the results. For instance, if you want to calculate an AUC (area under curve) of the concentration-time curve of a *drug*, you can write an expression such as `trapz(`*`drug`*`)`.

The **Live Plots** show the live simulation results as the task is running. You can also add experimental data points to compare with simulation results or define a mathematical expression to plot by right-clicking the live plot.

The tools for working with **Live Plots** and **Explorer Tools** are also listed in the **Explorer** tab. For instance, you can select which quantities or doses to explore, overlay simulation results in the live plot, or add new plots.



See Also

More About

- “Analysis Workflow” on page 1-60
- “Built-in Tasks” on page 1-66
- “Configuring Tasks” on page 1-70

- “General Workflow” on page 1-2
- “Modeling Workflow” on page 1-22

Built-in Tasks

The desktop provides the following built-in tasks that you can use to analyze models.

Task Name	Description	Shows Live Plots	Supports Deployment	Command-line Function
Simulate model	Simulates the dynamic behavior of a given model using a variety of deterministic and stochastic solvers.	Yes	Yes	`sbiosimulate`, `SimFunction` object
Fit data	Estimates model parameters by fitting the model to time-course data using nonlinear regression or nonlinear mixed-effects (NLME) techniques. You can fit data to a single individual to predict group-specific values or simultaneously fit all groups (pooled fit) to estimate a single set of values. **Note** For nonlinear mixed-effects (NLME) estimation, Statistics and Machine Learning Toolbox™ is required. For nonlinear regression, Optimization Toolbox™, Global Optimization Toolbox, and Statistics and Machine Learning Toolbox are recommended.	No. The progress plot on page 4-61 is shown instead.	No	`sbiofit` for nonlinear regression, `sbiofitmixed` for NLME modeling
Calculate conserved cycles	Calculates a complete set of linear conservation relations for species in a model and returns a list of species that are conserved in the system, regardless of reaction rates.	No	No	`sbioconsmoiety`

Task Name	Description	Shows Live Plots	Supports Deployment	Command-line Function
Calculate sensitivities	Computes the time-dependent derivatives of one or more species relative to either model parameters or species initial conditions. Calculating these time-dependent sensitivities helps you determine what effect the parameters or species have on another species or parameter.	Yes	No	`sbiosimulate`, `SimFunctionSensitivity` object
Run scan	Performs a parameter scan that lets you explore the model dynamics given different values of model quantities and doses. **Note** Statistics and Machine Learning Toolbox is recommended for more sampling options.	Yes	No	`sbiosimulate`, `SimFunction` object
Run scan with sensitivities	Performs a parameter scan while calculating sensitivities of model quantities. For each scanned value, you can see the corresponding time-dependent sensitivities of quantities with respect to the parameter of interest. **Note** Statistics and Machine Learning Toolbox is recommended for more sampling options.	Yes	No	`sbiosimulate`, `SimFunctionSensitivity` object

Task Name	Description	Shows Live Plots	Supports Deployment	Command-line Function
Run ensemble simulation	Performs a series of stochastic simulations of a model. When the behavior of a model is stochastic in nature, a single simulation run may not provide enough insight into the model. Use this task to perform a number of simulations.	Yes	No	`sbioensemblerun`
Run group simulation	Simulates a model for each group in the data. For example, the grouped data can contain measurements of drug plasma concentration at different times for multiple patients and dosing information for each patient. This task performs a simulation for each patient using the corresponding dosing information from the data and compares the simulation results to the data visually in the live plots section.	Yes	No	`sbiosimulate`, `SimFunction object`
Create custom analysis	Lets you create a task for custom analysis using MATLAB language. For instance, you can write a script to identify the optimal dosing strategy that suppresses tumor growth while satisfying safety constraints.	No	No	—
Search model(s)	Lets you search models for keywords.	No	No	`sbioselect`
Generate report	Creates a report of the model and analysis results. You can select various information to be included in the report such as the diagram representation of the model, model equations, or imported data.	No	No	—

See Also

More About

- “Analysis Workflow” on page 1-60
- “Task Editor” on page 1-63
- “Configuring Tasks” on page 1-70

Configuring Tasks

For each task, you can configure model-related settings such as the model to simulate, simulation-related settings such as the simulation stop time, and task-specific settings such as values to scan in a parameter scan task.

As you configure these settings, the desktop checks for errors and warnings and flags them using message indicators. You can hover over the indicators to get more information about the errors or warnings. More information about each section is described in the context-sensitive help of each section. You can open it by hovering over the information icon next to each section.

When you are selecting model elements such as parameters to estimate or doses to use during the analysis, you can use the **Component Palette** tool that provides a complete list of model quantities, doses, and variants. You can open the tool from the **Editor** tab and drag and drop model elements onto corresponding task sections. You can also use the down arrow key from within the table of each section to select model elements. In some of the tables, each row has a check box in the first column. When the box is checked, the corresponding row item is active and used during the analysis. You can use the context menu of each table for more options such as showing a quantity in the diagram view on page 1-25.

Configuring Model-Related Settings

Each task lets you specify a model to analyze. If the model simulation is expected to take a long time or the task is going to simulate the model multiple times (such as the Run Scan task), you can accelerate the model to improve performance. You can enable the acceleration by selecting the check box in the **Model** section.

You can also specify which doses and variants to use during the model analysis. Suppose that you have two different sets of parameter values for the normal and cancer patients stored in separate variants. You can specify which variant to use during simulation by selecting it in the **Variants to Apply** section of the task. Similarly, different doses can be selected in the **Doses to Apply** section to evaluate various dosing regimens or combine dosing schedules to assess different combination therapies. You can also generate variants with final quantity values after the model simulation or parameter estimation by selecting the corresponding option in the **Variants to Generate** section of the Simulation or Fit Data task.

Configuring Simulation-Related Settings

Each model has the default simulation settings associated with it. These settings include simulation time options, solver options, compile options, and data logging options. You can access these options by selecting **Simulation Settings** from the **Editor** tab of the task editor. When you change these simulation settings, it affects every task that is using the same model.

It is possible to overwrite some of the default simulation settings for certain tasks. These settings are the simulation stop time, states to log, solver type, and log decimation. In the corresponding section of a task, you can choose to use either the simulation settings value or a custom value specific for the task only. If you select the custom value option, that value is applied to the current task only and no other tasks.

Configuring Task-Specific Settings

Some of the tasks have unique settings that must be configured before the tasks can be run on page 1-76.

Fit Data

This task lets you estimate model parameters by fitting the model to experimental time-course data, using either nonlinear regression or nonlinear mixed-effects (NLME) methods.

Consider grouped data containing measured drug concentrations at different times for multiple individuals. You can estimate parameters for each individual or simultaneously fit all individuals to estimate a single set of values. Select the **Pool data** check box in the **Estimation Method** section to estimate one set of parameter values for all individuals. This option is available for all methods except for the mixed-effects methods (`nlmefit` and `nlmefitsa`).

In the **Estimated Parameters** section, you can select which parameters to estimate and specify parameter transformations as needed. For example, some parameters such as compartment volume and clearance are positive physical quantities, and log transformation reflects the underlying physical constraint and generally improves fitting. Use `logit` or `probit` transforms for parameters that have values from 0 through 1, such as bioavailability. You can also specify the lower and upper bounds for each estimated parameter for some of the estimation methods. For a list of methods that supports parameter bounds, see “Supported Methods for Parameter Estimation” on page 4-57.

If your data contains any dosing information such as dose amount for each patient at each dose time, use the **Dosing Information** section to define the mapping between the dose column of the data and the corresponding model species that is being dosed. In the table of the section, select the name of the dose variable (**Dose Column Name**), the dosed species (**Dose Component Name**), and the type of dose on page 5-28 (**Dose Configuration**).

You can map the measured or observed response data column (dependent variable) to the corresponding model quantity in the **Response and Error Model Information** section. For cases of multiple responses, SimBiology lets you specify an error model for each response or one error model for all responses. There are four error models on page 4-60, namely, constant, proportional, combined, and exponential. For a list of methods that support multiple error models, see “Supported Methods for Parameter Estimation” on page 4-57. In addition to these error models, you can also specify weights for each response.

You can also customize some of the common settings of the selected estimation method in **Algorithm Settings**. For instance, you can increase the maximum iterations if the algorithm fails to converge within the default limit. You can specify additional algorithm settings in **Advanced Algorithm Settings**. For example, if you want to use the Levenberg-Marquardt algorithm for the `lsqnonlin` method, enter `Algorithm = 'levenberg-marquardt'`. To see a complete list of all options for the selected estimation method, click the hyperlink provided in the section.

For an illustrated example of fitting PK profile data using a least-squares method, see “Estimate Pharmacokinetic Parameters Using SimBiology Desktop”.

For mixed-effects problems on page 4-42, SimBiology lets you estimate population parameters (fixed effects) while considering individual variations (random effects) using `nlmefit` or `nlmefitsa` estimation methods (Statistics and Machine Learning Toolbox is required). Consider grouped data containing measured drug concentrations at different times for multiple individuals. The objective is to estimate population PK parameters, such as volume of the central compartment *Central* and clearance *Cl*, and the random effect of each individual. For the *i*th individual, the mixed-effects model can be described as $Central_i = \theta_1 + \eta_i$ and $Cl_i = \theta_2 + \eta_i$, where θ_1 and θ_2 are fixed effects and η_i is the random effect of the *i*th individual. Random effects are assumed to be multivariate normally distributed $\eta_i \sim N(0, \Psi)$, where Ψ is the covariance matrix of random effects.

SimBiology represents the model as `Central = theta1+eta1` and `Cl = theta2+eta2` in **Estimated Parameters**. The drop-down menu of the **Expression** column displays a

list of available expressions for each parameter. You can also enter your own expression, but the fixed effect names must always start with `theta` and random effect names must start with `eta`.

You can define the structure of Ψ in **Covariance Matrix Pattern of Random Effect Parameters**. Each check box indicates a variance or covariance parameter that is being estimated. By default, SimBiology assumes no covariance among random effects, that is, uses a diagonal covariance matrix.

You can also specify individual-specific covariates such as patient weight that linearly relate to an estimated parameter in the **Covariates** section. In the table of the section, select the name of covariate column from the data. SimBiology allows centering of covariates to improve interpretability of the model. For instance, you may want to mean center the weight of each patient to help interpret the fixed effects and compare results with and without the covariate. If there are multiple covariates, you can standardize each of them by using an appropriate scaling method that may help you compare these covariates and select some of them.

Once you have defined a covariate, the task automatically updates the expression list for each parameter in **Estimated Parameters** to include additional parameter-covariate relationships, such as `Cl = exp(theta2+theta3*tWeight+eta2)`. *theta3* is the fixed effect of weight on *Cl* and *tWeight* is the (transformed) weight. For details, see `CovariateModel`.

After the completion of the task, you can generate a variant that contains model quantities with final estimated values. For an unpooled fit, that is, estimating one set of parameter values for each individual or group, you can generate group-specific variants, meaning one variant for each group. You can also generate a variant that contains the mean estimated values (averaged across all groups). You can select the corresponding option in the **Variants to Generate** section of the task.

Calculate Sensitivities

This task helps you investigate parameter effects on system dynamics. It lets you calculate local, time-dependent sensitivities on page 4-27 of one or more species with respect to parameter values and species initial conditions. Suppose that you want to calculate the sensitivity of a receptor protein with respect to a model parameter to see if the parameter has any influence on the receptor dynamics. You can specify the receptor species as the sensitivity output (numerator) and the parameter as the input (denominator) in the **Sensitivities to Compute** section. SimBiology lets you specify species, parameters, and constant compartments as inputs (and species and parameters

as outputs) for sensitivity calculation. The computed sensitivities can be normalized by selecting the appropriate method in the **Normalization** section. For instance, if you want to normalize with respect to the sensitivity output only, select the **Half** normalization. Select **Full** to make the data dimensionless. For details, see `Normalization`. You cannot run the sensitivity analysis task on models that contain events, algebraic rules, or non-constant compartments. For an illustrated example, see “Identify Important Network Components from an Apoptosis Model Using Sensitivity Analysis” on page 4-35.

Run Scan

This task lets you explore how a model behaves with different quantity values or repeat dose information, namely, dose start times, amounts, rates, and intervals. The task simulates a model multiple times, each time using different values for those quantities or doses of interest. Suppose you want to explore how varying the value of a forward rate parameter affects the final concentration of a product species. You can specify the parameter in **Values to Scan**. Use the **Values to Scan Defined With** section to specify what values to generate for scanning. You can define the values using custom MATLAB code. Alternatively, if you have Statistics and Machine Learning Toolbox, you can generate values from a multivariate normal distribution or using Latin hypercube sampling.

Run Ensemble Simulation

This task performs multiple simulations of a model using a stochastic solver on page 4-10. It lets you compare and analyze fluctuations in the behavior of a model over repeated stochastic simulations. Because stochastic simulations rely on an element of probability, sequential runs produce different results. Therefore, multiple stochastic runs are often needed to determine the probability distribution of the simulation results. Use the **Number of Runs** section to define the total number of stochastic simulations. If you want all the runs to have a consistent time vector, the data must be interpolated using the linear or zero-order hold method specified in the **Interpolation** section. By default, the task saves the time and quantity data of each state at each simulation time step. You can record the data less often by increasing the value of `LogDecimation`.

Run Group Simulation

This task lets you simulate each group or patient from grouped data. Suppose that the data contain measurements of drug plasma concentration at different times for multiple patients and dosing amount for each patient. You can use this task to simulate each patient and compare the results to the experimental data.

In the **Map Between Data and Model** section, you must specify a grouping variable, an independent variable, and a dependent variable (response). Map at least one response data column (`Dependent1`) to the corresponding model quantity. Similarly, you can map any dose column (`Dose1`) to the corresponding model species that is being dosed.

Once you start running the task, it applies any specified dose to the corresponding species, and simulates each group. It plots the response data column against the simulated values for the corresponding model quantity. As you compare the simulation results to experimental data, you can further explore the model behavior under different parameter values or species initial conditions using the **Explorer Tools**.

See Also

More About

- “Analysis Workflow” on page 1-60
- “Task Editor” on page 1-63
- “Built-in Tasks” on page 1-66
- “Running Tasks” on page 1-76

Running Tasks

The SimBiology desktop lets you analyze models of dynamic systems using tasks. Each task is a MATLAB script with a user interface that performs an analysis on the model, such as simulation, parameter estimation, or sensitivity calculation. You can configure on page 1-70 a task in the Task Editor on page 1-63. As you configure the task, the SimBiology desktop updates the corresponding script automatically and lets you view the task code.

For most tasks, the desktop shows simulation results in the **Live Plots** area of the Task Editor on page 1-63 while the tasks are running. There are certain tasks that do not show any plots. For tasks that support **Live Plots**, the desktop shows a time plot by default, and the results are updated live. Certain tasks show additional plots, such as a scan or sensitivity plot. You can customize these plots. For instance, you can add experimental data points to compare with simulation results, or select only a few quantities of interest to plot. For details, see “Configuring Live Plots” on page 1-76.

You can further explore model behavior using the **Explorer Tools**, which let you alter parameter values, initial conditions, and dose schedules. Each time you alter the value of a quantity, the task automatically reruns using the new value, and you can see the updated results in the **Live Plots** area. For details, see “Exploring Models” on page 1-79.

The desktop saves the task results as `Last Run (`*`task name - model name`*`)`. Each time you run the same task, the results are overwritten. You can save the results under a different name to avoid overwriting them. The **Explorer** tab of the Task Editor has options to save and export simulation data, or navigate on page 1-5 to the **Data** panel. For details, see Task Results on page 1-86.

Configuring Live Plots

For most tasks, a time plot of all quantities is shown by default. Depending on the task, the desktop shows additional plots such as sensitivity or scan plots or may not show any plot. For those tasks that support **Live Plots**, you can add more plots by first turning on **Live Plots** on the **Editor** tab and then selecting **Add Plots** from the **Explorer** tab.

You can configure what is plotted in each plot. Suppose that you are interested in selecting model quantities that are plotted in the **Live Plots** area after a task completes. Right-click the plot, and select **Define States to Plot > Plot State Data > Select**.

The desktop also lets you define a mathematical expression to plot by selecting **Define Math Data to Plot**. The expression must be a MATLAB expression that uses any of the legend names or time as variables. The expression must evaluate to a numeric value. For instance, suppose that you have concentration–time curves of two species with the legend names *s* and *sP*, where *s* corresponds to the concentration of an unphosphorylated protein, and *sP* corresponds to the phosphorylated concentration. You can add a plot to see the fraction of phosphorylated protein by entering `sP./(s+sP)` as an expression, where `s+sP` represents the total concentration of the protein. Another example would be to plot the amount of a species in terms of concentration. To do so, first add a plot entry for the species *S* and another for the compartment *V* via **Define States of Plot**. Then define the mathematical expression `S./V`, which plots the species concentration instead of amount.

You can add experimental data to the time plot to compare with the simulation results by selecting **Plot External Data** > **Add**. You can select a data set that is in the MATLAB workspace or load one from a file.

To configure axes and line properties of a plot, select **Properties** from the context menu of the plot. For instance, you can specify a linear or log scale for the *x*- or *y*-axis of each plot.

To highlight a line or data point of a simulated quantity in the plot, you can click its name in the legend. You can also click a line or data point, and the corresponding legend name is highlighted. Use this highlighting feature to look at the simulation data of a specific quantity among many others, or to see which quantity corresponds to the simulation data that you are interested in.

You can overlay results from each task run to compare simulation data. Select **Overlay Results** from the **Explorer** tab. Then every time you run the task, the desktop plots the results from the current run on top of the results from previous runs. For example, suppose that you can have a set of values for immunological parameters of a healthy person represented by a variant and a different set of values for a cancer patient represented by another variant. You can then simulate the model with each variant and overlay the predictions for both cases to see the changes in model behavior.

Most tasks show one or more plots while they are running. Details on the plots for the most common tasks are described in the following sections.

Simulation

This task shows a time plot of all simulated quantities. You can disable the automatic creation of the time plot when you run the task in the **Preferences > Tasks > Create line plot in Task Explorer on Simulation Task run**.

Calculate Sensitivities

This task shows a time plot and a sensitivity plot. The sensitivity plot is shown after the task finishes running. The plot displays the time-dependent sensitivities to parameter values and initial conditions as a bar graph. Use the context menu of the plot for more options, such as sorting the values in an ascending or descending order. You can disable the automatic creation of the sensitivity plot by clearing **Preferences > Tasks > Create sensitivity plot in Task Explorer on Sensitivity Task run completion**.

Run Scan

This task shows a scan plot and a time plot. For each parameter scan, the desktop performs a mathematical evaluation on the simulation data and returns a scalar value. The scan plot plots this value versus the scan parameter. You can add more scan plots by selecting **Add Plot > Scan Plot** on the **Editor** tab. Then you can define the mathematical expression to evaluate by selecting **Plot Math Data > Add**.

You must define the expression as `out = expression`, and the expression must return a scalar. In the scan plot, the evaluated results (*out*) are plotted on the *y*-axis and the scan parameter on the *x*-axis. If you have multiple scan variables, you can select which scan parameter is plotted on the x-axis. In the expression, you can refer to any quantity by its name. If the name of a quantity is not a valid MATLAB variable name, enclose it in square brackets, such as `[DNA polymerase+]`. To check if the expression has any errors or warning, click **Verify**.

You can disable the automatic creation of the scan plot by clearing **Preferences > Tasks > Create scan plot in Task Explorer on Scan Task run**.

As an illustration, the scan plot shows the maximum value of the first quantity in the time plot versus the scan variable. You can write custom MATLAB code to plot. For instance, suppose *Drug_Central* is the name of a species that represents the concentration of the drug in the system, and *time* is the simulation time variable. You can plot the time point at which the drug concentration is at its maximum using the following expression.

```
out = time(Drug_Central == max(Drug_Central));
```

Run Scan with Sensitivities

This task combines the **Calculate Sensitivities** and **Run Scan** tasks. It shows a scan plot and time plot. As an illustration, the scan plot shows time-dependent sensitivities of the first quantity with respect to the scan parameter. For details on how to configure a scan plot, see “Run Scan” on page 1-78. You can disable the automatic creation of the scan plot by clearing **Preferences > Tasks > Create scan plot in Task Explorer on Scan with Sensitivities Task run**.

Run Group Simulation

This task shows a trellis time plot, where each subplot represents a group or an individual. The plot shows the response data against the simulated values for each group. If you have multiple responses, you can select responses to plot by selecting **Edit Properties** on the **Explorer** tab.

Fit Data

Instead of **Live Plots**, this task shows the **Progress Plot** that provides live feedback on the status of the fit, except when using `nlinfit` as the estimation method. For instance, it displays fitting quality measures, such as the log-likelihood, and estimated parameter values for each function iteration. The **Progress Plot** opens in a new figure window. This task does not support plots in the **Live Plots** area or the **Explorer Tools**.

To turn off the **Progress Plot**, clear **Show progress of the Fit Data task** in the **Algorithm Settings** section. For details, see “Progress Plot” on page 4-61.

Exploring Models

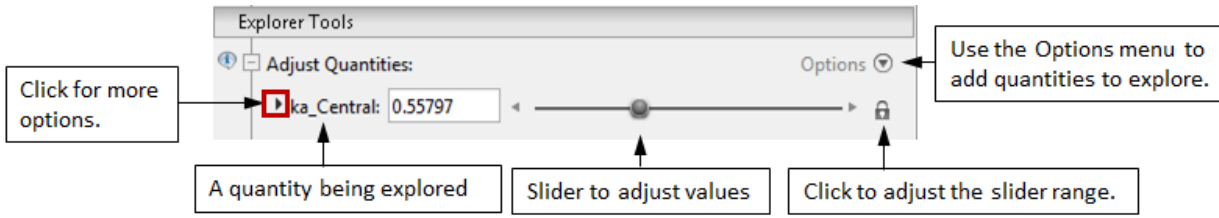
The SimBiology desktop provides explorer tools that let you iterate through different parameter values, initial conditions, and dosing schedules without modifying the base model. When you change a value, the desktop automatically reruns the task using the new value, and the results are updated in the **Live Plots**. You can overlay results from each run to see the changes in model behavior. For a summary of quantity values used for each run, open the **MultiRun Viewer** from the **Explorer** tab. You can also calculate statistics, such as the area under the curve of a drug concentration profile, to gain additional insight.

You can turn the **Explorer Tools** on and off by selecting **Explorer** on the **Editor** tab. If you turn it off, the desktop does not use the values defined in the tools during the task run.

Adjust Quantities

You can explore how changes to quantity values affect the model behavior. To add a quantity to explore, select **Define Quantities to Adjust** from the **Options** menu. The desktop adds a slider for you to change the quantity value. Each time you move the slider, the desktop reruns the task using the new slider value by default. You can overlay the simulation results from each run and compare them.

Use the **Options** menu to configure the slider behavior and properties. For instance, to stop the task from being run each time you move the slider, clear **Run Task When Quantity Changes Value**. To change the range of the slider, select **Define Slider Properties**.

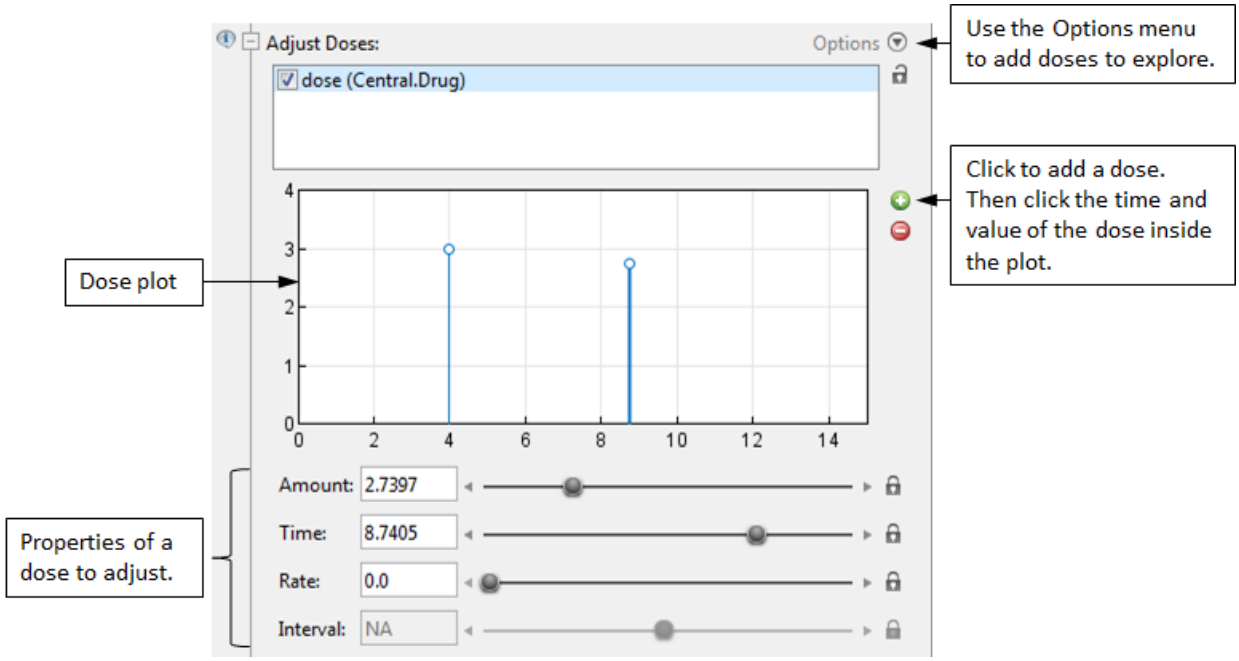


Adjust Doses

An increase in a species amount or concentration due to an external stimulus, such as an oral or intravenous administration of a drug, can be modeled using an element called a dose. The **Adjust Doses** tool lets you create doses to explore different dosing regimens and their effects on the model behavior.

To add a dose to explore, select **Create Dose Schedule to Explore** from the **Options** menu. A dose plot is shown along with the dose properties that you can adjust. To add doses at specific times and values, first click the green plus button on the right of the plot, and then click inside the plot. You can use the sliders to adjust the dose properties such as dose amount or dose time. To remove a dose, select the corresponding line in the plot. Then click the red minus button on the right of the plot. Any unchecked dose is not applied to the model during the task run.

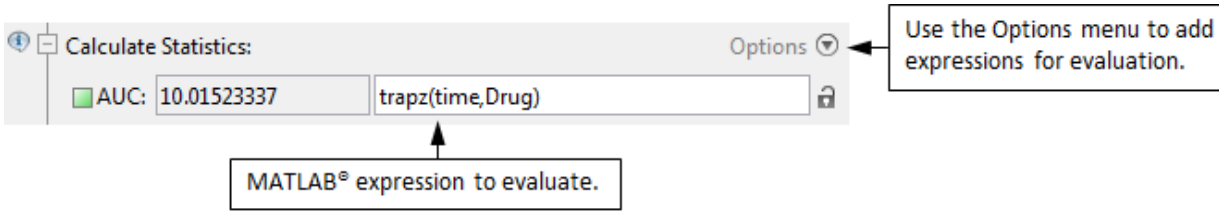
You can also explore an existing dose by selecting **Choose Existing Dose to Explore**. When exploring, the desktop does not change the existing dose properties. To avoid double dosing, do not select the same dose in the **Doses to Apply** section of the task.



Calculate Statistics

You can evaluate a mathematical expression on the simulation results. The expression can be any valid MATLAB expression that returns a numeric value. In your expression, you can refer to any quantity by its name. If the name of a quantity is not a valid MATLAB variable name, enclose it in square brackets such as `[DNA polymerase+]`. You can also refer to the simulation time variable as *time*. To add an expression, select **Define Statistics to Calculate** from the **Options** menu. For instance, to calculate the AUC (area under the curve) of the concentration-time curve of a drug, use the expression `trapz(time,Drug)`.

For a task that runs multiple simulations, such as the **Run Scan** task, the desktop evaluates the expression at the end of each simulation. Open **MultiRun Viewer** to view all the calculated results.



See Also

More About

- “Configuring Tasks” on page 1-70
- “Task Editor” on page 1-63
- “Built-in Tasks” on page 1-66
- “Analysis Workflow” on page 1-60

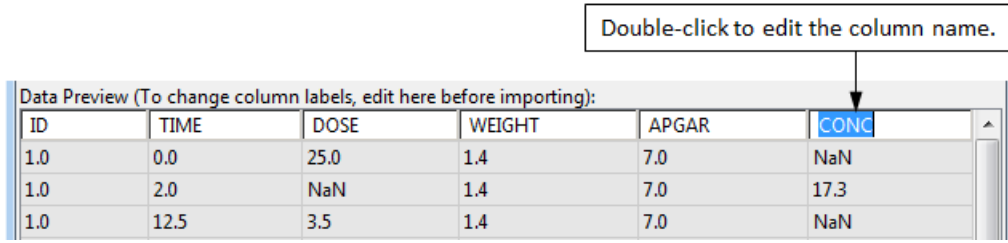
External Data and Task Results

SimBiology lets you build models of dynamic systems and perform model analyses using available research data. For instance, you can import time-course data from experiments, preprocess data, and use data in analysis tasks on page 1-66, such as parameter estimation. Task results are generated by each task after it finishes running, and consist of simulation data, summary of simulation options and task settings used during the task run, and plots for data visualization and exploration.

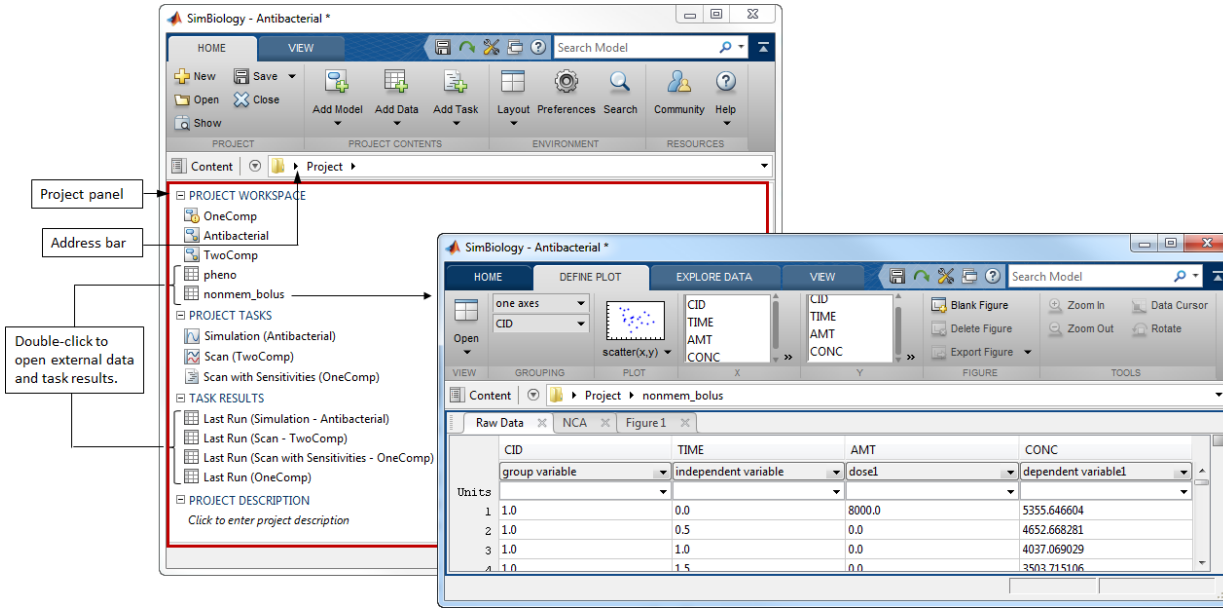
You can import data from several file formats, such as Excel files (.xls, .xlsx), text files (.csv, .txt), and SAS® XPORT files (.xpt). NONMEM®-formatted data are also supported, and SimBiology interprets the columns according to the NONMEM definitions. For details, see Importing Data-Supported Files and Data Types on page 5-6. You can also import grouped data which have multiple groups of observations, such as measured concentrations from multiple patients.

In addition, SimBiology lets you import variables from the MATLAB workspace. If variables have the same dimension, you can concatenate them horizontally by multiple selection (**Ctrl** + click or **Command** + click on a Mac).

You can edit column names while importing. After imported, the names are final, and SimBiology refers to the columns by those names in subsequent workflows such as plotting or fitting the data. To edit, double-click the name of a column in the **Data Preview** section of the **Import** dialog box.



Both external data and task results are saved in a SimBiology project (`*.sbproj`). They are accessible from the project panel in the workspace area on page 1-5 of the desktop. Select **Project** in the address bar and double-click the data set or task result to open it. The desktop then displays the corresponding contents in the workspace area. The next figure shows an example of the project workspace and an external data set when open.



You can visualize data using various built-in plots, such as time or scatter plots. First open the data and select the plot type from the drop-down menu on the **Define Plot** tab. Then define the corresponding input arguments for the plot. If you have grouped data, you can create a trellis plot where each group is plotted on a separate axis.

To add more plots, select **Blank Figure** on the **Define Plot** tab. Plots are shown as separate figure tabs in the workspace area. You can open and close the tabs. To reopen any closed tab, select it from the **Open** drop-down menu. To remove a figure from the project permanently, select **Delete Figure**.

The desktop provides built-in tools to help you explore data. For instance, you can take a closer look at a particular area of the plot by zooming in. To query the values of data points, select **Data Cursor** on the **Define Plot** tab and then click a data point.

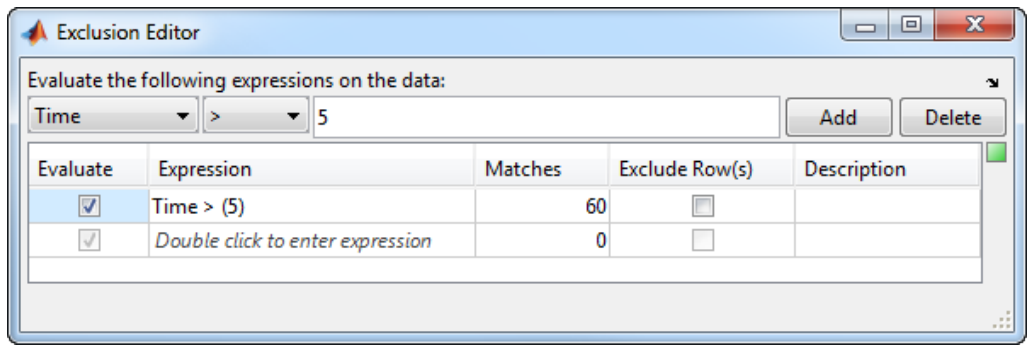
You can also export data to MATLAB workspace as separate variables, a `dataset`, or `SimData object`. In addition, you can export task results to a MAT or Excel file.

External Data

You can open any imported data set from the project panel. Each external data set appears as a table on the **Raw Data** tab in the workspace area. The table shows the

values of each data column. You can classify the columns as a group variable, independent variable, dependent variable, covariate, dose amount, and dose rate variables from the drop-down menu of each column. You can also specify the corresponding unit for each column. SimBiology uses these data classifications accordingly in analysis tasks. For instance, the Group Simulation task on page 1-74 uses the independent and dependent variables to plot the data while applying the dose amounts from the dose column to each group during simulation.

The desktop lets you preprocess data by excluding data points, such as outliers. Excluded data points are not used during model analyses such as data fitting. The excluded rows in the data are grayed out on the **Raw Data** tab. If you have any plots, these data points are labeled as **Excluded data**. To define exclusion rules or edit existing ones, select **Edit Exclusions** on the **Explore Data** tab and enter a valid MATLAB expression. You can refer to the column names in your expressions. Alternatively, you can use the drop-down menus from the exclusion editor to select a column and operator to define the expression. For instance, the next figure shows an expression rule to exclude any data points recorded after `Time > 5`.



You can transform data to make them easier to visualize or improve interpretability. To add a new column of derived data from one or more existing columns using a MATLAB expression, select **Edit Derived Data** on the **Explore Data** tab and enter the expression. For instance, if you want to log-transform the measured drug concentrations, enter `log(`*`Conc`*`)` where *Conc* is the name of the concentration column in your data.

The desktop also lets you compare time-course data to simulation results. To compare, add the data to the time plot of simulated quantities in the **Live Plots** area of the task editor on page 1-63. Right-click the time plot and select **Define External Data to Plot**.

If you have the time course of measured drug concentrations, you can calculate pharmacokinetic (PK) parameters of the drug using non-compartmental analysis (NCA). To see a table of PK parameters, select **Open > NCA** on the **Define Plot** tab. For details, see NCA on page 4-111.

Task Results

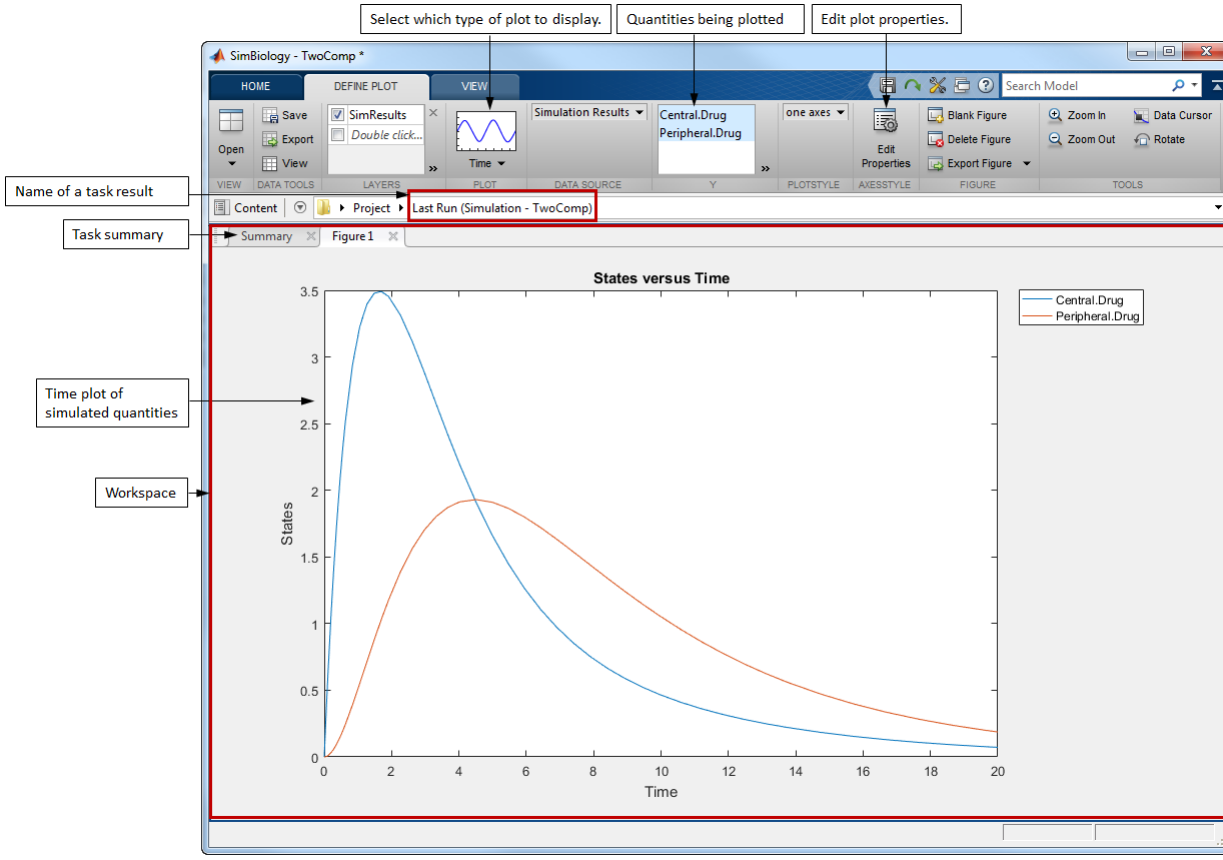
Task results are saved as `Last Run (`*`task name - model name`*`)` after each run. If you rerun the same task, the results are overwritten. To avoid overwriting them, save the results with a different name. Right-click the **Last Run** result of the task from the project panel in the workspace area and select **Save Data**.

When you open task results, the desktop shows the **Summary** tab and other figure tabs in the workspace area. The **Summary** tab contains the details of the task, such as simulation and compile options used during the task run. Generated plots, that is, plots specified in the **Plots to Generate** section of the task, are shown in separate tabs. To edit properties of each plot, such as changing *x*-axis or *y*-axis to a log scale, select **Edit Properties** on the **Define Plot** tab.

In addition to generated plots, you can add new plots to visualize the results in different ways and gain additional insight. For instance, you can explore the relationship between two variables by adding an **XY** plot or check the quality of a fit with additional residual plots.

To view or export simulation data, select **View** or **Export** on the **Define Plot** tab. You can export the results to the MATLAB workspace and other files, such as MAT or Excel files.

The next figure is an example of simulation task results from a two-compartment model showing the concentration–time profile of a drug in the *Central* and *Peripheral* compartments.



Tip You can navigate to the task results panel from the task editor on page 1-63 by selecting **Go To** on the **Editor** tab. You can also save the task results by selecting **Save**.

See Also

More About

- “Importing Data — Supported Files and Data Types” on page 5-6
- “Import Data from a NONMEM-Formatted File Using the SimBiology Desktop” on page 5-15

- “Analysis Workflow” on page 1-60
- “Task Editor” on page 1-63
- “Built-in Tasks” on page 1-66
- “Noncompartmental Analysis” on page 4-111

Modeling

- “What is a Model?” on page 2-2
- “Species Object” on page 2-6
- “Definitions and Evaluations of Reactions” on page 2-10
- “Definitions and Evaluations of Rules” on page 2-17
- “Events” on page 2-30
- “Variants” on page 2-40
- “Doses” on page 2-42
- “Simulate Biological Variability of the Yeast G Protein Cycle Using the Wild-Type and Mutant Strains” on page 2-47
- “Create and Simulate a Model with a Custom Function” on page 2-49
- “View Model Equations” on page 2-57
- “Component Usage” on page 2-58
- “Evaluation of Model Component Names in Expressions” on page 2-61

What is a Model?

In this section...
“Model Definition” on page 2-2
“Expressions” on page 2-2
“Quantities” on page 2-3
“Model Hierarchy” on page 2-4
“Representing a Model” on page 2-4

Model Definition

A SimBiology model is composed of a set of expressions (reactions, differential equations, discrete events), which together describe the dynamics of a biological system. You write expressions in terms of quantities (compartments, species, parameters), which are also enumerated in the model.

Expressions

There are three distinct types of expressions in SimBiology:

- Reactions
- Rules
- Events

Reactions

A reaction describes a process such as a transformation, transport, or binding/unbinding process between reactants and products.

Example reactions include:

```
Creatine + ATP <-> ADP + phosphocreatine
cytoplasm.speciesA -> nucleus.speciesA
```

Rules

A rule is a class of mathematical expressions that include differential equations, initial assignments, repeated assignments, and algebraic constraints.

For example, you can use a rule to:

- Specify values for model components that are required for comparison with experimental data. For example, specify the active fraction of total protein.
- Assign values to model components based on the values of other components in the model. For example, define a parameter's value as being proportional to a species or another parameter.
- Define mass balance equations.
- For species, use rate rules as an alternative to the differential rate expression generated from reactions.

Events

An event describes an instantaneous change in the value of a quantity (compartment, species, parameter). The discrete transition occurs when a user-specified condition becomes true. The condition can be a specific time or a specific time-independent condition.

For example, you can use an event to:

- Activate or deactivate a specific species (activator or inhibitor species)
- Change a parameter value based on external signals
- Change reaction rates in response to addition or removal of a species
- Replicate an experimental condition, such as the addition or removal of an activating agent (such as a drug) to or from a sample

Quantities

SimBiology uses three types of quantities in models:

- Compartments
- Species
- Parameters

Compartments

A compartment defines a physically bounded region that contains species. A compartment is characterized by a capacity expressed as volume, area, or length. A compartment can also contain other compartments, which adds hierarchy to a model. For example, a

compartment named `cytoplasm` might contain a compartment named `nucleus`, thereby partitioning species based on their location.

Species

A species characterizes the state of the biological system by representing the amount (or concentration) present in the system for that entity. Examples of species are `DNA`, `ATP`, and `creatine`. Species' amounts (or concentrations) vary during a simulation as a result of their participation in reactions, differential equations, and events. Therefore, species represent the dynamical state of a biological system.

Parameters

A parameter is a quantity that is referred to by expressions. It typically remains constant during a simulation. For example, parameters are used as rate constants in reactions.

You can configure a parameter to vary during a simulation. This is useful, for example, to model the change in a reaction rate given the concentration of a catalyst or a change in temperature.

Model Hierarchy

Note the following conditions imposed on quantities in the model hierarchy:

- Models must contain at least one compartment.
- A compartment can contain one or more compartments.
- Species are always contained within a compartment.

Representing a Model

In SimBiology, models and their components are implemented as objects. These objects have properties and methods that you can use to access and configure them. Use the `get` method to list the property values of an object. Use the dot notation to change the property values of an object.

SimBiology objects are handle objects, which has implications for how they behave during copy operations. Handle objects are referenced by their handle variable, and copies of the handle variable refer to the same object. To learn how handle objects affect copy operations, see Copying Objects (MATLAB).

See Also

`Model`

Species Object

A `species object` represents a species, which is the amount of a chemical or entity that participates in reactions. A species is always scoped to a compartment.

When adding species to a model with multiple compartments, you must specify qualified names, using *`compartmentName`*.*`speciesName`*. For example, `nucleus.DNA` denotes the species `DNA` in the compartment `nucleus`.

For information about...	See...
Creating and adding a species to a model	`addspecies`
Methods and properties of a species	`species object`

How Species Amounts Change During Simulations

The amount of a species can remain constant or vary during the simulation of a model. Use the following properties of a `species object` to specify how the amount of a species changes during a simulation:

- `ConstantAmount` property — When set to `true`, the species amount does not change during a simulation. The species can be part of a reaction or rule, but the reaction or rule cannot change its amount. When set to `false`, the species amount is determined by a reaction or a rule, but not both.
- `BoundaryCondition` property — When set to `true`, the species amount is either constant or determined by a rule, but not determined by a chemical reaction. In other words, the simulation does not create a differential rate term from the reactions for this species, even if it is in a reaction, but it can have a differential rate term created from a rule.

Keeping a Species Amount Unchanged

Set `ConstantAmount` to `true` and `BoundaryCondition` to `false` for a constant species, whose amount is not changed by a reaction or rule. In this case, the species acts like a parameter. It cannot be in a reaction, and it cannot be varied by a rule.

ConstantAmount	BoundaryCondition	Reaction	Rule	Changed By
True	False	No	No	Never

Example — Species **E** is not part of the reaction, but it is part of the reaction rate equation. **E** is constant and could be replaced with the constant `Vm = k2*E`.

```
     reaction: S -> P
reaction rate: kcat*E*S/(Km + S)
```

Changing a Species Amount with a Reaction or Rule

Set `ConstantAmount` to `false` and `BoundaryCondition` to `false` for a species whose amount is changed by a reaction or rule, but not both.

ConstantAmount	BoundaryCondition	Reaction	Rule	Changed By
False	False	Yes	No	Reaction
False	False	No	Yes	Rule

Example 1 — Species **A** is part of a reaction, and it is in the reaction rate equation. The species amount or concentration is determined by the reaction. This is the most common category of a species. A differential rate equation for the species is created from the reactions.

```
     reaction: A -> B
reaction rate: k*A
```

Example 2 — Species **E** is not part of the reaction, but it is in the reaction rate equation. **E** varies with another reaction or rule.

```
     reaction: S -> P
reaction rate: kcat*E*S/(Km + S)
```

Example 3 — Species **G** is not part of a reaction, and it is not in a rate equation. **G** varies with an algebraic rule or rate rule.

```
    rate rule: dG/dt = k
```

Changing a Species Amount with a Rule When Species is Part of a Reaction

Set `ConstantAmount` to `false` and `BoundaryCondition` to `true` for a species whose amount is changed by a rule, but the species is also part of a reaction, and a differential rate term from the reaction is not created. The amount of the species changes with the rule, and a differential rate term is created from the rule.

ConstantAmount	BoundaryCondition	Reaction	Rule	Changed By
False	True	Yes	Yes	Rule

Example 1 — Species **A** is not changed by the rate equation, but changes according to a rate rule. However, **A** could be in the rate equation that changes other species in the reaction.

```
     reaction: A -> B
reaction rate: k1 or k1*A
    rate rule: dA/dt = k2*A (solution is  A = k2*t)
               (enter in SimBiology as A = k2*A)
```

Example 2 — Species **A** is not in the rate equation, but changes according to an algebraic rule.

```
      reaction: A -> B + C
 reaction rate: k or k*A
algebraic rule: A = 2*C
                (enter in SimBiology as 2*C - A)
```

Keeping a Species Amount Unchanged When Species is Part of a Reaction that Adds or Removes Mass

Set `ConstantAmount` to `false` and `BoundaryCondition` to `true` for a constant species that is part of a reaction, but a differential rate term is not created from the reaction. The differential rate term is created from a rule.

ConstantAmount	BoundaryCondition	Reaction	Rule	Changed By
True	True	Yes	No	Never

During simulation, a differential rate equation is not created for the species. `dSpecies/dt` does not exist.

Example 1 — **A** is a *`infinite source`* and its amount does not change. B increases with a zero order rate (`k` and `k*A` are both constants). A source refers to a species where mass is added to the system.

```
     reaction: A -> B
reaction rate: k or k*A
```

Example 2 — B decreases with a first-order rate, but **A** is an *infinite sink* and its amount does not change. A *sink* refers to a species where mass is subtracted from the system.

```
     reaction: B -> A
reaction rate: k*B
```

Example 3 — The **null** species is a reserved species name that can act as a source or a sink.

```
     reaction: null -> B
reaction rate: k

     reaction: B -> null
reaction rate: k*B
```

Example 4 — **ATP** and **ADP** are in the reaction and have constant values, but they are not in the reaction rate equation.

```
     reaction: S + ATP -> P + ADP
reaction rate: Vm*S/(Km + S)
```

Definitions and Evaluations of Reactions

A reaction is a mathematical expression that describe a transformation, transport, or binding process that changes one or more species. Typically, an amount of a species is changed through a reaction.

In SimBiology, a reaction is represented by a `reaction object`, which has the following properties.

- Reaction property — Mathematical expression that describes the reaction
- ReactionRate property — Mathematical expression that defines the rate at which the reactants combine to form products. You can provide this information explicitly or use the `KineticLaw` property to populate this information.
- KineticLaw property — Object that specifies a rate law that defines the type of reaction rate. Examples include Henri-Michaelis-Menten and Mass Action. The object also specifies `species objects`, or `parameter objects`. This property is optional. It serves as a template for a reaction rate and provides a convenient way of applying a specific rate law to multiple reactions. If you use this property, it automatically populates the `ReactionRate` property.

A reaction is scoped to a model.

For information about...	See...
Creating and adding a reaction to a model	`addreaction`
Methods and properties of a reaction	`reaction object`
Creating and adding a kinetic law to a reaction	`addkineticlaw`
Methods and properties of a kinetic law	`KineticLaw object`

Writing Reaction Expressions

Use standard chemistry reaction notation to create the mathematical expression for a reaction (`Reaction` property of a `reaction object`).

Following are rules for writing reaction expressions:

- Use spaces before and after species names and stoichiometric values.

- Stoichiometry values must be positive.
- If a stoichiometry value is not specified, it is assumed to be `1`.
- In a model with a single compartment, specify species using *`speciesName`*. In a model with multiple compartments, specify species using qualified names: *`compartmentName`*.*`speciesName`*. For example, `nucleus.DNA` denotes the species `DNA` in the compartment `nucleus`.
- Enclose names with non-alphanumeric characters (including spaces) in brackets.
- Reactions can be reversible (`<->`) or irreversible (`->`).

Examples of reaction expressions include:

```
Creatine + ATP <-> ADP + phosphocreatine
glucose + 2 ADP + 2 Pi -> 2 lactic acid + 2 ATP + 2 H2O
cytoplasm.A -> nucleus.A
[compartment 1].[species A] -> [compartment 2].[species A]
```

Note Same species can be used multiple times in the list of reactions or products. The expression `'2 A'` is equivalent to `'A + A'`.

Writing Reaction Rate Expressions Explicitly

Use any valid MATLAB code to create the mathematical expression for a reaction rate (`ReactionRate` property of a `reaction object`). The reaction rate can specify compartments, species, or parameters.

Following are rules for writing reaction rate expressions:

- The expression must be a single MATLAB statement that returns a scalar.
- In a model with a single compartment, specify species using *`speciesName`*. In a model with multiple compartments, specify species using qualified names: *`compartmentName`*.*`speciesName`*. For example, `nucleus.DNA` denotes the species `DNA` in the compartment `nucleus`.
- Enclose names with non-alphanumeric characters (including spaces) in brackets.
- Do not end the reaction rate expression with any of the following:
  - Semicolon
  - Comma

- Comment text preceded by `%`
- Line continuations indicated by `...`

For example, if you have the following reaction expression:

```
Creatine + ATP <-> ADP + phosphocreatine
```

and the reaction follows Mass Action kinetics, then the reaction rate expression would be:

```
K*Creatine*ATP - Krev*ADP*phosphocreatine
```

Tip If your reaction rate expression is not continuous and differentiable, see "Using Events to Address Discontinuities in Rule and Reaction Rate Expressions" on page 2-38 before simulating your model.

Creating Reaction Rate Expressions Using Kinetic Law Objects

A `KineticLaw object` is scoped to a reaction and specifies:

- A rate law that defines the type of reaction rate. Examples include Henri-Michaelis-Menten and Mass Action.
- species and parameters

A `KineticLaw object` serves as a template for a reaction rate and provides a convenient way of applying a specific rate law to multiple reactions. You can use this object to create a reaction rate, which populates the `ReactionRate` property of the `reaction object`.

For example, if you create a `KineticLaw object` that specifies Henri-Michaelis-Menten for the `KineticLawName`, species `S`, and parameters `Vm` and `Km`, the reaction rate law is:

$$V_{\mathrm{m}} * S / (K_m + S)$$

Then if you create a `reaction object` that specifies the previous `KineticLaw object` and species the following reaction expression:

```
A -> B
```

with `Vm` = `Va` and `Km` = `Ka` and `S` = `A`, then the reaction rate equation is:

Va*A/(Ka + A)

Examples of Creating Reaction Rates

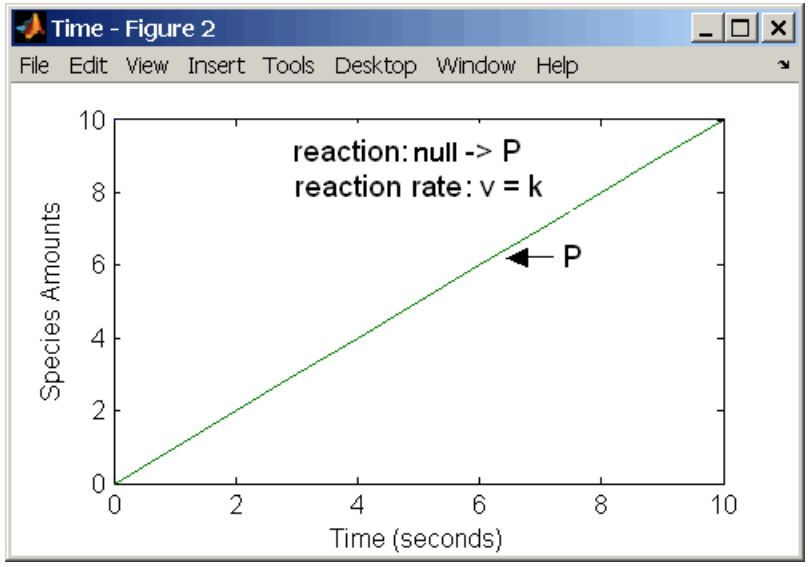
Example of Creating a Zero-Order Reaction

With a zero-order reaction, the reaction rate does not depend on the concentration of reactants. Examples of zero-order reactions are synthesis from a `null` species, and modeling a source species that is added to the system at a specified rate.

```
     reaction: null -> P
reaction rate: k mole/second
      species: P =  0 mole
   parameters: k =  1 mole/second
```

Note When specifying a `null` species, the reaction rate must be defined in units of amount per unit time not concentration per unit time.

Entering the reaction above into the software and simulating produces the following result:



Zero-Order Mass Action Kinetics

Note If the amount of a reactant with zero-order kinetics reaches zero before the end of a simulation, then the amount of reactant can go below zero regardless of the solver or tolerances you set.

Examples of Creating Other Reactions

For examples of creating other reaction rates, see “Define Reaction Rates with Mass Action Kinetics” on page A-2 and “Define Reaction Rates with Enzyme Kinetics” on page A-8.

How Reaction Rates Are Evaluated

Reaction Rate Dimensions

When calculating species fluxes, SimBiology must determine whether you specified reaction rates in dimensions of amount/time or concentration/time. When all compartments in a model have a capacity of one unit, amount and concentration are numerically equivalent.

For all other models, the numerical results of the simulation depend on which interpretation SimBiology selects. SimBiology determines whether a reaction rate is in dimensions of amount/time or concentration/time via dimensional analysis of `ReactionRate` expressions. This minimum level of dimensional analysis always occurs, even when `DimensionalAnalysis` and `UnitConversion` are off.

The `DefaultSpeciesDimension` property defines the dimensions of species appearing in a reaction rate. SimBiology infers the dimensions of parameters appearing in a reaction rate from their `ValueUnits` property. If any parameters appearing in a reaction rate expression do not have units, SimBiology interprets the reaction rate in dimensions of amount/time. Therefore, the only way to specify that a reaction rate has dimensions of concentration/time is to assign appropriate units to all parameters.

Reactions Spanning Multiple Compartments

Specify reactions that span compartments using the syntax *`compartment1Name`*`.`*`species1Name`* `->` *`compartment2Name`*`.`*`species2Name`*. The reaction rate dimensions must resolve to amount/time if either of the following conditions are true:

- Reactant species are in different compartments.
- The reaction is reversible mass action and the products are in multiple compartments.

Examples

Consider a reaction `a + b –> c`. Using mass action kinetics, the reaction rate is `k*a*b`, where `k` is the rate constant of the reaction. If you specify that initial amounts of `a` and `b` are `0.01 molarity` and `0.005 molarity` respectively, then the reaction rate is in concentration/time (and units of `molarity/second`) if the units of `k` are `1/(molarity*second)`. If you specify `k` with another equivalent unit definition, for example, `1/((moles/liter)*second)`, SimBiology checks whether the physical quantities match. If the physical quantities do not match, you see an error and the model is not simulated.

If, in the previous example, you specify that initial amounts of `a` and `b` are `0.01` and `0.005` respectively, without specifying units, SimBiology checks whether `DefaultSpeciesDimension` is `substance` or `concentration`. If `DefaultSpeciesDimension` is `concentration`, and you set units on the rate constant such that the reaction rate dimensions resolve to concentration/time, SimBiology scales the species amounts for compartment capacity, and returns the species values in concentration.

If you specify initial amounts of `a` and `b` as `0.01 molarity` and `0.005 mole` respectively, include the volume scaling for `b` in the reaction rate expression. Include volume scaling in the rate constant, and set the units of the rate constant accordingly (`1/(mole*second)` for concentration/time, or `1/(molarity*second)` for amount/time).

Viewing Equations for Reactions

You can view the system of equations that SimBiology creates when you build a model using reaction expressions. For details, see “View Model Equations” on page 2-57.

See Also

More About

- “Create and Simulate a Model with a Custom Function” on page 2-49
- “Model Simulation” on page 4-2
- “Deriving ODEs from Reactions” on page 4-4
- “What is a Model?” on page 2-2
- “Definitions and Evaluations of Rules” on page 2-17
- “Events” on page 2-30
- “Component Usage” on page 2-58
- “Evaluation of Model Component Names in Expressions” on page 2-61

Definitions and Evaluations of Rules

In this section...
“Overview” on page 2-17
“Initial Assignment” on page 2-17
“Repeated Assignment” on page 2-18
“Algebraic Rules” on page 2-18
“Repeated Assignment vs. Algebraic Rules” on page 2-19
“Rate Rules” on page 2-19
“Evaluation Order of Rules” on page 2-19
“Conservation of Amounts During Simulation” on page 2-20
“Writing Rule Expressions” on page 2-21
“Considerations When Imposing Constraints” on page 2-21
“Rate Rule Examples” on page 2-22

Overview

Rules are mathematical expressions that allow you to define or modify model quantities, namely compartment capacity, species amount, or parameter value.

Rules can take the form of initial assignments, assignments during the course of a simulation (repeated assignments), algebraic relationships, or differential equations (rate rules). Details of each type of rule are described next.

Initial Assignment

An initial assignment rule lets you specify the initial value of a model quantity as a numeric value or as a function of other model quantities. It is evaluated once at the beginning of a simulation.

An initial assignment rule is expressed as `Variable = Expression`, and the rule is specified as the `Expression`. For example, you could write an initial assignment rule to set the initial amount of `species2` to be proportional to `species1` as `species2 = k * species1` where `k` is a constant parameter.

Repeated Assignment

A repeated assignment rule lets you specify the value of a quantity as a numeric value or as a function of other quantities repeatedly during the simulation. It is evaluated at every time step, which is determined by the solver during the simulation process.

A repeated assignment rule is expressed as `Variable = Expression`, and the rule is specified as the `Expression`. For example, to repeatedly evaluate the total species amount by summing up the species in different compartments, you could enter: `xTotal = c1.X + c2.X`, where `xTotal` is a nonconstant parameter, `c1` and `c2` are the name of compartments where species x resides.

Algebraic Rules

An algebraic rule lets you specify mathematical constraints on one or more model quantities that must hold during a simulation. It is evaluated continuously during a simulation.

An algebraic rule takes the form `0 = Expression`, and the rule is specified as the `Expression`. For example, if you have a mass conservation equation such as `species_total = species1 + species2`, write the corresponding algebraic rule as `species1 + species2 - species_total`.

However, repeated assignment rules are mathematically equivalent to algebraic rules, but result in exact solutions instead of approximated solutions. Therefore, it is recommended that you use repeated assignment rules instead of algebraic rules whenever possible. Use algebraic rules only when:

- You cannot analytically solve the equations to get a closed-form solution. For example, there is no closed-form solution for `x^4 + ax^3 + bx^2 + cx + k = 0` whereas the closed-form solution for `kx − c = 0` is `x = c/k`.
- You have multiple equations with multiple unknowns, and they could be inconvenient to solve.

If you use an algebraic rule, rate rule, or repeated assignment to vary the value of a parameter or compartment during the simulation, make sure the `ConstantValue` property of the parameter or `ConstantCapacity` of the compartment is set to `false`.

Repeated Assignment vs. Algebraic Rules

Repeated assignment rules are mathematically equivalent to algebraic rules, but result in exact solutions. However, algebraic rules are solved numerically, and the accuracy depends on the error tolerances specified in the simulation settings. In addition, there are several advantages to repeated assignment rules such as better computational performance, more accurate results since no rules have to be solved numerically (hence no approximations), and sensitivity analysis support.

Tip

- If you can analytically solve for a variable, use a repeated assignment rule instead of an algebraic rule.
- In repeated assignment rules, the constrained variable is explicitly defined as the left-hand side, whereas in algebraic rules it is inferred from the degrees of freedom in the system of equations. See also “Considerations When Imposing Constraints” on page 2-21.

Rate Rules

A rate rule represents a differential equation and lets you specify the time derivative of a model quantity. It is evaluated continuously during a simulation.

A rate rule is represented as $\frac{dVariable}{dt} = Expression$, which is expressed in SimBiology as `Variable = Expression`. For example, if you have a differential equation for species x, $\frac{dx}{dt} = k(y + z)$, write the rate rule as: `x = k * (y + z)`.

For more examples, see “Rate Rule Examples” on page 2-22.

Evaluation Order of Rules

At the start of the simulation (that is, at simulation time = 0), SimBiology evaluates the initial assignment and repeated assignment rules as a set of simultaneous constraints. SimBiology treates the rules as a unified system of constraints and automatically reorders

and evaluates them. The order in which the rules appear in the model has no effect on the simulation results.

If a quantity is being modified by an assignment rule, the rule replaces initial value properties, such as `InitialAmount`, `Capacity`, or `Value`. Similarly, a variant altering such quantities has no effect because the value is superseded by the assignment rules.

SimBiology throws an error if the model has circular dependencies in the initial assignment and repeated assignment rules. In other words, initial assignments and repeated assignments cannot have a variable that is explicitly or implicitly referenced on both the left- and right-hand sides of the equation.

For instance, you cannot create circular sets of assignments such as `a = b + 1` and `b = a + 1`, where *a* and *b* are explicitly referenced on both sides of the equation. An example of an implicit reference is when an assignment rule references a species in concentration. In this case, the compartment that contains the species is implicitly referenced.

Compatibility Considerations

You might observe different simulation results with respect to initial assignments for previous releases of SimBiology (R2017a or earlier). To recover the same simulation results at time = 0, as in R2017a or earlier, use the `updateInitialAssignments` function in the command line. If you are using the SimBiology desktop, right-click the model from the **Project Workspace** and select **Remove Order Dependencies**.

Conservation of Amounts During Simulation

During a simulation (that is, at simulation time > 0), SimBiology conserves species amounts rather than concentrations if there are any changes to the volume of a compartment where the species reside. In other words, if you have a repeated assignment rule or an event that changes the volume, then you see the effect of conservation of species amounts at time > 0.

However, at the beginning of a simulation (that is, at simulation time = 0), the concept of amount conservation does not apply because there are no changes before time = 0. Only one set of initial conditions exists and SimBiology uses the conditions at the start of the simulation. Specifically, at time = 0, SimBiology:

1. Initializes variables for species, compartments, and parameters using the corresponding `InitialAmount`, `Capacity`, and `Value` properties.

2. Updates the values by replacing them with the corresponding alternate values from variants, if any.
3. Updates the values by evaluating initial assignment and repeated assignment rules as a set of simultaneous constraints. Therefore, the assignment rules replace initial values if model quantities are being modified by such rules or variants.

Compatibility Considerations

In previous releases (R2017a or earlier), if a repeated assignment changed a compartment volume, SimBiology used the compartment capacity to determine the initial amount and conserved it when the compartment volume changed at time = 0. In R2017b or later, SimBiology uses the `InitialAmount` property of the species as the initial condition at time = 0. Consider the following model.

```
m = sbiomodel('m1')
v = addcompartment(m,'v',10,'ConstantCapacity',0,'CapacityUnit','liter')
p = addparameter(m,'p','ValueUnit','liter')
r = addrule(m,'v = 100 * p','repeatedAssignment')
s = addspecies(v,'s',50,'InitialAmountUnit','milligram/liter')
```

In R2017a or earlier, SimBiology first calculated the initial amount of *s* as `50 milligram/liter * 10 liter = 500 milligram`, and then applied the repeated assignment rule `v = 100 liter`. So, the concentration of *s* was then calculated and reported as `500 milligram/100 liter = 5 milligram/liter` at time = 0.

In R2017b or later, SimBiology uses the `InitialAmount` property of species *s*, and reports the initial amount of *s* as `50 milligram/liter` instead.

Writing Rule Expressions

Use MATLAB syntax to write a mathematical expression for a rule. Note that no semicolon or comma is needed at the end of a rule expression. If your algebraic, repeated assignment, or rate rule expression is not continuous or differentiable, see “Using Events to Address Discontinuities in Rule and Reaction Rate Expressions” on page 2-38 before simulating your model.

Considerations When Imposing Constraints

Suppose that you have a species `y` whose amount is determined by the equation `y = m * x - c`. In SimBiology, the algebraic rule to describe this constraint is written as `m * x -`

`c - y`. If you want to use this rule to determine the value of `y`, then `m`, `x`, and `c` must be variables or constants whose values are known or determined by other equations. Therefore, you must ensure that the system of equations is not overconstrained or underconstrained. For instance, if you have more equations than unknowns, then the system is overconstrained. Conversely, if you have more unknowns than the equations, then the system is underconstrained.

Tip The behavior of an underconstrained system could be fixed by adding additional rules or by setting the `ConstantValue` or `ConstantCapacity` or `ConstantAmount` property of some of the components in the model.

Rate Rule Examples

The following examples show how to create rate rules for different applications.

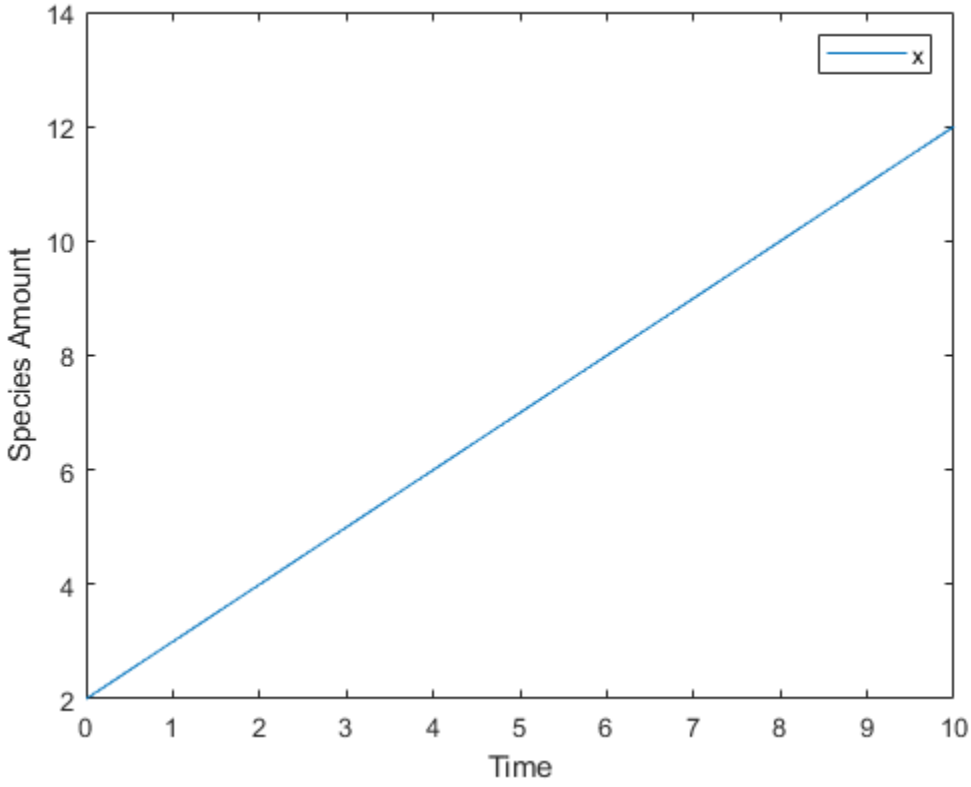
Create a Rate Rule for a Constant Rate of Change

This example shows how to increase the amount or concentration of a species by a constant value using the zero-order rate rule. For example, suppose species `x` increases by a constant rate `k`. The rate of change is:

$$dx/dt = k$$

Set the initial amount of species `x` to 2, and the value of parameter `k` to 1. Use the following commands to set up a SimBiology model accordingly and simulate it.

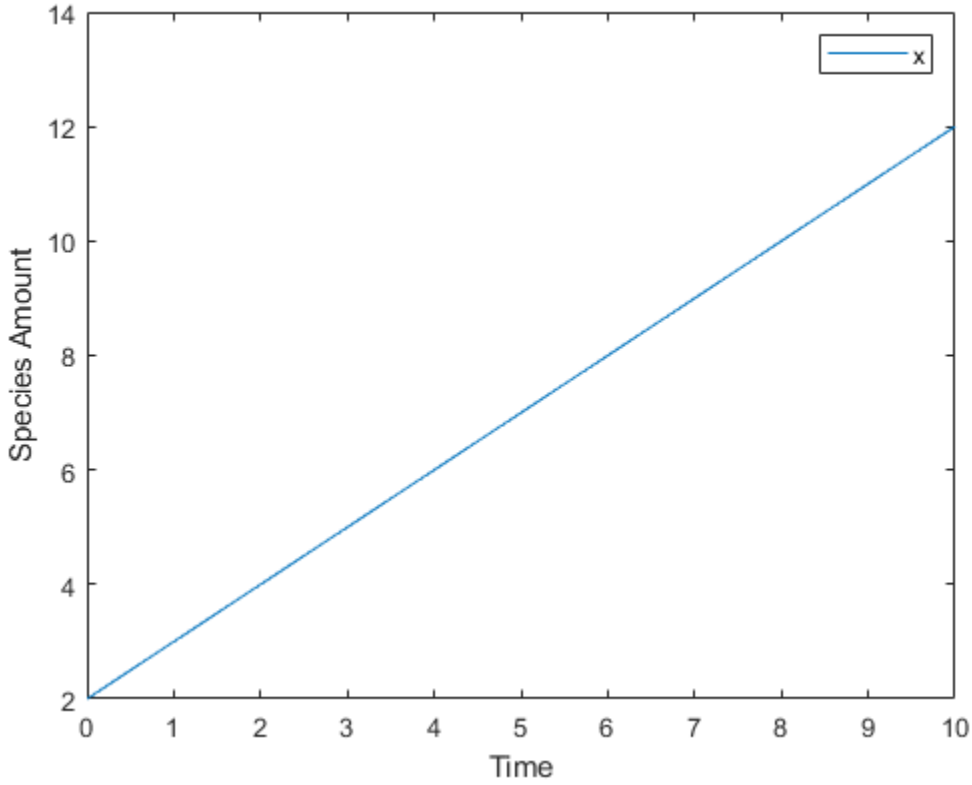
```
m = sbiomodel('m');
c = addcompartment(m,'comp');
s = addspecies(m,'x','InitialAmount',2);
p = addparameter(m,'k','Value',1);
r = addrule(m,'x = k','RuleType','rate');
[t,sd,species] = sbiosimulate(m);
plot(t,sd);
legend(species)
xlabel('Time');
ylabel('Species Amount');
```



Alternatively, you could model a constant increase in a species using the Mass Action reaction `null -> x` with the forward rate constant `k`.

```
clear
m = sbiomodel('m');
c = addcompartment(m,'comp');
s = addspecies(m,'x','InitialAmount',2);
r = addreaction(m,'null -> x');
kl = addkineticlaw(r,'MassAction');
p = addparameter(kl,'k','Value',1);
kl.ParameterVariableNames = 'k';
[t,sd,species] = sbiosimulate(m);
plot(t,sd);
legend(species)
```

```
xlabel('Time');
ylabel('Species Amount');
```



Create a Rate Rule for an Exponential Rate of Change

This example shows how to change the amount of a species similar to a first-order reaction using the first-order rate rule. For example, suppose the species x decays exponentially. The rate of change of species x is:

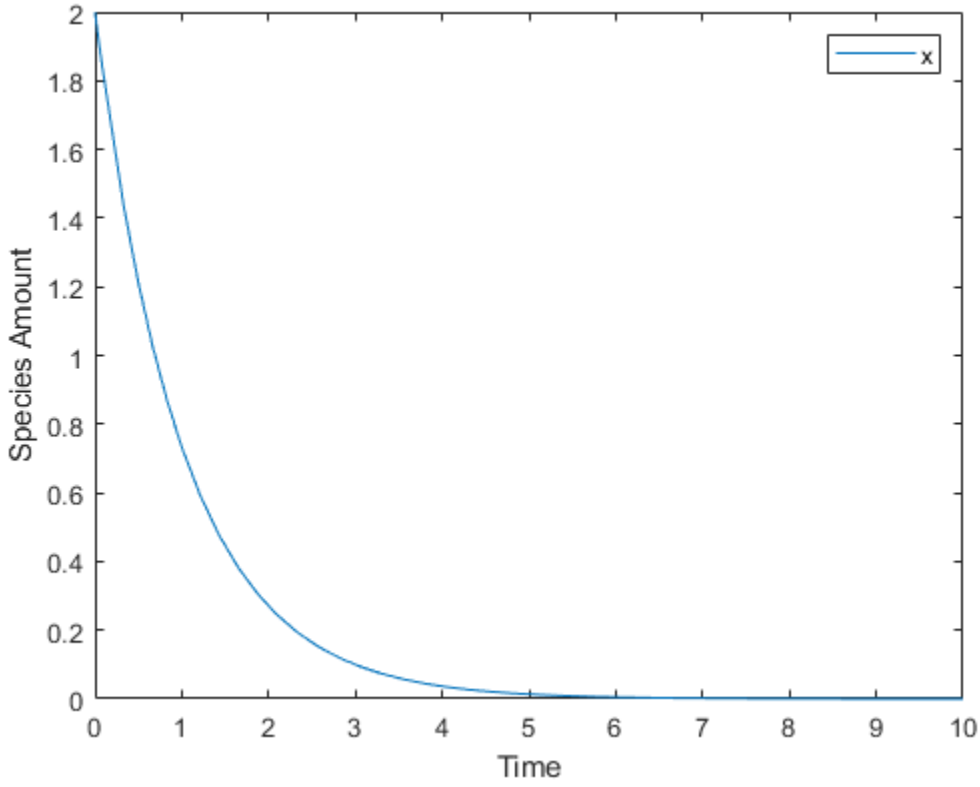
$$dx/dt = -k * x$$

The analytical solution is:

$$C_t = C_0 * e^{-kt}$$

where C_t is the amount of species at time t, and C_0 is the initial amount. Use the following commands to set up a SimBiology model accordingly and simulate it.

```
m = sbiomodel('m');
c = addcompartment(m,'comp');
s = addspecies(m,'x','InitialAmount',2);
p = addparameter(m,'k','Value',1);
r = addrule(m,'x = -k * x','RuleType','rate');
[t,sd,species] = sbiosimulate(m);
plot(t,sd);
legend(species);
xlabel('Time');
ylabel('Species Amount');
```



If the amount of a species `x` is determined by a rate rule and `x` is also in a reaction, `x` must have its `BoundaryCondition` property set to `true`. For example, with a reaction `a -> x` and a rate rule $\frac{dx}{dt} = k * x$, set the `BoundaryCondition` property of species `x` to `true` so that a differential rate term is not created from the reaction. The amount of `x` is determined solely by a differential rate term from the rate rule. If the `BoundaryCondition` property is set to `false`, you will get the following error message such as `Invalid rule variable 'x' in rate rule or reaction`.

Create a Rate Rule to Define a Differential Rate Equation

Many mathematical models in the literature are described with differential rate equations for the species. You could manually convert the equations to reactions, or you could enter the equations as rate rules. For example, you could enter the following differential rate equation for a species `C`:

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

as a rate rule in SimBiology: `C = vi - (vd*X*C)/(Kc + C) - kd*C`

Create a Rate Rule for the Rate of Change That Is Determined by Another Species

This example shows how to create a rate rule where a species from one reaction can determine the rate of another reaction if it is in the second reaction rate equation. Similarly, a species from a reaction can determine the rate of another species if it is in the rate rule that defines that other species. Suppose you have a SimBiology model with three species (`a`, `b`, and `c`), one reaction (`a -> b`), and two parameters (`k1` and `k2`). The rate equation is defined as $b = -k_1 * a$, and rate rule is $dc/dt = k_2 * a$. The solution for the species in the reaction are:

$$a = a_o e^{-k_1 t},\ b = a_o(1 - e^{-k_1 t}).$$

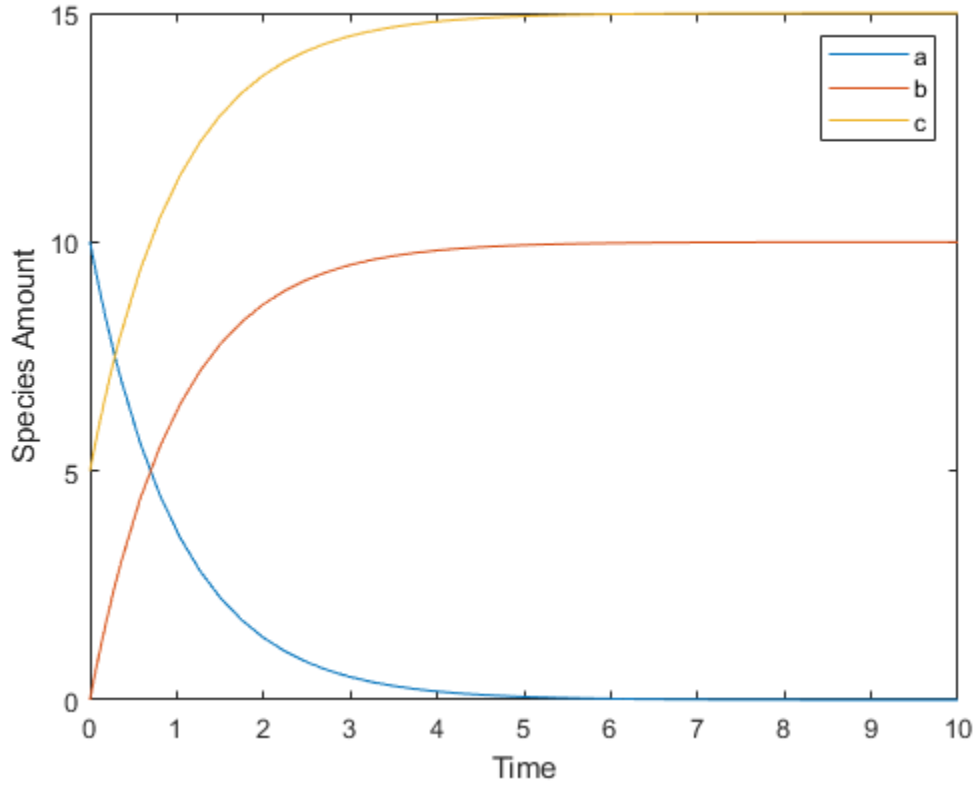
Since the rate rule $dc/dt = k_2 * a$ is dependent on the reaction, $dc/dt = k_2(a_o e^{-k_1 t})$. The solution is:

$$c = c_o + k_2 a_o / k_1 (1 - e^{-k_1 t})$$

Enter the following commands to set up a SimBiology model accordingly and simulate it.

```
m = sbiomodel('m');
c = addcompartment(m,'comp');
s1 = addspecies(m,'a','InitialAmount',10,'InitialAmountUnits','mole');
s2 = addspecies(m,'b','InitialAmount',0,'InitialAmountUnits','mole');
s3 = addspecies(m,'c','InitialAmount',5,'InitialAmountUnits','mole');
rxn = addreaction(m,'a -> b');
```

```
kl = addkineticlaw(rxn,'MassAction');
p1 = addparameter(kl,'k1','Value',1,'ValueUnits','1/second');
rule = addrule(m,'c = k2 * a','RuleType','rate');
kl.ParameterVariableNames = 'k1';
p2 = addparameter(m,'k2','Value',1,'ValueUnits','1/second');
[t,sd,species] = sbiosimulate(m);
plot(t,sd);
legend(species);
xlabel('Time');
ylabel('Species Amount');
```



See Also

More About

- “What is a Model?” on page 2-2
- “Definitions and Evaluations of Reactions” on page 2-10
- “Events” on page 2-30
- “Component Usage” on page 2-58
- “Evaluation of Model Component Names in Expressions” on page 2-61

Events

In this section...
“Overview” on page 2-30
“Event Triggers” on page 2-30
“Event Functions” on page 2-31
“Specifying Event Triggers” on page 2-31
“Specifying Event Functions” on page 2-33
“Simulation Solvers for Models Containing Events” on page 2-34
“How Events Are Evaluated” on page 2-34
“Evaluation of Simultaneous Events” on page 2-36
“Evaluation of Multiple Event Functions” on page 2-37
“When One Event Triggers Another Event” on page 2-37
“Cyclical Events” on page 2-38
“Using Events to Address Discontinuities in Rule and Reaction Rate Expressions” on page 2-38

Overview

In SimBiology, an event is a discrete transition in value of a quantity or expression in a model. This discrete transition occurs when a customized condition becomes true. The condition can be a specific time and/or a time-independent condition. Such conditions are defined in an `Event object`.

Event Triggers

An event object has a `Trigger` property that specifies a condition that must be true to trigger the event to execute.

Typical event triggers are:

- A specific simulation time — Specify that the event must change the amounts or values of species or parameters. For example, at time = 5 s, increase the amount of an inhibitor species above the threshold to inhibit a given reaction.

- In response to state or changes in the system — Change amounts/values of certain species/parameters in response to events that are not tied to any specific time. For example, when species `A` reaches an amount of `30` molecules, double the value of reaction rate constant `k`. Or when temperature reaches `42 °C`, inhibit a particular reaction by setting its reaction rate to zero.

Note Currently, events cannot be triggered at time = 0. However, you can get the event to happen just after time = 0 by using `time > timeSmall` as the event trigger where `timeSmall` can be a tiny fraction of a second such as 1.0 picosecond.

Event Functions

An event has an `EventFcns` property that specifies what occurs when the event is triggered. Event functions can range from simple to complex. For example, an event function might:

- Change the values of compartments, species, or parameters.
- Double the value of a reaction rate constant.

Specifying Event Triggers

The `Trigger` property of an event specifies a condition that must become true for an event to execute. Typically, the condition uses a combination of relational and logical operators to build a trigger expression.

A trigger can contain the keyword `time` and relational operators to trigger an event that occurs at a specific time during the simulation. For example, `time >= x`. For more information see the `Trigger` property.

Use MATLAB syntax to write expressions for event triggers. Note that the expression must be a single MATLAB statement that returns a logical. No semicolon or comma is needed at the end of an expression. MATLAB uses specific operator precedence to evaluate trigger expressions. Precedence levels determine the order in which MATLAB evaluates an expression. Within each precedence level, operators have equal precedence and are evaluated from left to right. To find more information on how relational and logical operators are evaluated see “Relational Operations” (MATLAB) and “Logical Operations” (MATLAB).

Some examples of triggers are:

Trigger	Explanation
`(time >= 5) && (speciesA < 1000)`	Execute the event when the following condition becomes true: Time is greater than or equal to `5`, and `speciesA` is less than 1000. **Tip** Using a && (instead of &) evaluates the first part of the expression for whether the statement is true or false, and skips evaluating the second statement if this statement is false.
`(time >= 5) || (speciesA < 1000)`	Execute the event when the following condition becomes true: Time is greater than or equal to `5`, or if `speciesA` is less than 1000.
`(s1 >= 10.0) || (time >= 250) && (s2 < 5.0E17)`	Execute the event when the following condition becomes true: Species, `s1` is greater than or equal to `10.0` or, time is greater than or equal to `250` and species `s2` is less than `5.0E17`. Because of operator precedence, the expression is treated as if it were `(s1 >=10.0) || ((time>= 250) && (s2<5.0E17))`. Thus, it is always a good idea to use parenthesis to explicitly specify the intended precedence of the statements.
`((s1 >= 10.0) || (time >= 250)) && (s2 < 5.0E17)`	Execute the event when the following condition becomes true: Species `s1` is greater than or equal to `10` or time is greater than or equal to `250`, and species `s2` is less than `5.0E17`.

Trigger	Explanation
`((s1 >= 5000.0) && (time >= 250)) || (s2 < 5.0E17)`	Execute the event when the following condition becomes true: Species `s1` is greater than or equal to `5000` and time is greater than or equal to `250`, or species `s2` is less than `5.0E17`.

Tip If UnitConversion is on and your model has any event, follow the recommendation below.

Non-dimensionalize any parameters used in the event `trigger` if they are not already dimensionless. For example, suppose you have a trigger `x > 1`, where *x* is the species concentration in mole/liter. Non-dimensionalize *x* by scaling (dividing) it with a constant such as `x/x0 > 1`, where *x0* is a parameter defined as 1.0 mole/liter. Note that *x* does not have to have the same unit as the constant *x0*, but must be dimensionally consistent with it. For example, the unit of *x* can be picomole/liter instead of mole/liter.

Specifying Event Functions

The `EventFcns` property of an event specifies what occurs when the event is triggered. You can use an event function to change the value of a compartment, species, or parameter, or you can specify complex tasks by calling a custom function or script.

Use MATLAB syntax to define expressions for event functions. The expression must be a single MATLAB assignment statement that includes `=`, or a cell array of such statements. No semicolon or comma is needed at the end of the expression.

Following are rules for writing expressions for event functions:

EventFcn	Explanation
`speciesA = speciesB`	When the event is executed, set the amount of `speciesA` equal to that of `speciesB`.
`k = k/2`	When the event is executed, halve the value of the rate constant `k`.

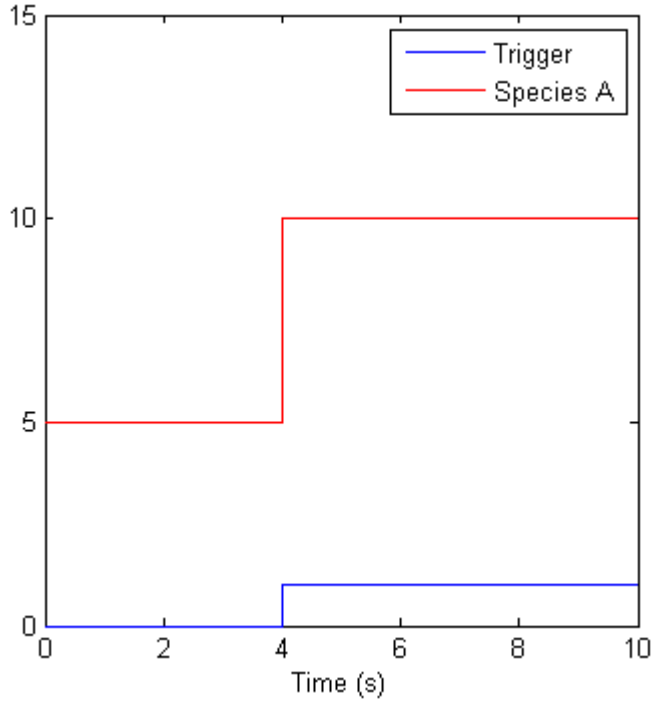
EventFcn	Explanation
`{'speciesA = speciesB','k = k/2'}`	When the event is executed, set the amount of `speciesA` equal to that of `speciesB`, and halve the value of the rate constant `k`.
`kC = my_func(A,B,kC)`	When the event is executed, call the custom function `my_func()`. This function takes three arguments: The first two arguments are the current amounts of two species (`A` and `B`) during simulation and the third argument is the current value of a parameter, `kC`. The function returns the modified value of `kC` as its output.

Simulation Solvers for Models Containing Events

To simulate models containing events, use a deterministic (ODE or SUNDIALS) solver or the stochastic `ssa` solver. Other stochastic solvers do not support events. For more information, see “Choosing a Simulation Solver” on page 4-7.

How Events Are Evaluated

Consider the example of a simple event where you specify that at `4s`, you want to assign a value of `10` to species `A`.



At `time = 4 s` the trigger becomes true and the event executes. In the previous figure assuming that `0` is false and `1` is true, when the trigger becomes true, the amount of species `A` is set to `10`. In theory, with a perfect solver, the event would be executed exactly at `time = 4.00 s`. In practice there is a very minute delay (for example you might notice that the event is executed at `time = 4.00001 s`). Thus, you must specify that the trigger can become true at or after `4s`, which is `time >= 4 s`.

Trigger	EventFcn
`time >= 4`	`A = 10`

The point at which the trigger becomes true is called a rising edge. SimBiology events execute the `EventFcn` *only* at rising edges.

The trigger is evaluated at every time step to check whether the condition specified in the trigger transitions from false to true. The solver detects and tracks falling edges, which is when the trigger becomes false, so if another rising edge is encountered, the event is reexecuted. If a trigger is already true before a simulation starts, then the event does not

execute at the start of the simulation. The event is not executed until the solver encounters a rising edge. Very rarely, the solver might miss a rising edge. An example of this is when a rising edge follows very quickly after a falling edge, and the step size results in the solver skipping the transition point.

If the trigger becomes true exactly at the stop time of the simulation, the event might or might not execute. If you want the event to execute, increase the stop time.

Note Since the rising edge is instantaneous and changes the system state, there are two values for the state at the same time. The simulation data thus contains the state before and after the event, but both points are at the same time value. This leads to multiple values of the system state at a single instant in time.

Evaluation of Simultaneous Events

When two or more trigger conditions simultaneously become true, the solver executes the events sequentially in the order in which they are listed in the model. You can reorder events using the `reorder` method. For example, consider this case.

Event Number	Trigger	EventFcn
1	`SpeciesA >= 4`	`SpeciesB = 10`
2	`SpeciesC >= 15`	`SpeciesB = 25`

The solver tries to find the rising edge for these events within a certain level of tolerance. If this results in both events occurring simultaneously, then the value of `SpeciesB` after the time step in which these two events occur, will be `25`. If you reorder the events to reverse the event order, then the value of `SpeciesB` after the time step in which these two events occur, will be `10`.

Consider an example in which you include event functions that change model components in a dependent fashion. For example, the event function in Event 2, stipulates that `SpeciesB` takes the value of `SpeciesC`.

Event Number	Trigger	EventFcn
1	`SpeciesA >= 4`	`SpeciesC = 10`

Event Number	Trigger	EventFcn
2	`time >= 15`	`SpeciesB = SpeciesC`

Event 1 and Event 2 might or might not occur simultaneously.

- If Event 1 and Event 2 do not occur simultaneously, when Event 2 is triggered, `SpeciesB` is assigned the value that `SpeciesC` has at the time of the event trigger.
- If Event 1 and Event 2 occur simultaneously, the solver executes Event 1 first, then executes Event 2. In this example, if `SpeciesC = 15` when the events are triggered, after the events are executed, `SpeciesC = 10` and `SpeciesB = 10`.

Evaluation of Multiple Event Functions

Consider an event function in which you specify that the value of a model component (`SpeciesB`) depends on the value of model component (`SpeciesA`), but `SpeciesA` also is changed by the event function.

Trigger	EventFcn
`time >= 4`	`{'SpeciesA = 10, SpeciesB = SpeciesA'}`

The solver stores the value of `SpeciesA` at the rising edge and before any event functions are executed and uses this stored value to assign `SpeciesB` its value. So in this example if `SpeciesA = 15` at the time the event is triggered, after the event is executed, `SpeciesA = 10` and `SpeciesB = 15`.

When One Event Triggers Another Event

In the next example, Event 1 includes an expression in the event function that causes Event 2 to be triggered (assuming that `SpeciesA` has amount less than `5` when Event 1 is executed).

Event Number	Trigger	EventFcn
1	`time >= 5`	`{'SpeciesA = 10, SpeciesB = 5'}`
2	`SpeciesA >= 5`	`SpeciesC = SpeciesB`

When Event 1 is triggered, the solver evaluates and executes Event 1 with the result that `SpeciesA = 10` and `SpeciesB = 5`. Now, the trigger for Event 2 becomes true and the

solver executes the event function for Event 2. Thus, `SpeciesC = 5` at the end of this event execution.

You can thus have event cascades of arbitrary length, for example, Event 1 triggers Event 2, which in turn triggers Event 3, and so on.

Cyclical Events

In some situations, a series of events can trigger a cascade that becomes cyclical. Once you trigger a cyclical set of events, the only way to stop the simulation is by pressing **Ctrl +C**. You lose any data acquired in the current simulation. Here is an example of cyclical events. This example assumes that `Species B <= 4` at the start of the cycle.

Event Number	Trigger	EventFcn
1	`SpeciesA > 10`	`{'SpeciesB = 5', 'SpeciesC = 1'}`
2	`SpeciesB > 4`	`{'SpeciesC = 10', 'SpeciesA = 1'}`
3	`SpeciesC > 9`	`{'SpeciesA = 15', 'SpeciesB = 1'}`

Using Events to Address Discontinuities in Rule and Reaction Rate Expressions

The solvers provided with SimBiology gives inaccurate results when the following expressions are not continuous and differentiable:

- Repeated assignment rule
- Algebraic rule
- Rate rule
- Reaction rate

Either ensure that the previous expressions are continuous and differentiable or use events to reset the solver at the discontinuity, as described in “Deterministic Simulation of a Model Containing a Discontinuity”.

See Also

More About

- “What is a Model?” on page 2-2
- “Definitions and Evaluations of Reactions” on page 2-10
- “Definitions and Evaluations of Rules” on page 2-17
- “Component Usage” on page 2-58
- “Evaluation of Model Component Names in Expressions” on page 2-61

Variants

A variant stores alternate values of model parameters and initial conditions. You can use variants to evaluate model behavior under different experimental or initial conditions, without having to change the existing values or create additional models with the new values.

A variant lets you store an alternate value for any of the following model elements:

- Compartment `Capacity` property
- Species `InitialAmount` property
- Parameter `Value` property

Simulating using a variant does not alter the model original values. The values specified in the variant are temporarily applied to the model during simulation. You can permanently replace the values in your model with the values stored in the variant object by committing it to the model. When you use multiple variants during a simulation, and there are duplicate specifications for a property value, the last occurrence for the property value in the array of variants is used during simulation.

Creating Variants Programmatically

There are two ways to create variants or add variants to a model. To create a standalone variant that is not attached to any model, use `sbiovariant`. To add a variant to an existing model, use `addvariant`. Use the `commit` function to replace the values in your model with the variant values permanently.

For illustrated examples of using variants, see the following.

- “Simulate Biological Variability of the Yeast G Protein Cycle Using the Wild-Type and Mutant Strains” on page 2-47
- “Simulate Model of Glucose-Insulin Response with Different Initial Conditions” on page 3-22

Creating Variants Graphically

You can interactively create and add variants using the SimBiology desktop. For details, see “Diagram View” on page 1-25.

See Also

`addcontent` | `addvariant` | `commit` | `sbiovariant` | `variant object`

More About

- “SimBiology Desktop”

Doses

Doses let you increase the amount of a species in a SimBiology model during simulation, either at specific time points or regular intervals. For example, you can use a dose object to model an instantaneous supply of a drug regimen during the simulation of a model. The increase in the amount of a species occurs only during simulation and does not alter the species value permanently (that is, the value in the model is not changed).

Representing Doses

There are two types of dose objects.

- `ScheduleDose object` — Applies a dose to a single species at a predefined list of time points
- `RepeatDose object` — Applies a dose to a single species at regular intervals

SimBiology dose objects support the following dosing types.

Dosing Strategy	Description	Dose Object Properties Configuration
Bolus	Instantaneous increase in the amount of drug in the compartment	To create a bolus dose, set the Amount and TargetName properties of a dose object. You might also need to configure other properties such as RepeatCount, Interval, or scheduled dose times (Time) if you are applying a series of doses. For details on these properties, see `ScheduleDose object` and `RepeatDose object` .
Infusion	Increase of the drug at a fixed rate over a period of time, which is calculated from the dose amount	Unlike a bolus dose, you also need to specify the infusion rate (Rate property) of the dose object.

Dosing Strategy	Description	Dose Object Properties Configuration
Zero-order	Increase of the drug at a fixed rate calculated from the dose amount and dose duration	Unlike a bolus dose, you also need to create a zero-order duration parameter and specify the duration parameter name (`DurationParameterName`property) of the dose object.
First-order	Increase of the drug via first-order absorption kinetics	Unlike bolus, infusion, or zero-order, you need to create an additional reaction for the drug absorption.

Creating Doses Programmatically

There are two common ways to create dose objects using the command-line interface. One way is to create a dose object using the `sbiodose` or `adddose` function. Another is to create dose objects automatically from data containing dosing information. This first approach is useful when you want to explore different dosing strategies through simulation. The second approach is useful if you already have a data set with dosing information and plan to use this dosing information in your simulation or parameter estimation.

Create a Dose Object Using sbiodose or adddose

`sbiodose` creates a standalone dose object that is not attached to any model. You can apply a standalone dose to different models during simulation by specifying it as a dosing argument for `sbiosimulate`, or attach it to any model using `adddose`. You can also use it during parameter estimation using `sbiofit` or `sbiofitmixed`.

`adddose` creates a dose object and adds it to a model. You must set its `Active` property to `true` to apply the dose to the model during simulation.

The following examples show how to add a dose object to a one-compartment PK model using `sbiodose` and set up the dose properties manually. Alternatively, you can use the built-in PK models with different dosing types. For details, see “Create Pharmacokinetic Models” on page 5-24.

Dosing Strategy	Example
Bolus	“Add a Series of Bolus Doses to a One-Compartment Model”
Infusion	“Add an Infusion Dose to a One-Compartment Model”
Zero-order	“Increase Drug Concentration in a One-Compartment Model via Zero-order Dosing”
First-order	“Increase Drug Concentration in a One-Compartment Model via First-order Dosing”

Create Dose Objects from Dosing Data

If you already have dosing data for one or more subjects or patients that you would like to use in your parameter estimation, first create a `groupedData object` from your data set. Use `createDoses` function to automatically generate an array of dose objects. You can then use the dose array during parameter estimation using `sbiofit` or `sbiofitmixed`. For a complete workflow, see “Modeling the Population Pharmacokinetics of Phenobarbital in Neonates”.

Creating Doses Graphically

You can interactively create and add doses using the SimBiology desktop. For details, see “Diagram View” on page 1-25.

Parameterized and Adaptive Doses

You can specify some of the properties of `RepeatDose` and `ScheduleDose` objects by using model parameters. This parameterization of dose properties gives you more flexibility in modeling different dosing applications, such as scaling the dose amount by body weight.

`RepeatDose` properties that you can parameterize are: Amount, Rate, Interval, StartTime, RepeatCount, LagParameterName, and DurationParameterName. `ScheduleDose` properties that can be parameterized are LagParameterName and DurationParameterName. You can set these `RepeatDose` properties, except `LagParameterName` and `DurationParameterName`, to either a numeric value or the name of a model-scoped parameter (as a character vector or string).

You can make doses adaptive to events, such as increasing the dose amount when the drug concentration drops below some threshold. This adaptive feature of doses is useful

for doses that are not instantaneous. Consider an IV infusion for a drug being added at a fixed rate over a fixed duration. If an event modifies a dose parameter while this dose is in progress, you have two options:

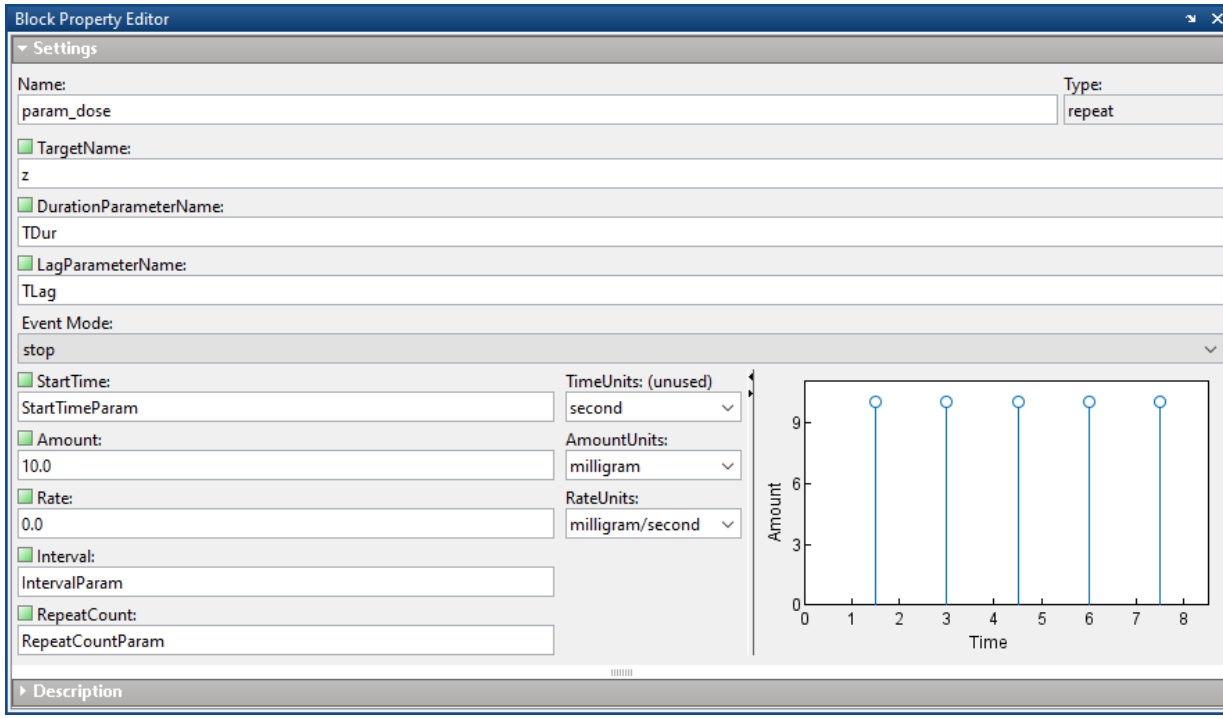
- Stop the ongoing dose if any relevant parameter values change by setting the `EventMode` property of the dose object to `'stop'`.
- Continue the ongoing dose to completion, and updated parameter values affect only subsequent doses by setting `EventMode` to `'continue'`.

For details, see the EventMode property. For illustrated examples, see “Scale Dose Amount by Body Weight” and “Change Dose Behavior In Response to Changes in Model Parameters”.

Units Validation on Parameterized Dose Properties

If you parameterize a dose property and enable dimensional analysis, the unit of the dose property (dose unit) is validated. The dose unit is valid either if it is empty or if it exactly matches the unit of the parameter. If the dose unit is invalid, SimBiology issues a warning in the command line and uses the unit of the parameter instead. To remove the warning, set the dose unit to empty (`''`) or to the same unit as the parameter unit.

In the SimBiology desktop, the text `(unused)` appears next to the corresponding unit field in the **Block Property Editor** or **Doses** tab, meaning that SimBiology is using the parameter unit instead of the dose unit during simulation. For example, this figure shows the details of the repeat dose object *param_dose* in the **Block Property Editor**. The **TimeUnits** option is designated as `unused`. Instead, SimBiology uses the unit of the parameter *StartTimeParam* during simulation. The **Amount** and **Rate** properties are not parameterized, and hence the selected **AmountUnits** and **RateUnits** options are used. To see the units of referenced parameters, click the indicator next to the corresponding property. The **TimeUnits** option corresponds to both the **StartTime** and **Interval** properties.



Simulation Solvers for Models Containing Doses

To simulate models containing doses, use a deterministic (ODE or SUNDIALS) solver. Stochastic solvers do not support doses. For details, see “Choosing a Simulation Solver” on page 4-7.

See Also

`RepeatDose object` | `ScheduleDose object` | `adddose` | `sbiodose`

More About

- “SimBiology Desktop”

Simulate Biological Variability of the Yeast G Protein Cycle Using the Wild-Type and Mutant Strains

This example shows how to create and apply a variant to the G protein model of a wild-type strain. The variant represents a parameter value for the G protein model of a mutant strain. Thus, when you simulate the model without applying the variant, you see results for the wild type strain, and when you simulate the model with the variant, you see results for the mutant strain. This example uses the model described in Model of the Yeast Heterotrimeric G Protein Cycle on page B-18.

The value of the parameter `kGd` is `0.11` for the wild-type strain and `0.004` for the mutant strain. To represent the mutant strain, you will store an alternate value of `0.004` for the `kGd` parameter in a variant object, and apply this variant when simulating the model.

For information on variants, see “Variants” on page 2-40.

Load the `gprotein.sbproj` project, which includes the variable `m1`, a SimBiology model object.

```
sbioloadproject gprotein
```

You can create a variant of the original model by specifying a different parameter value for the `kGd` parameter of the model. First, add a variant to the `m1` model object.

```
v1 = addvariant(m1,'mutant_strain');
```

Next, add a parameter `kGd` with a value of `0.004` to the variant object `v1`.

```
addcontent(v1,{'parameter','kGd','Value',0.004});
```

Simulate the wild type model.

```
[t,x,names] = sbiosimulate(m1);
```

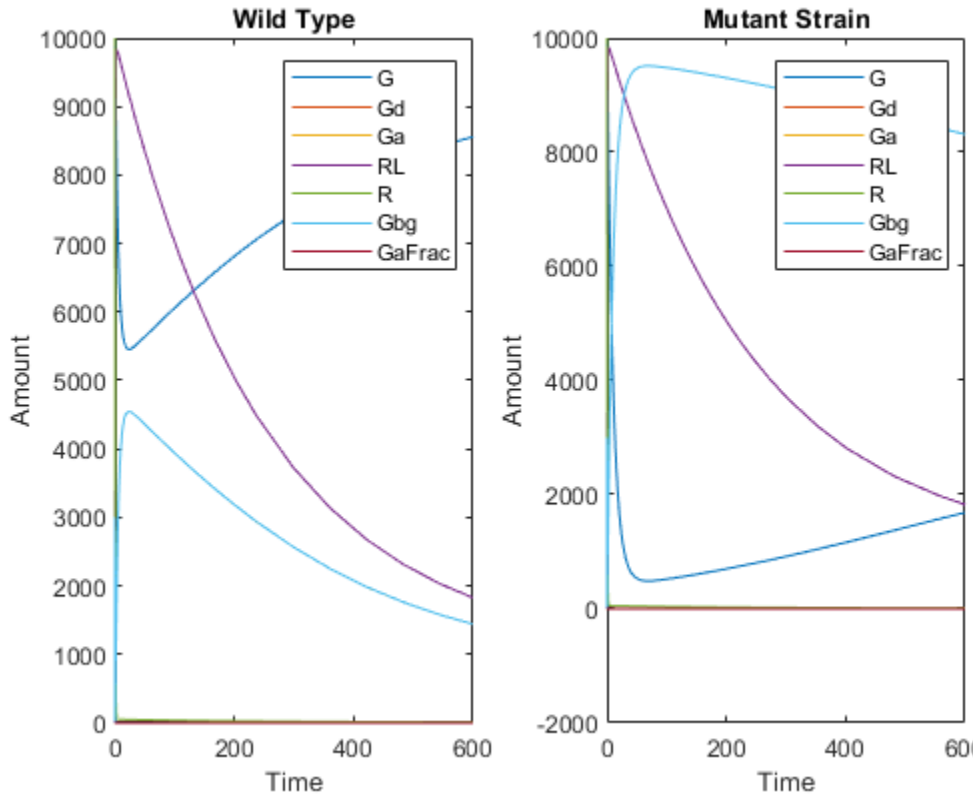
Simulate the mutant strain model by applying the variant.

```
[tV,xV,names] = sbiosimulate(m1,v1);
```

Plot and compare the simulated results.

```
subplot(1,2,1)
plot(t,x);
legend(names);
```

```
xlabel('Time');
ylabel('Amount');
title('Wild Type');

subplot(1,2,2)
plot(tV,xV);
legend(names);
xlabel('Time');
ylabel('Amount');
title('Mutant Strain');
```



Create and Simulate a Model with a Custom Function

This example shows how to create a custom function and incorporate it in model simulation.

Overview

Prerequisites for the Example

This example assumes you have a working knowledge of:

- MATLAB desktop
- Creating and saving MATLAB programs

About the Example Model

This example uses the model described in Model of the Yeast Heterotrimeric G Protein Cycle on page B-18.

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

No.	Name	Reaction[1]	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`
5	Receptor-ligand degradation	`RL -> null`	`kRD1`
6	G protein inactivation	`Ga -> Gd`	`kGd`

[1] Legend of species: L = ligand (alpha factor), R = alpha-factor receptor, Gd = inactive G-alpha-GDP, Gbg = free levels of G-beta:G-gamma complex, G = inactive Gbg:Gd complex, Ga = active G-alpha-GTP

Assumptions of the Example

This example assumes that:

- An inhibitor (`Inhib` species) slows the inactivation of the active G protein (reaction 6 above, `Ga –> Gd`).
- The variation in the amount of inhibitor (`Inhib` species) is defined in a custom function, `inhibvalex`.
- The inhibitor (`Inhib` species) affects the reaction by changing the amount of rate parameter `kGd`.

About the Example

This example shows how to create and call a custom function in a SimBiology expression. Specifically, it shows how to use a custom function in a rule expression.

About Using Custom Functions in SimBiology Expressions

You can use custom functions in:

- Reaction rate expressions (`ReactionRate` property)
- Rule expressions (`Rule` property)
- Event expressions (`EventFcns` property or `Trigger` property)

The requirements for using custom functions in SimBiology expressions are:

- Create a custom function. For more information, see `function`.
- Change the current folder to the folder containing your custom MATLAB file. Do this by using the `cd` command or by using the Current Folder field in the MATLAB desktop toolbar. Alternatively, add the folder containing your file to the search path. Do this by using the `addpath` command or see “Change Folders on the Search Path” (MATLAB).
- Call the custom function in a SimBiology reaction, rule, or event expression.

Tip If your rule or reaction rate expression is not continuous and differentiable, see “Using Events to Address Discontinuities in Rule and Reaction Rate Expressions” on page 2-38 before simulating your model.

Create a Custom Function

The following procedure creates a custom function, `inhibvalex`, which lets you specify how the inhibitor amount changes over time. The inputs are time, the initial amount of inhibitor, and a parameter that governs the amount of inhibitor. The output of the function is the amount of inhibitor.

1. In the MATLAB desktop, select **File > New > Script**, to open the MATLAB Editor.
2. Copy and paste the following function declaration:

```
% inhibvalex.m
function Cp = inhibvalex(t, Cpo, kel)

% This function takes the input arguments t, Cpo, and kel
% and returns the value of the inhibitor Cp.
% You can later specify the input arguments in a
% SimBiology rule expression.
% For example in the rule expression, specify:
% t as time (a keyword recognized as simulation time),
% Cpo as a parameter that represents the initial amount of inhibitor,
% and kel as a parameter that governs the amount of inhibitor.

if  t < 400
    Cp = Cpo*exp(-kel*(t));
else
    Cp = Cpo*exp(-kel*(t-400));
end
```

3. Save the file (name the file `inhibvalex.m`) in a directory that is on the MATLAB search path, or to a directory that you can access.
4. If the location of the file is not on the MATLAB search path, change the working directory to the file location.

Load the Example Model

Load the `gprotein` example project, which includes the variable `m1`, a model object:

```
sbioloadproject gprotein
```

The `m1` model object appears in the MATLAB Workspace.

Add the Custom Function to the Example Model

The following procedure creates a rule expression that calls the custom function, `inhibvalex`, and specifies the three input values to this function.

1. Add a repeated assignment rule to the model that specifies the three input values to the custom function, `inhibvalex`:

   ```
   rule1 = addrule(m1, 'Inhib = inhibvalex(time, Cpo, Kel)',...
                   'repeatedAssignment');
   ```

   The `time` input is a SimBiology keyword recognized as simulation time
2. Create the two parameters used by the `rule1` rule and assign values to them:

   ```
   p1 = addparameter(m1, 'Cpo', 250);
   p2 = addparameter(m1, 'Kel', 0.01);
   ```
3. Create the species used by the `rule1` rule:

   ```
   s1 = addspecies(m1.Compartments, 'Inhib');
   ```

Define a Rule to Change Parameter Value

The value of rate parameter `kGd` is affected by the amount of inhibitor present in the system. Add a rule to the model to describe this action, but first change the `ConstantValue` property of the parameter `kGd` so that it can be varied by a rule.

1. Change the `ConstantValue` property of the `kGd` parameter to `false`.

   ```
   p3 = sbioselect(m1, 'Type', 'parameter', 'Name', 'kGd');
   p3.ConstantValue = false;
   ```
2. Add a repeated assignment rule to the model to define how the `kGd` parameter is affected by the `Inhib` species.

   ```
   rule2 = addrule(m1, 'kGd = 1/Inhib', 'repeatedAssignment');
   ```

Add an Event to Reset the Solver at a Discontinuity

The custom function, `inhibvalex`, introduces a discontinuity in the model when time = 400. To ensure accurate simulation results, add an event to the model to reset the solver at the time of the discontinuity. Set the event to trigger at the time of the discontinuity

(time = 400). The event does not need to modify the model, so create an event function that multiplies a species value by 1.

```
addevent(m1, 'time>=400', 'G=1*G');
```

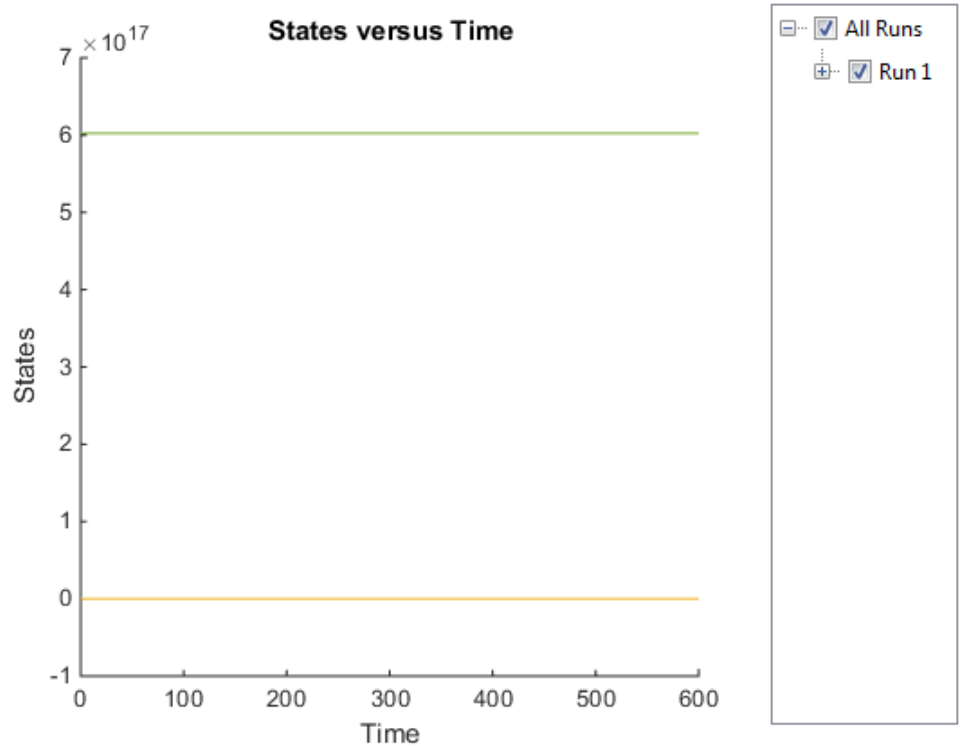
Simulate the Modified Model

1. Configure the simulation settings (`configset object`) for the `m1` model object to log all states during the simulation.

   ```
   cs = getconfigset(m1);
   cs.RuntimeOptions.StatesToLog = 'all';
   ```
2. Simulate the model.

   ```
   simDataObj = sbiosimulate(m1);
   ```
3. Plot the results.

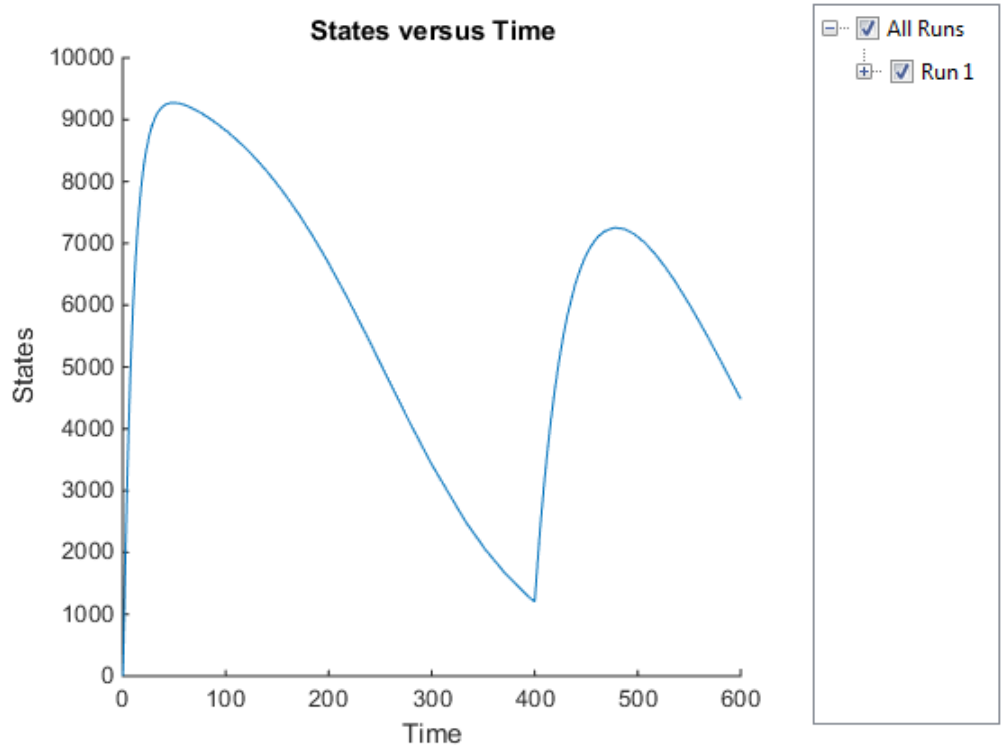
   ```
   sbioplot(simDataObj);
   ```



The plot does not show the species of interest due to the wide range in species amounts/concentrations.

4 Plot only the species of interest. `Ga`.

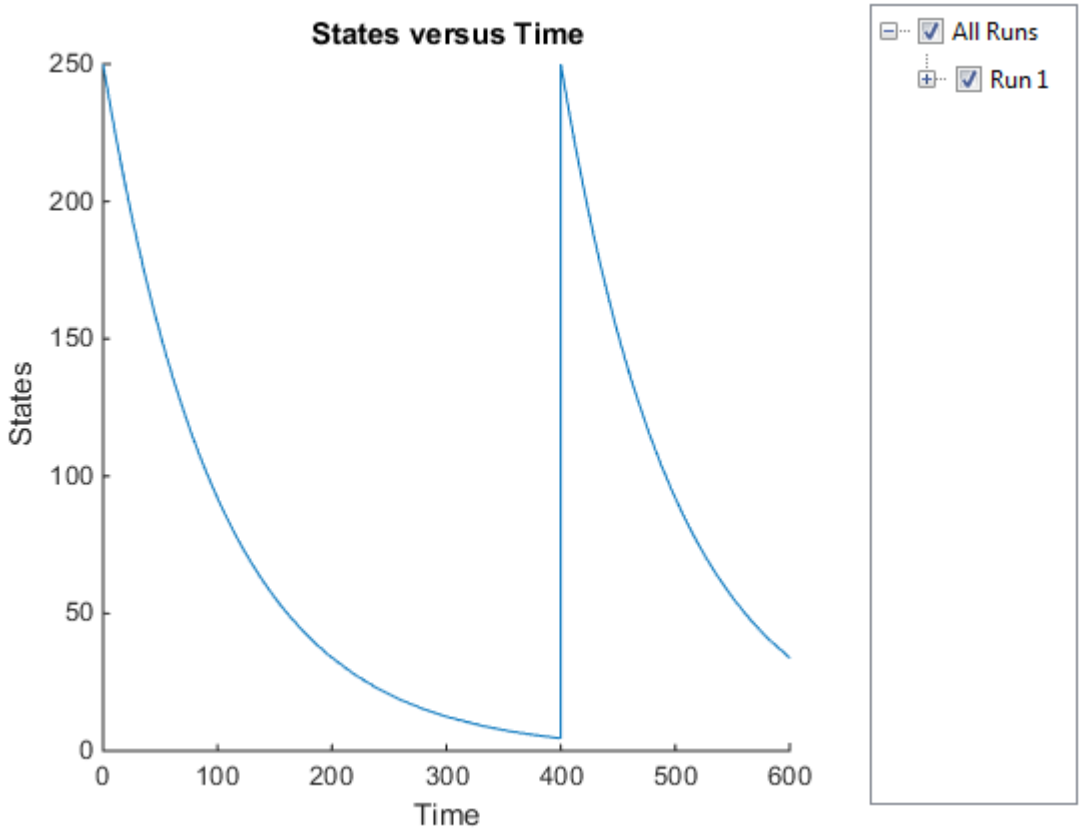
```
GaSimDataObj = selectbyname(simDataObj,'Ga');
sbioplot(GaSimDataObj);
```



Notice the change in the profile of species `Ga` at time = `400` seconds (simulation time). This is the time when the inhibitor amount is changed to reflect the re-addition of inhibitor to the model.

5 Plot only the inhibitor (`Inhib` species).

```
InhibSimDataObj = selectbyname(simDataObj,'Inhib');
sbioplot(InhibSimDataObj)
```



See Also

`addpath` | `cd` | `function`

More About

- “Change Folders on the Search Path” (MATLAB)

N	Number of experimental observations
y_i	The ith experimental observation
	Predicted value of the ith observation
	Standard deviation of the ith observation. • For the constant error model, • For the proportional error model, • For the combined error model,
	The weight of the ith predicted value

When you use numeric weights or the weight function, the weights are assumed to be

inversely proportional to the variance of the error, i.e., where a is the constant error parameter. If you use weights, you cannot specify an error model except the constant error model.

Various optimization methods have different requirements on the function that is being minimized. For some methods, the estimation of model parameters is performed independently of the estimation of the error model parameters. The following table summarizes the error models and any separate formulas used for the estimation of error model parameters, where a and b are error model parameters and e is the standard mean-zero and unit-variance (Gaussian) variable.

Error Model	Error Parameter Estimation Function
`'constant'`: $y_i = f_i + ae$	$a^2 = \frac{1}{N}\sum_i^N (y_i - f_i)^2$
`'exponential'`: $y_i = f_i \exp(ae)$	$a^2 = \frac{1}{N}\sum_i^N (\ln y_i - \ln f_i)^2$
`'proportional'`: $y_i = f_i + b\lvert f_i\rvert e$	$b^2 = \frac{1}{N}\sum_i^N \left(\frac{y_i - f_i}{f_i}\right)^2$
`'combined'`: $y_i = f_i + (a + b\lvert f_i\rvert)e$	Error parameters are included in the minimization.
Weights	$a^2 = \frac{1}{N}\sum_i^N (y_i - f_i)^2 w_i$

Note `nlinfit` only support single error models, not multiple-error models, that is, response-specific error models. For a combined error model, it uses an iterative **WLS** algorithm. For other error models, it uses the **WLS** algorithm as described previously. For details, see `nlinfit`.

Fitting Workflow for sbiofit

The following steps show one of the workflows you can use at the command line to fit a PK model.

1. Import data.
2. Convert the data to the `groupedData` format.
3. Define dosing data. For details, see “Doses” on page 2-42.

4. Create a structural model (one-, two-, or a multicompartment model). For details, see “Create Pharmacokinetic Models” on page 5-24.
5. Map the response variable from data to the model component. For example, if you have the measured drug concentration data for the central compartment, then map it to the drug species in the central compartment (typically the `Drug_Central` species).
6. Specify parameters to estimate using an `estimatedInfo object`. Optionally, you can specify parameter transformations, initial values, and parameter bounds.
7. Perform parameter estimation using `sbiofit`.

For illustrated examples, see the following.

- “Fit a One-Compartment Model to an Individual's PK Profile”
- “Fit a Two-Compartment Model to PK Profiles of Multiple Individuals”
- “Estimate Category-Specific PK Parameters for Multiple Individuals”

See Also

`EstimatedInfo object` | `groupedData` | `sbiofit` | `sbiofitmixed`

More About

- “Multiple Parameter Estimations in Parallel”
- “Parameter Estimation with Hybrid Solvers”
- “Progress Plot” on page 4-61
- “Supported Methods for Parameter Estimation” on page 4-57

Supported Methods for Parameter Estimation

SimBiology supports a variety of optimization methods for least-squares and mixed-effects estimation problems. Depending on the optimization method, you can specify parameter bounds for estimated parameters as well as response-specific error models, that is, an error model for each response variable. The following table summarizes the supported optimization methods in SimBiology, fitting options, and the corresponding toolboxes that are required in addition to MATLAB and SimBiology.

Method	Additional Toolbox Required	Supports Parameter Bounds	Uses Parameter Sensitivities†	Response-specific Error Models	Fixed or Mixed Effects	Supports Stochastic EM Algorithm	SimBiology Function to Use
`fminsearch`	—	Yes*	No	Yes	Fixed	No	`sbiofit`
scattersearch	—	Yes	Depends on the selected local solver.	Depends on the selected local solver.	Fixed	No	
`nlinfit`	Statistics and Machine Learning Toolbox	Yes*	No	No	Fixed	No	
`fminunc`	Optimization Toolbox	Yes*	Yes	Yes	Fixed	No	
`fmincon`	Optimization Toolbox	Yes	Yes	Yes	Fixed	No	
`lsqcurvefit`	Optimization Toolbox	Yes	Yes	Yes	Fixed	No	

Method	Additional Toolbox Required	Supports Parameter Bounds	Uses Parameter Sensitivities†	Response-specific Error Models	Fixed or Mixed Effects	Supports Stochastic EM Algorithm	SimBiology Function to Use
`lsqnonlin`	Optimization Toolbox	Yes	Yes	Yes	Fixed	No	
`patternsearch`	Global Optimization Toolbox	Yes	No	Yes	Fixed	No	
`ga`	Global Optimization Toolbox	Yes	No	Yes	Fixed	No	
`particleswarm`	Global Optimization Toolbox	Yes	No	Yes	Fixed	No	
`nlmefit`	Statistics and Machine Learning Toolbox	No	No	No	Mixed	No	`sbiofitmixed`
`nlmefitsa`	Statistics and Machine Learning Toolbox	No	No	No	Mixed	Yes	

† This column indicates whether the algorithm allows using parameter sensitivities to determine gradients of the objective function.

* When using `fminsearch`, `nlinfit`, or `fminunc` with bounds, the objective function returns `Inf` if bounds are exceeded. When you turn on options such as `FunValCheck`, the optimization may error if bounds are exceeded during estimation. If using `nlinfit`, it

may report warnings about the Jacobian being ill-conditioned or not being able to estimate if the final result is too close to the bounds.

See Also

`sbiofit` | `sbiofitmixed`

More About

- “Nonlinear Regression”
- “Nonlinear Mixed-Effects Modeling”

Error Models

SimBiology supports constant, proportional, combined, and exponential error models.

SimBiology supports the error models described in the following table. For instance, if you assume every observation has a constant amount of noise, use the constant error model, which is the default. Instead, if you assume the error is proportional to the response data, then the proportional error model might be more appropriate.

Error Model	Mathematical Representation	Standard Deviation of Error Model
constant (default)	$y = f + a\varepsilon$	a
proportional	$y = f + b\lvert f\rvert\varepsilon$	$b\lvert f\rvert$
combined	$y = f + (a + b\lvert f\rvert)\varepsilon$	$a+b\lvert f\rvert$
exponential	$y = f * \exp(a\varepsilon)$ or equivalently,	$\sqrt{e^{a^2} - 1} * e^a$
	$\log(y) = \log(f) + a\varepsilon$	a
Here, y is the response, f is the function value, ε is a standard mean-zero and unit-variance (Gaussian) variable, and a and b are error parameters. For instance, if you assume the error is approximately 5% of each observation, use the proportional error model with b = 0.05. In SimBiology, f typically represents the simulation result.		

See Also

More About

- “Nonlinear Regression”
- “Nonlinear Mixed-Effects Modeling”

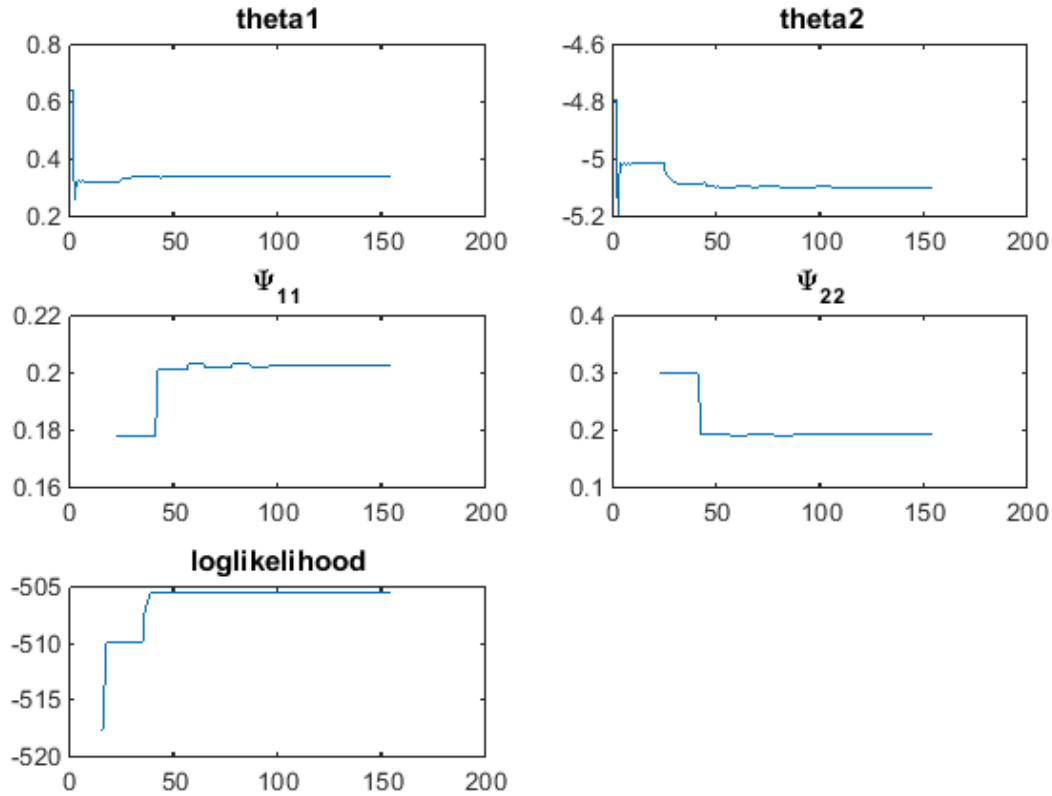
Progress Plot

The progress plot provides the live feedback on the status of parameter estimation while using `sbiofit`, `sbiofitmixed`, or the **Fit Data** task in the SimBiology desktop. When you enable this feature, a new figure opens and shows the fitting quality measures such as log-likelihood and estimated parameter values for each function iteration. The plot monitors the progress whether you are running the fit on a local machine or in parallel using remote clusters.

When you estimate parameters, you can specify which estimation method on page 4-57 to use during the fitting. The progress plot is shown for all the estimation methods except for `nlinfit`. However, the progress plot differs depending on whether you are using a nonlinear mixed-effects method (`nlmefit` or `nlmefitsa`) or a nonlinear regression method, such as `lsqnonlin`.

Progress Plot for Nonlinear Mixed-Effects Methods

The progress plot figure contains a series of subplots. Specifically, the subplots show the values of fixed-effect parameters (theta), the estimates of the variance parameters, that is, the diagonal elements of the covariance matrix of the random effects (Ψ), and the log-likelihood.



Here are some tips for interpreting the plots.

- The fitting function tries to maximize the log-likelihood. When the plot begins to display a flat line, this might indicate that maximization is complete. Try setting the maximum iterations to a lower number to reduce the number of iterations you need and improve performance.
- Plots for the fixed effects (`theta`s) and the variance parameters (Ψs) should show convergence. If you see oscillations, or jumps without accompanying improvements in the log-likelihood, the model may be overparameterized. Try the following:

- Reduce the number of fixed effects.
- Reduce the number of random effects.
- Simplify the covariance matrix pattern of random effects (if you have previously changed it from the default diagonal matrix).

Progress Plot for Nonlinear Regression Methods

The progress plot figure shows a series of subplots, and there are two categories of plots: quality measure plots on page 4-63 and estimated parameter plots on page 4-66. For a pooled fit, that is, estimating one set of parameter values for all groups (or individuals), there is only one line for each plot and the line is faded when the fit is finished. For an unpooled fit, that is, estimating one set of parameter values for each group (or individual), each line represents a single individual or group. You can select one or more lines by clicking and dragging the mouse cursor to create a rectangle on any plot. All lines that intersect the rectangle are selected and highlighted across all plots.

You can terminate the fitting at any time by selecting **Stop**, and partial results are returned. Specifically, for a pooled fit, the result up to the last function iteration is returned. For an unpooled fit, results for any groups that have finished running are returned. The groups currently running are interrupted and partial results from the last iteration are also returned.

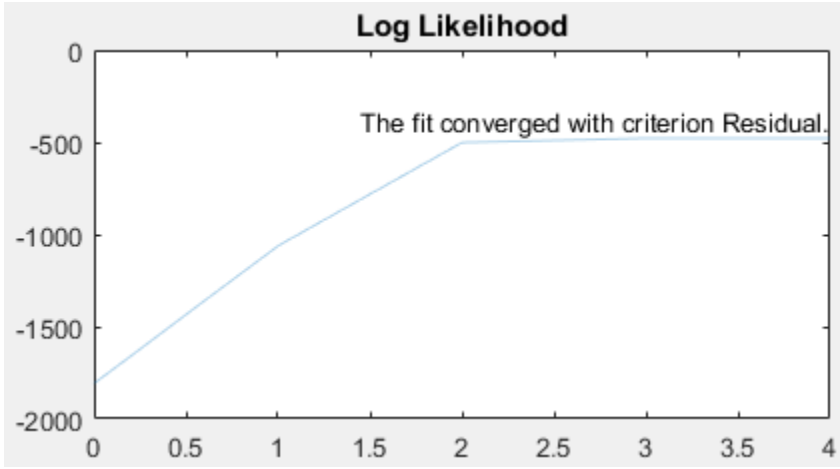
Quality Measure Plots

The quality measure plots include the log-likelihood, first-order optimality, and termination condition plots. They occupy the first row of the figure.

Log-likelihood

The estimation method tries to maximize the log-likelihood, and the plot shows the log-likelihood value for each function iteration. When the plot begins to display a flat line, it often indicates that maximization is complete. Try setting the maximum iterations to a lower number to reduce the number of iterations you need and improve performance.

For a pooled fit, there is only one line in the plot and the line is faded when the fit finishes. The log-likelihood plot shows whether the fit converges or fails along with the information on the estimation method termination condition. The next figure is an example of the log-likelihood plot of a pooled fit.



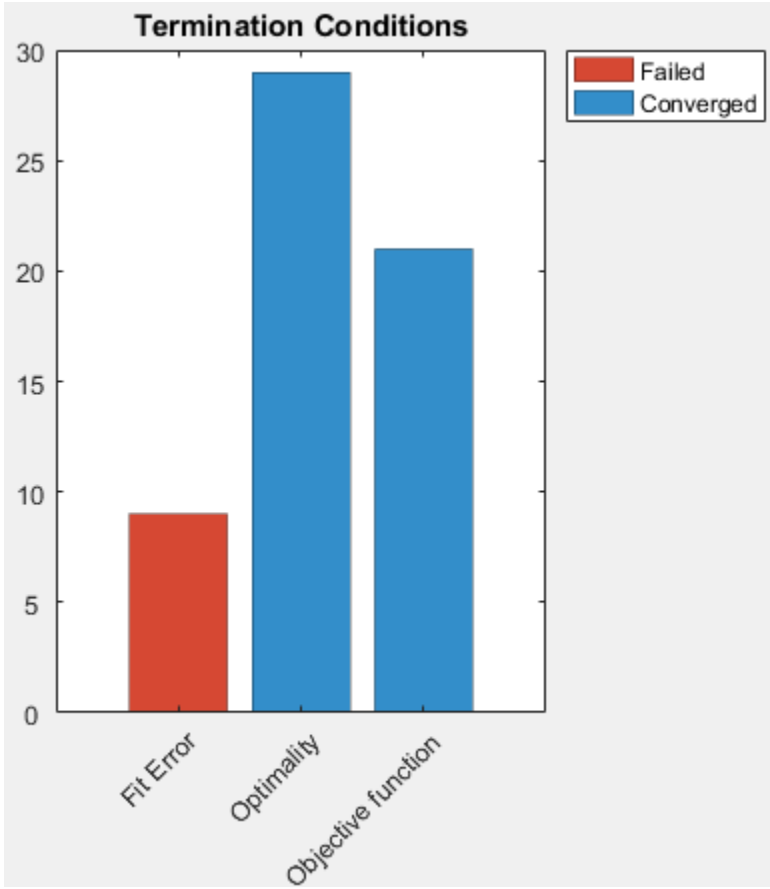
First-order Optimality

First-order optimality is a measure of how close a point x is to optimal, and the plot is shown when you are using the Optimization Toolbox methods (`lsqnonlin`, `lsqcurvefit`, `fminunc`, and `fmincon`). The first-order optimality measure must be zero at a minimum, but a point with first-order optimality equal to zero is not necessarily a minimum. For details, see First-order optimality (Optimization Toolbox).

Termination Condition

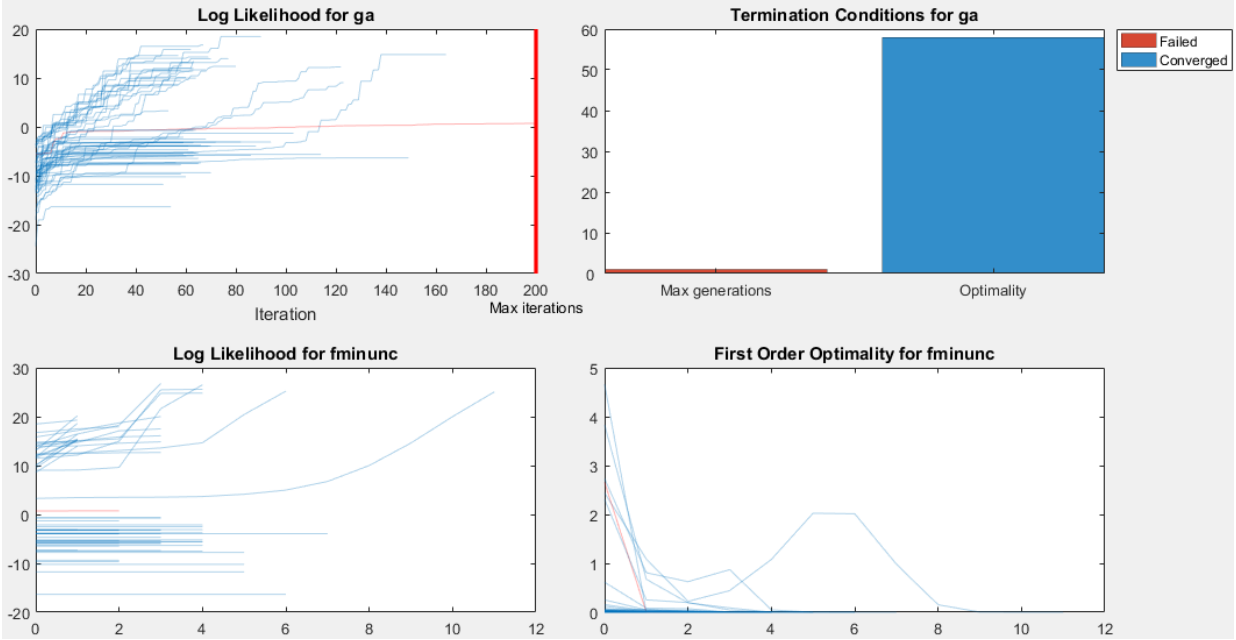
For a pooled fit, the termination condition is displayed together with the log-likelihood plot. For details about the termination condition, refer to the `exitflag` output argument description of the corresponding estimation method. Suppose that you are using the `lsqnonlin` method and see a message: `The fit converged with criterion Residual`. By checking the `exitflag` conditions of the `lsqnonlin` with the keyword `Residual`, this termination condition corresponds to the `exitflag` value of `3`, that is, change in the residual was less than the specified tolerance.

For an unpooled fit, the **Termination Conditions** plot contains the summary (histogram) of termination criteria for all groups (or individuals) as shown in the next figure. The *y*-axis represents the total number of fits for each termination condition, and the *x*-axis displays all the termination criteria.



Hybrid Functions

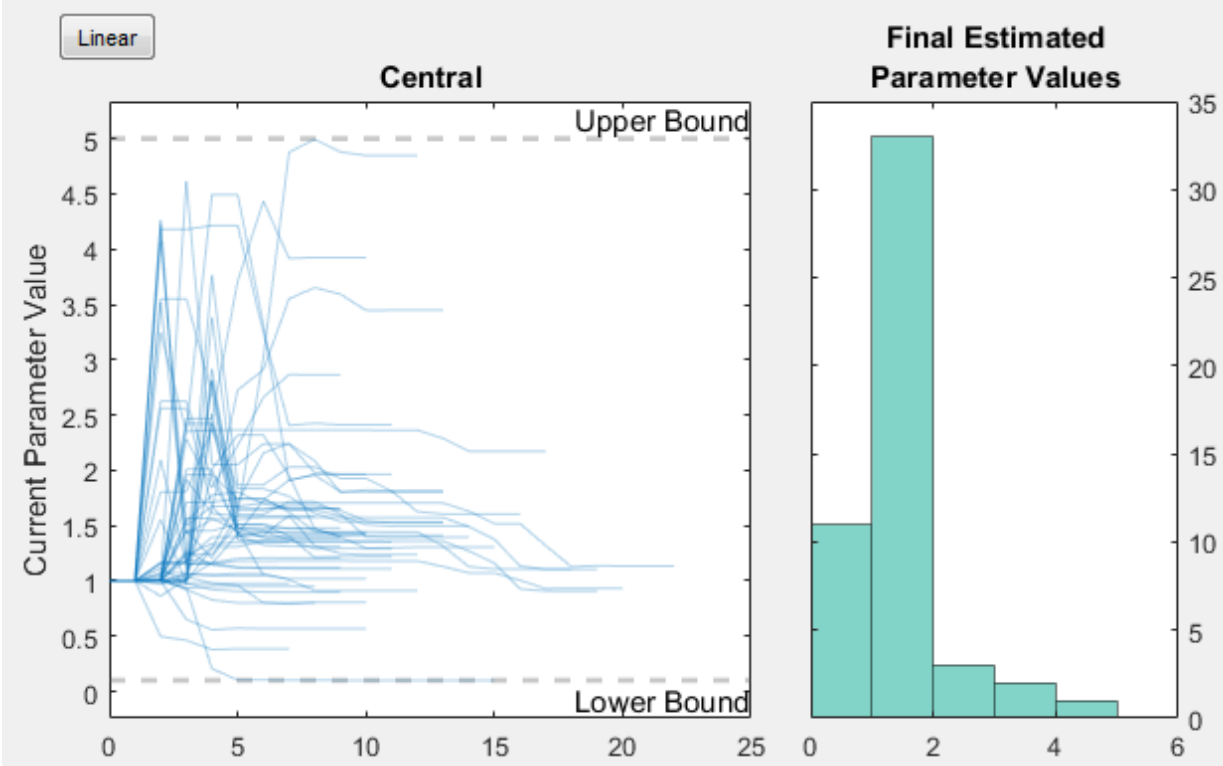
If you are performing a hybrid optimization by first running a global solver, such as `ga` or `particleswarm`, followed by a hybrid function, the `ProgressPlot` also shows the quality measure plots for the hybrid function in the second row. The following figure is an example where the global optimization algorithm is `ga` and the hybrid function is `fminunc`. For an illustrated example, see “Parameter Estimation with Hybrid Solvers”.



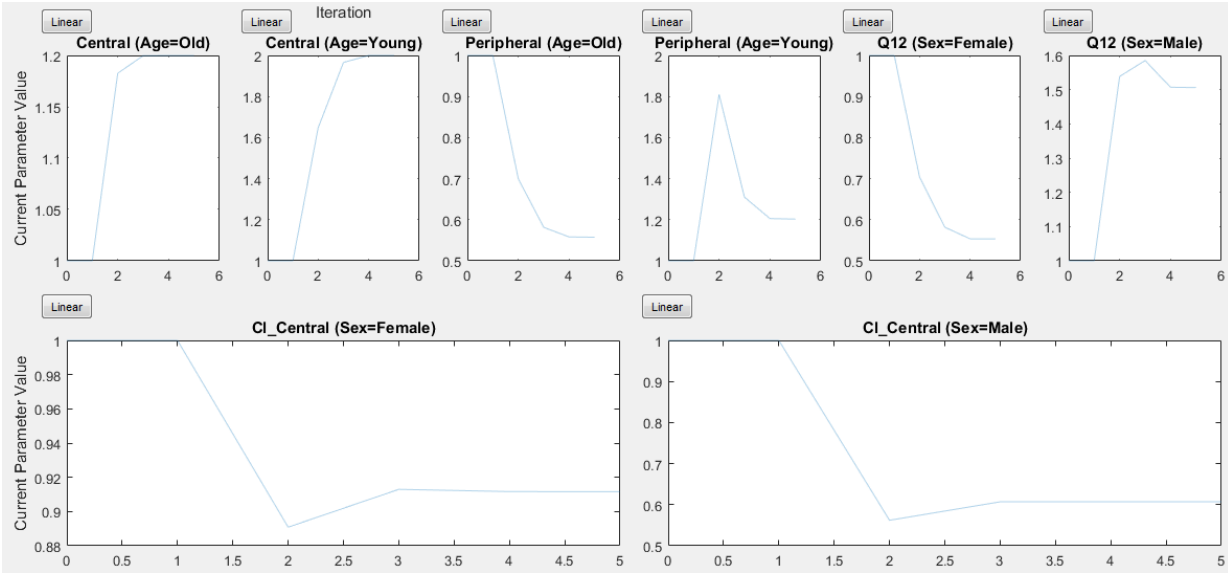
Estimated Parameter Plot

This plot displays the value of the estimated parameter versus iteration for each group. One estimated parameter plot is displayed for each parameter. The plots start on the second row of the figure and can span multiple rows. Each plot displays a horizontal dashed line for any lower or upper bound you specify for the estimated parameter. The bound lines show only if the range of the plot can include the lines.

For an unpooled fit, the Progress Plot also displays a histogram that shows the distribution of the parameter values for the completed runs. Use the toggle button over the *y*-axis for each plot to switch between the log and linear scale. The next figure shows an example of an estimated parameter plot with the bound information and distribution of estimated values.

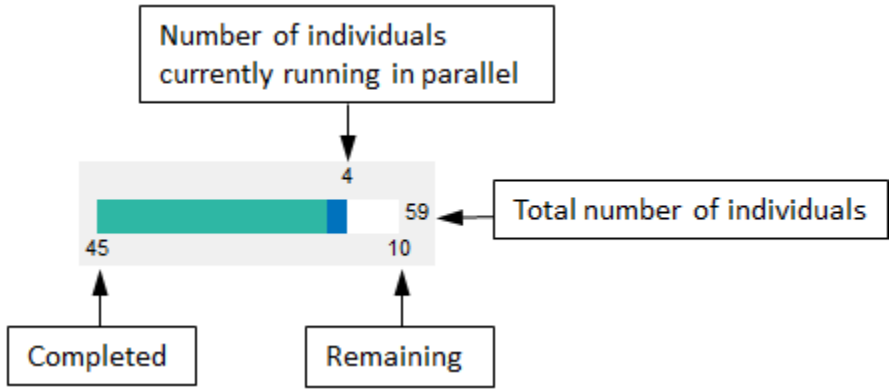


If you have a hierarchical model and are estimating parameters for each category such as estimating parameters for males versus females, the Progress Plot displays one plot per estimated parameter for each category. For example, in the next figure, the *Central* and *Peripheral* parameters are estimated for the age categories while *Q12* and *Cl_Central* are estimated for the sex categories.



Status Bar

For an unpooled fit running in parallel, the Progress Plot displays a status bar in the bottom right corner. The bar shows information about the remaining and completed number of individuals (or groups) throughout the fit.



See Also

`sbiofit` | `sbiofitmixed` | `sbiofitstatusplot`

More About

- “Nonlinear Regression” on page 4-50
- “Nonlinear Mixed-Effects Modeling” on page 4-42
- “Supported Methods for Parameter Estimation” on page 4-57

Fit a One-Compartment Model to an Individual's PK Profile

Background

This example shows how to fit an individual's PK profile data to one-compartment model and estimate pharmacokinetic parameters.

Suppose you have drug plasma concentration data from an individual and want to estimate the volume of the central compartment and the clearance. Assume the drug concentration versus the time profile follows the monoexponential decline $C_t = C_0 e^{-k_e t}$, where C_t is the drug concentration at time t, C_0 is the initial concentration, and k_e is the elimination rate constant that depends on the clearance and volume of the central compartment $k_e = Cl/V$.

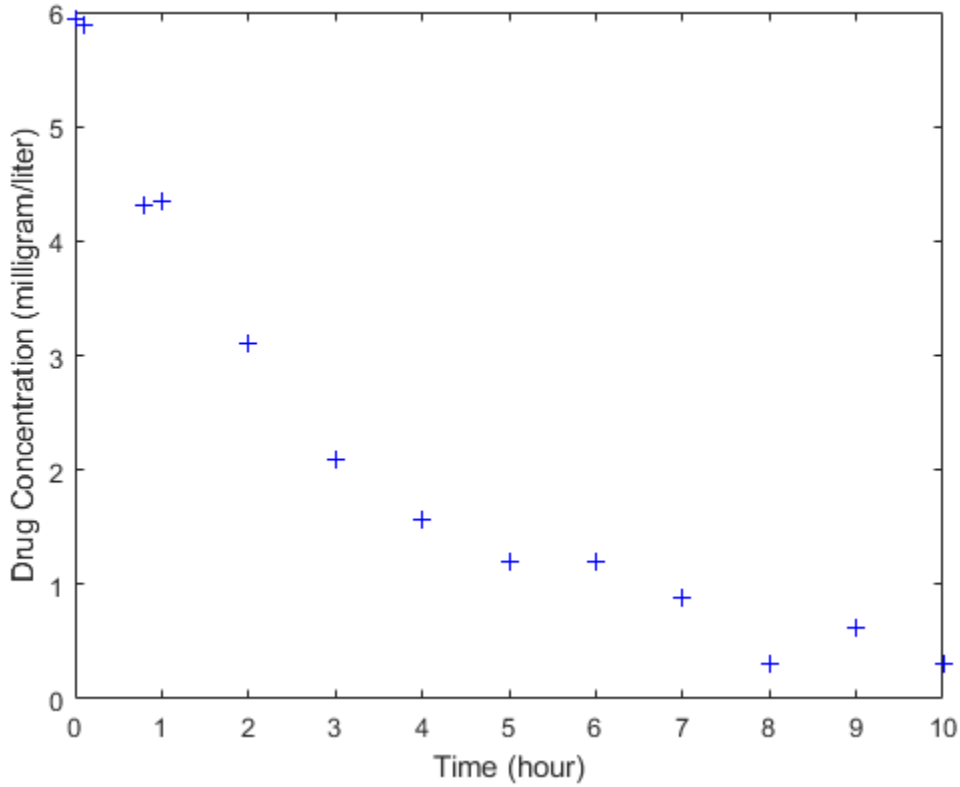
The synthetic data in this example was generated using the following model and parameters:

- One-compartment model with bolus dosing and first-order elimination
- Volume of the central compartment (`Central`) = 1.70 liter
- Clearance parameter (`Cl_Central`) = 0.55 liter/hour
- Constant error model

Load Data and Visualize

The data is stored as a table with variables `Time` and `Conc` that represent the time course of the plasma concentration of an individual after an intravenous bolus administration measured at 13 different time points. The variable units for `Time` and `Conc` are hour and milligram/liter, respectively.

```
clear all
load(fullfile(matlabroot,'examples','simbio','data15.mat'))

plot(data.Time,data.Conc,'b+')
xlabel('Time (hour)');
ylabel('Drug Concentration (milligram/liter)');
```



Convert to groupedData Format

Convert the data set to a `groupedData` object, which is the required data format for the fitting function `sbiofit` for later use. A `groupedData` object also lets you set independent variable and group variable names (if they exist). Set the units of the `Time` and `Conc` variables. The units are optional and only required for the `UnitConversion` feature, which automatically converts matching physical quantities to one consistent unit system.

```
gData = groupedData(data);
gData.Properties.VariableUnits = {'hour','milligram/liter'};
gData.Properties
```

```
ans = struct with fields:
                Description: ''
                   UserData: []
             DimensionNames: {'Row'  'Variables'}
              VariableNames: {'Time'  'Conc'}
       VariableDescriptions: {}
              VariableUnits: {'hour'  'milligram/liter'}
         VariableContinuity: []
                   RowNames: {}
          GroupVariableName: ''
    IndependentVariableName: 'Time'
```

`groupedData` automatically set the name of the `IndependentVariableName` property to the `Time` variable of the data.

Construct a One-Compartment Model

Use the built-in PK library to construct a one-compartment model with bolus dosing and first-order elimination where the elimination rate depends on the clearance and volume of the central compartment. Use the `configset` object to turn on unit conversion.

```
pkmd                    = PKModelDesign;
pkc1                    = addCompartment(pkmd,'Central');
pkc1.DosingType         = 'Bolus';
pkc1.EliminationType    = 'linear-clearance';
pkc1.HasResponseVariable = true;
model                   = construct(pkmd);
configset               = getconfigset(model);
configset.CompileOptions.UnitConversion = true;
```

For details on creating compartmental PK models using the SimBiology® built-in library, see “Create Pharmacokinetic Models” on page 5-24.

Define Dosing

Define a single bolus dose of 10 milligram given at time = 0. For details on setting up different dosing schedules, see “Doses” on page 2-42.

```
dose                = sbiodose('dose');
dose.TargetName     = 'Drug_Central';
dose.StartTime      = 0;
dose.Amount         = 10;
dose.AmountUnits    = 'milligram';
dose.TimeUnits      = 'hour';
```

Map Response Data to the Corresponding Model Component

The data contains drug concentration data stored in the `Conc` variable. This data corresponds to the `Drug_Central` species in the model. Therefore, map the data to `Drug_Central` as follows.

```
responseMap = {'Drug_Central = Conc'};
```

Specify Parameters to Estimate

The parameters to fit in this model are the volume of the central compartment (Central) and the clearance rate (Cl_Central). In this case, specify log-transformation for these biological parameters since they are constrained to be positive. The `estimatedInfo` object lets you specify parameter transforms, initial values, and parameter bounds if needed.

```
paramsToEstimate    = {'log(Central)','log(Cl_Central)'};
estimatedParams     = estimatedInfo(paramsToEstimate,'InitialValue',[1 1],'Bounds',[1 5
```

Estimate Parameters

Now that you have defined one-compartment model, data to fit, mapped response data, parameters to estimate, and dosing, use `sbiofit` to estimate parameters. The default estimation function that `sbiofit` uses will change depending on which toolboxes are available. To see which function was used during fitting, check the `EstimationFunction` property of the corresponding results object.

```
fitConst = sbiofit(model,gData,responseMap,estimatedParams,dose);
```

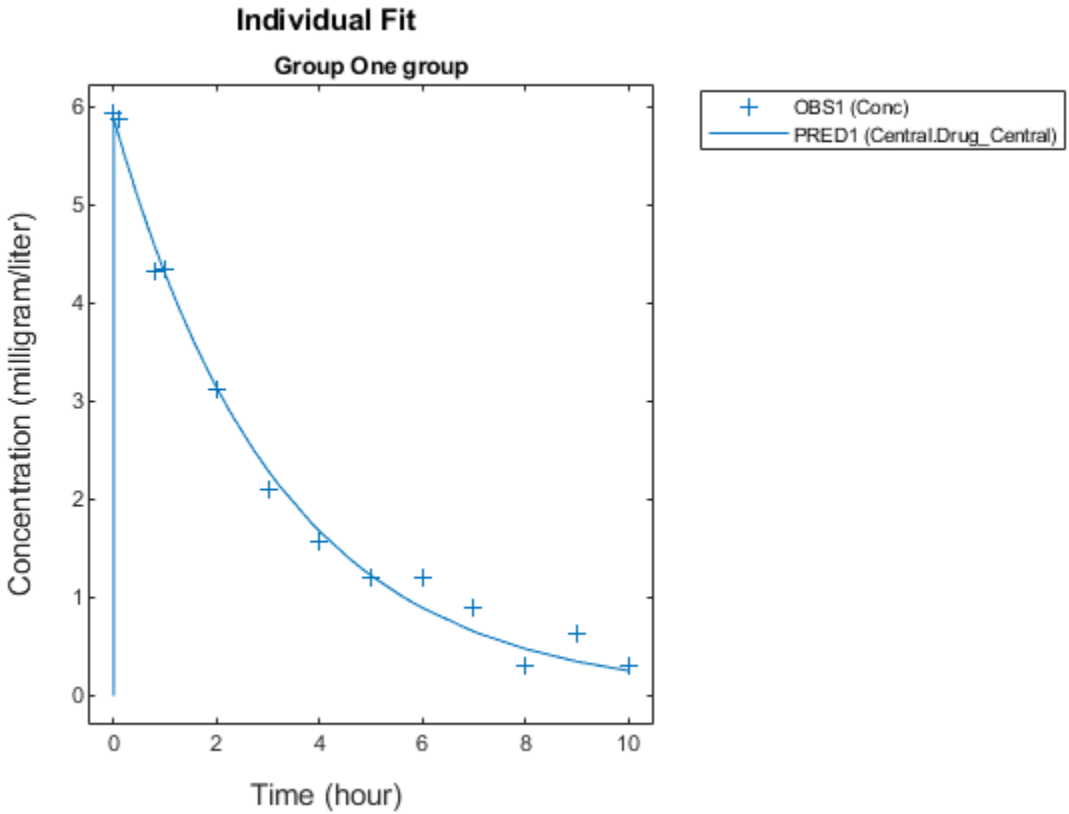
Display Estimated Parameters and Plot Results

Notice the parameter estimates were not far off from the true values (1.70 and 0.55) that were used to generate the data. You may also try different error models to see if they could further improve the parameter estimates.

```
fitConst.ParameterEstimates
```

```
ans=2×4 table
        Name         Estimate    StandardError      Bounds
    _____________    ________    _____________    __________

    'Central'         1.6993       0.034821          1      5
    'Cl_Central'     0.53358        0.01968        0.5      2
```

```
s.Labels.XLabel     = 'Time (hour)';
s.Labels.YLabel     = 'Concentration (milligram/liter)';
plot(fitConst,'AxesStyle',s);
```



Use Different Error Models

Try three other supported error models (proportional, combination of constant and proportional error models, and exponential).

```
fitProp = sbiofit(model,gData,responseMap,estimatedParams,dose,...
                      'ErrorModel','proportional');
fitExp  = sbiofit(model,gData,responseMap,estimatedParams,dose,...
                      'ErrorModel','exponential');
```

```
fitComb = sbiofit(model,gData,responseMap,estimatedParams,dose,...
                      'ErrorModel','combined');
```

Use Weights Instead of an Error Model

You can specify weights as a numeric matrix, where the number of columns corresponds to the number of responses. Setting all weights to 1 is equivalent to the constant error model.

```
weightsNumeric = ones(size(gData.Conc));
fitWeightsNumeric = sbiofit(model,gData,responseMap,estimatedParams,dose,'Weights',weig
```

Alternatively, you can use a function handle that accepts a vector of predicted response values and returns a vector of weights. In this example, use a function handle that is equivalent to the proportional error model.

```
weightsFunction = @(y) 1./y.^2;
fitWeightsFunction = sbiofit(model,gData,responseMap,estimatedParams,dose,'Weights',wei
```

Compare Information Criteria for Model Selection

Compare the loglikelihood, AIC, and BIC values of each model to see which error model best fits the data. A larger likelihood value indicates the corresponding model fits the model better. For AIC and BIC, the smaller values are better.

```
allResults = [fitConst,fitWeightsNumeric,fitWeightsFunction,fitProp,fitExp,fitComb];
errorModelNames = {'constant error model','equal weights','proportional weights', ...
                   'proportional error model','exponential error model',...
                   'combined error model'};
LogLikelihood = [allResults.LogLikelihood]';
AIC = [allResults.AIC]';
BIC = [allResults.BIC]';
t = table(LogLikelihood,AIC,BIC);
t.Properties.RowNames = errorModelNames;
t
```

```
t=6×3 table
                                LogLikelihood      AIC        BIC
                                _____________    _______    _______

    constant error model            3.9866       -3.9732    -2.8433
    equal weights                   3.9866       -3.9732    -2.8433
    proportional weights           -3.8472        11.694     12.824
    proportional error model       -3.8257        11.651     12.781
    exponential error model         1.1984        1.6032     2.7331
```

```
    combined error model            3.9163       -3.8326    -2.7027
```

Based on the information criteria, the constant error model (or equal weights) fits the data best since it has the largest loglikelihood value and the smallest AIC and BIC.

Display Estimated Parameter Values

Show the estimated parameter values of each model.

```
Estimated_Central       = zeros(6,1);
Estimated_Cl_Central    = zeros(6,1);
t2 = table(Estimated_Central,Estimated_Cl_Central);
t2.Properties.RowNames = errorModelNames;
for i = 1:height(t2)
    t2{i,1} = allResults(i).ParameterEstimates.Estimate(1);
    t2{i,2} = allResults(i).ParameterEstimates.Estimate(2);
end
t2
```

```
t2=6×2 table
                              Estimated_Central    Estimated_Cl_Central
                              _________________    ____________________

    constant error model            1.6993               0.53358
    equal weights                   1.6993               0.53358
    proportional weights            1.9045               0.51734
    proportional error model        1.8777               0.51147
    exponential error model         1.7872               0.51701
    combined error model            1.7008               0.53271
```

Conclusion

This example showed how to estimate PK parameters, namely the volume of the central compartment and clearance parameter of an individual, by fitting the PK profile data to one-compartment model. You compared the information criteria of each model and estimated parameter values of different error models to see which model best explained the data. Final fitted results suggested both the constant and combined error models provided the closest estimates to the parameter values used to generate the data.

However, the constant error model is a better model as indicated by the loglikelihood, AIC, and BIC information criteria.

See Also

`sbiofit`

More About

- “Fit a Two-Compartment Model to PK Profiles of Multiple Individuals” on page 4-96
- “Estimate Category-Specific PK Parameters for Multiple Individuals” on page 4-78
- “Nonlinear Regression” on page 4-50

Estimate Category-Specific PK Parameters for Multiple Individuals

This example shows how to estimate category-specific (such as young versus old, male versus female), individual-specific, and population-wide parameters using PK profile data from multiple individuals.

Background

Suppose you have drug plasma concentration data from 30 individuals and want to estimate pharmacokinetic parameters, namely the volumes of central and peripheral compartment, the clearance, and intercompartmental clearance. Assume the drug concentration versus the time profile follows the biexponential decline $C_t = Ae^{-at} + Be^{-bt}$, where C_t is the drug concentration at time t, and a and b are slopes for corresponding exponential declines.

Load Data

This synthetic data contains the time course of plasma concentrations of 30 individuals after a bolus dose (100 mg) measured at different times for both central and peripheral compartments. It also contains categorical variables, namely `Sex` and `Age`.

```
clear
load(fullfile(matlabroot,'examples','simbio','sd5_302RAgeSex.mat'))
```

Convert to groupedData Format

Convert the data set to a `groupedData` object, which is the required data format for the fitting function `sbiofit`. A `groupedData` object also allows you set independent variable and group variable names (if they exist). Set the units of the `ID`, `Time`, `CentralConc`, `PeripheralConc`, `Age`, and `Sex` variables. The units are optional and only required for the `UnitConversion` feature, which automatically converts matching physical quantities to one consistent unit system.

```
gData = groupedData(data);
gData.Properties.VariableUnits = {'','hour','milligram/liter','milligram/liter','',''};
gData.Properties
```

```
ans = struct with fields:
                Description: ''
                   UserData: []
```

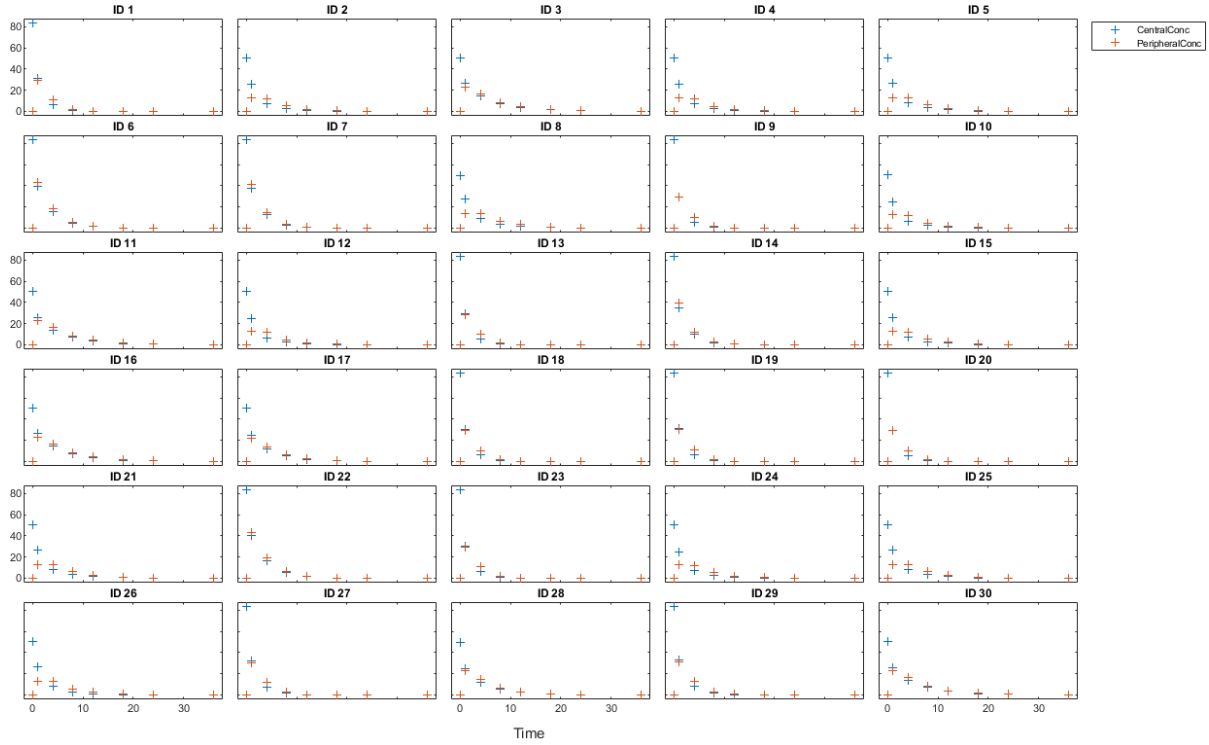
```
            DimensionNames: {'Row'  'Variables'}
             VariableNames: {1x6 cell}
       VariableDescriptions: {}
              VariableUnits: {1x6 cell}
         VariableContinuity: []
                   RowNames: {}
          GroupVariableName: 'ID'
    IndependentVariableName: 'Time'
```

The `IndependentVariableName` and `GroupVariableName` properties have been automatically set to the `Time` and `ID` variables of the data.

Visualize Data

Display the response data for each individual.

```
t = sbiotrellis(gData,'ID','Time',{'CentralConc','PeripheralConc'},...
                'Marker','+','LineStyle','none');
% Resize the figure.
t.hFig.Position(:) = [100 100 1280 800];
```



Set Up a Two-Compartment Model

Use the built-in PK library to construct a two-compartment model with infusion dosing and first-order elimination where the elimination rate depends on the clearance and volume of the central compartment. Use the `configset` object to turn on unit conversion.

```
pkmd                                    = PKModelDesign;
pkc1                                    = addCompartment(pkmd,'Central');
pkc1.DosingType                         = 'Bolus';
pkc1.EliminationType                    = 'linear-clearance';
pkc1.HasResponseVariable                = true;
pkc2                                    = addCompartment(pkmd,'Peripheral');
model                                   = construct(pkmd);
configset                               = getconfigset(model);
configset.CompileOptions.UnitConversion = true;
```

For details on creating compartmental PK models using the SimBiology® built-in library, see “Create Pharmacokinetic Models” on page 5-24.

Define Dosing

Assume every individual receives a bolus dose of 100 mg at time = 0. For details on setting up different dosing strategies, see “Doses” on page 2-42.

```
dose             = sbiodose('dose','TargetName','Drug_Central');
dose.StartTime   = 0;
dose.Amount      = 100;
dose.AmountUnits = 'milligram';
dose.TimeUnits   = 'hour';
```

Map the Response Data to Corresponding Model Components

The data contains measured plasma concentration in the central and peripheral compartments. Map these variables to the appropriate model components, which are `Drug_Central` and `Drug_Peripheral`.

```
responseMap = {'Drug_Central = CentralConc','Drug_Peripheral = PeripheralConc'};
```

Specify Parameters to Estimate

Specify the volumes of central and peripheral compartments `Central` and `Peripheral`, intercompartmental clearance `Q12`, and clearance `Cl_Central` as parameters to estimate. The `estimatedInfo` object lets you optionally specify parameter transforms, initial values, and parameter bounds. Since both `Central` and `Peripheral` are constrained to be positive, specify a log-transform for each parameter.

```
paramsToEstimate    = {'log(Central)', 'log(Peripheral)', 'Q12', 'Cl_Central'};
estimatedParam      = estimatedInfo(paramsToEstimate,'InitialValue',[1 1 1 1]);
```

Estimate Individual-Specific Parameters

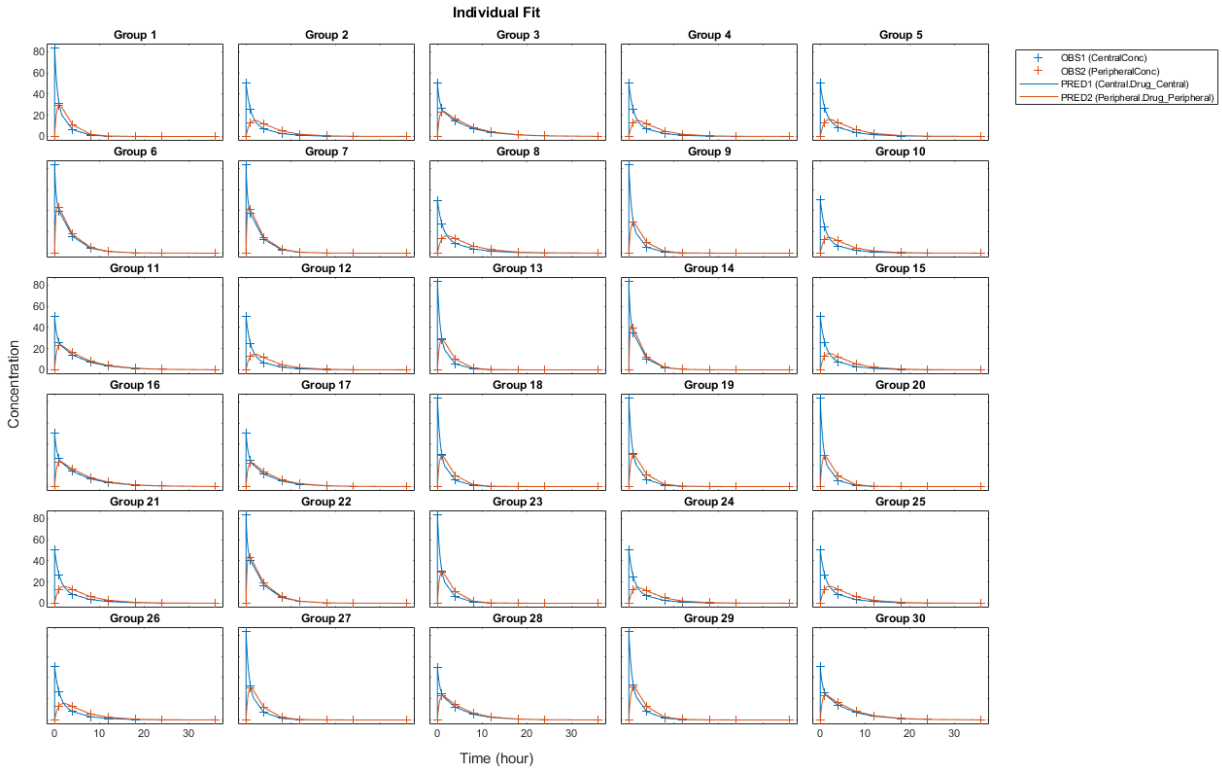
Estimate one set of parameters for each individual by setting the `'Pooled'` name-value pair argument to `false`.

```
unpooledFit =  sbiofit(model,gData,responseMap,estimatedParam,dose,'Pooled',false);
```

Display Results

Plot the fitted results versus the original data for each individual (group).

```
t = plot(unpooledFit);
% Resize the figure.
t.hFig.Position(:) = [100 100 1280 800];
```



For an unpooled fit, `sbiofit` always returns one results object for each individual.

Examine Parameter Estimates for Category Dependencies

Explore the unpooled estimates to see if there is any category-specific parameters, that is, if some parameters are related to one or more categories. If there are any category dependencies, it might be possible to reduce the number of degrees of freedom by estimating just category-specific values for those parameters.

First extract the ID and category values for each ID

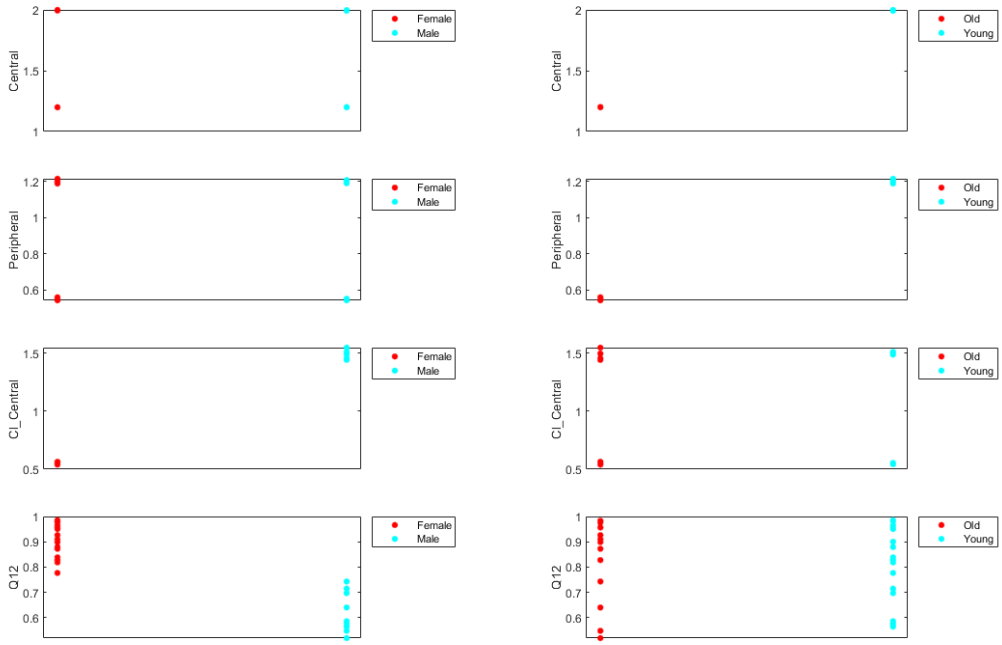
```
catParamValues = unique(gData(:,{'ID','Sex','Age'}));
```

Add variables to the table containing each parameter's estimate.

```
allParamValues            = vertcat(unpooledFit.ParameterEstimates);
catParamValues.Central    = allParamValues.Estimate(strcmp(allParamValues.Name, 'Centra
catParamValues.Peripheral = allParamValues.Estimate(strcmp(allParamValues.Name, 'Periph
catParamValues.Q12        = allParamValues.Estimate(strcmp(allParamValues.Name, 'Q12'))
catParamValues.Cl_Central = allParamValues.Estimate(strcmp(allParamValues.Name, 'Cl_Cen
```

Plot estimates of each parameter for each category. `gscatter` requires Statistics and Machine Learning Toolbox™. If you do not have it, use other alternative plotting functions such as `plot`.

```
h           = figure;
ylabels     = {'Central','Peripheral','Cl\_Central','Q12'};
plotNumber  = 1;
for i = 1:4
    thisParam = estimatedParam(i).Name;

    % Plot for Sex category
    subplot(4,2,plotNumber);
    plotNumber  = plotNumber + 1;
    gscatter(double(catParamValues.Sex), catParamValues.(thisParam), catParamValues.Sex
    ax          = gca;
    ax.XTick    = [];
    ylabel(ylabels(i));
    legend('Location','bestoutside')
    % Plot for Age category
    subplot(4,2,plotNumber);
    plotNumber  = plotNumber + 1;
    gscatter(double(catParamValues.Age), catParamValues.(thisParam), catParamValues.Age
    ax          = gca;
    ax.XTick    = [];
    ylabel(ylabels(i));
    legend('Location','bestoutside')
end
% Resize the figure.
h.Position(:) = [100 100 1280 800];
```



Based on the plot, it seems that young individuals tend to have higher volumes of central and peripheral compartments (`Central`, `Peripheral`) than old invididuals (that is, the volumes seem to be age-specific). In addition, males tend to have lower clearance rates (`Cl_Central`) than females but the opposite for the Q12 parameter (that is, the clearance and Q12 seem to be sex-specific).

Estimate Category-Specific Parameters

Use the `'CategoryVariableName'` property of the `estimatedInfo` object to specify which category to use during fitting. Use `'Sex'` as the group to fit for the clearrance `Cl_Central` and `Q12` parameters. Use `'Age'` as the group to fit for the `Central` and `Peripheral` parameters.

```
estimatedParam(1).CategoryVariableName = 'Age';
estimatedParam(2).CategoryVariableName = 'Age';
estimatedParam(3).CategoryVariableName = 'Sex';
```

```
estimatedParam(4).CategoryVariableName = 'Sex';
categoryFit = sbiofit(model,gData,responseMap,estimatedParam,dose)
```

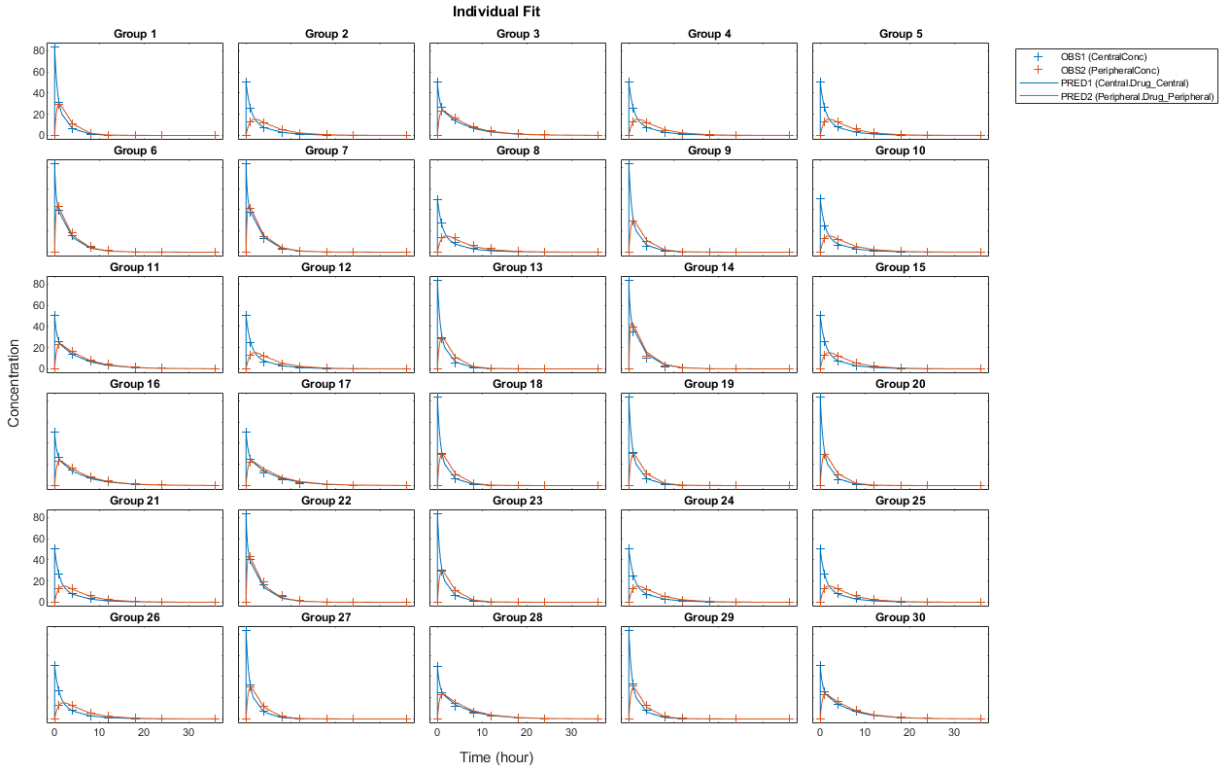
```
categoryFit =
  OptimResults with properties:

                  ExitFlag: 3
                    Output: [1x1 struct]
                 GroupName: []
                      Beta: [8x5 table]
        ParameterEstimates: [120x6 table]
                         J: [240x8x2 double]
                      COVB: [8x8 double]
          CovarianceMatrix: [8x8 double]
                         R: [240x2 double]
                       MSE: 0.4362
                       SSE: 205.8690
                   Weights: []
              LogLikelihood: -477.9195
                       AIC: 971.8390
                       BIC: 1.0052e+03
                       DFE: 472
            DependentFiles: {1x3 cell}
    EstimatedParameterNames: {'Central'  'Peripheral'  'Q12'  'Cl_Central'}
             ErrorModelInfo: [1x3 table]
         EstimationFunction: 'lsqnonlin'
```

When fitting by category (or group), `sbiofit` always returns one results object, not one for each category level. This is because both male and female individuals are considered to be part of the same optimization using the same error model and error parameters, similarly for the young and old individuals.

Plot Results

Plot the category-specific estimated results.

```
t = plot(categoryFit);
% Resize the figure.
t.hFig.Position(:) = [100 100 1280 800];
```



For the `Cl_Central` and `Q12` parameters, all males had the same estimates, and similarly for the females. For the `Central` and `Peripheral` parameters, all young individuals had the same estimates, and similarly for the old individuals.

Estimate Population-Wide Parameters

To better compare the results, fit the model to all of the data pooled together, that is, estimate one set of parameters for all individuals by setting the `'Pooled'` name-value pair argument to `true`. The warning message tells you that this option will ignore any category-specific information (if they exist).

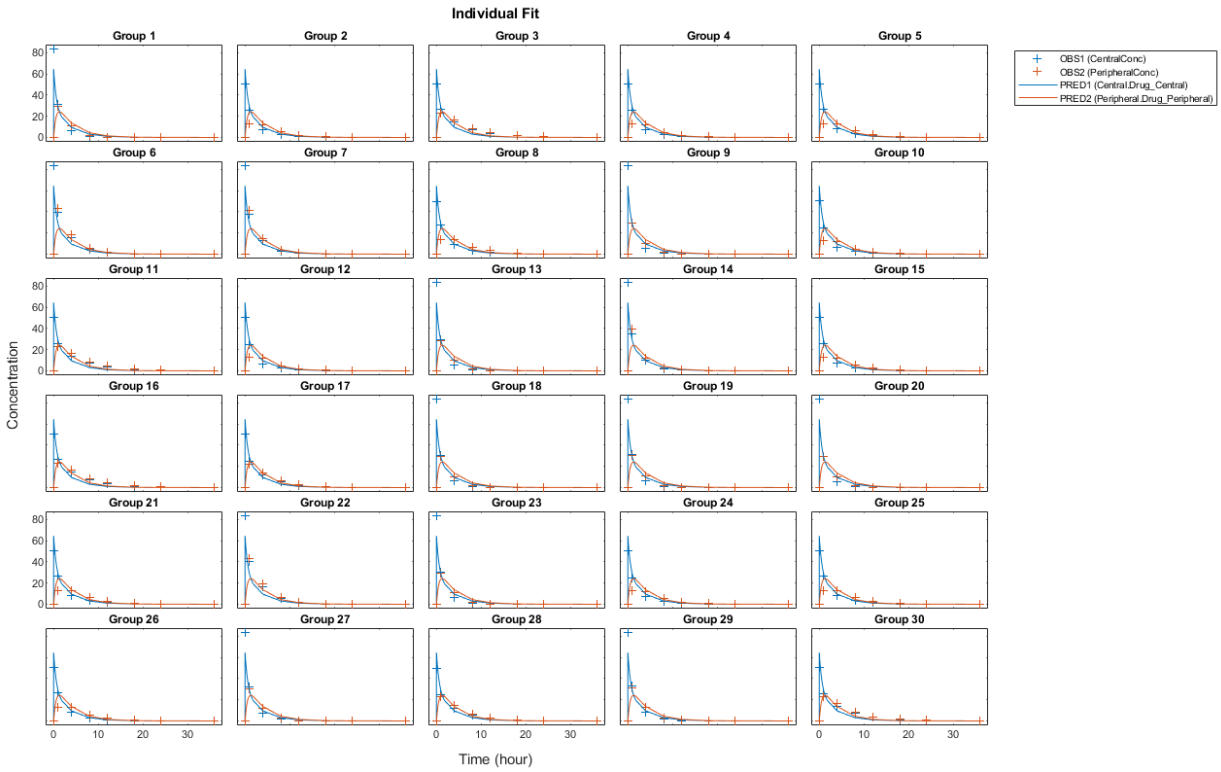
```
pooledFit = sbiofit(model,gData,responseMap,estimatedParam,dose,'Pooled',true);
```

```
Warning: CategoryVariableName property of the estimatedInfo object is ignored when usi
```

Plot Results

Plot the fitted results versus the original data. Although a separate plot was generated for each individual, the data was fitted using the same set of parameters (that is, all individuals had the same fitted line).

```
t = plot(pooledFit);
% Resize the figure.
t.hFig.Position(:) = [100 100 1280 800];
```

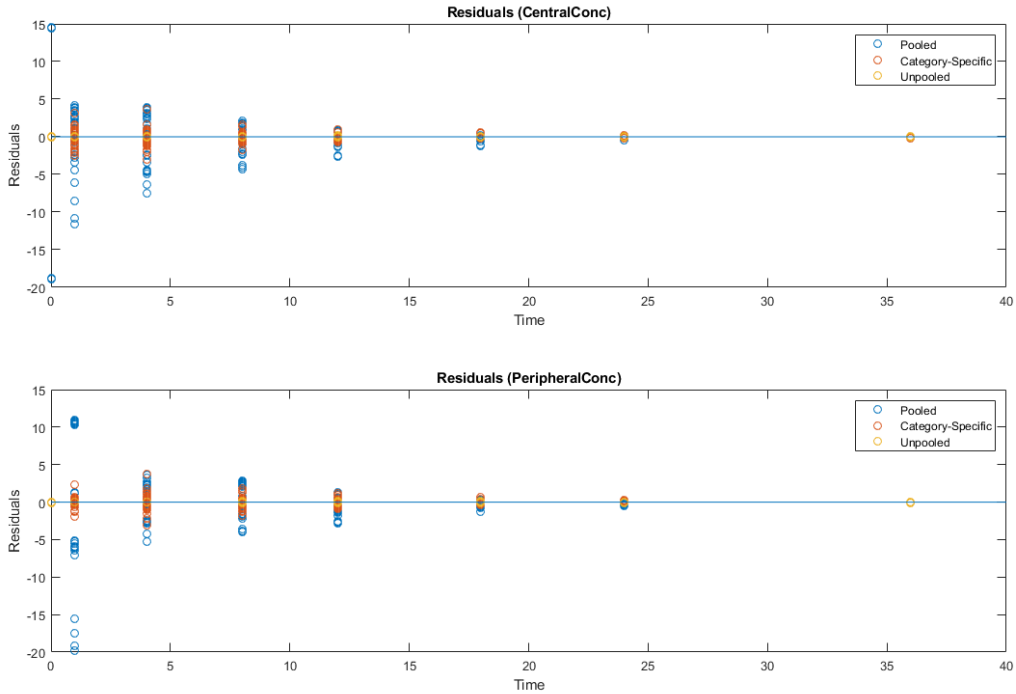


Compare Residuals

Compare residuals of `CentralConc` and `PeripheralConc` responses for each fit.

```
t = gData.Time;
allResid(:,:,1) = pooledFit.R;
```

```
allResid(:,:,2) = categoryFit.R;
allResid(:,:,3) = vertcat(unpooledFit.R);

h = figure;
responseList = {'CentralConc', 'PeripheralConc'};
for i = 1:2
    subplot(2,1,i);
    oneResid = squeeze(allResid(:,i,:));
    plot(t,oneResid,'o');
    refline(0,0); % A reference line representing a zero residual
    title(sprintf('Residuals (%s)', responseList{i}));
    xlabel('Time');
    ylabel('Residuals');
    legend({'Pooled','Category-Specific','Unpooled'});
end
% Resize the figure.
h.Position(:) = [100 100 1280 800];
```



As shown in the plot, the unpooled fit produced the best fit to the data as it fit the data to each indivdual. This was expected since it used the most number of degrees of freedom. The category-fit reduced the number of degrees of freedom by fitting the data to two categories (sex and age). As a result, the residuals were larger than the unpooled fit, but still smaller than the population-fit, which estimated just one set of parameters for all individuals. The category-fit might be a good compromise between the unpooled and pooled fitting provided that any hierarchical model exists within your data.

See Also

`sbiofit`

More About

- “Fit a One-Compartment Model to an Individual's PK Profile” on page 4-70
- “Fit a Two-Compartment Model to PK Profiles of Multiple Individuals” on page 4-96
- “Nonlinear Regression” on page 4-50

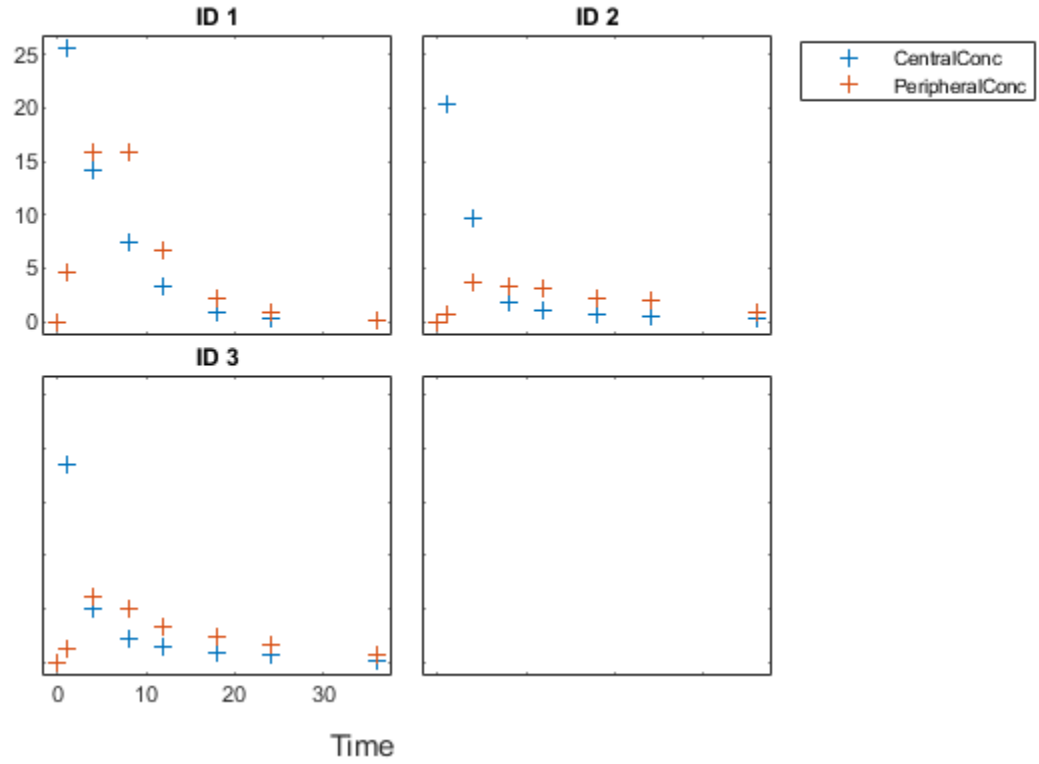
Perform Hybrid Optimization Using sbiofit

This example shows how to configure `sbiofit` to perform a hybrid optimization by first running the global solver `particleswarm`, followed by another minimization function, `fmincon`.

Load Data

Load the sample data to fit. The data is stored as a table with variables *ID*, *Time*, *CentralConc*, and *PeripheralConc*. This synthetic data represents the time course of plasma concentrations measured at eight different time points for both central and peripheral compartments after an infusion dose for three individuals.

```
clear all
load(fullfile(matlabroot,'examples','simbio','data10_32R.mat'))
gData = groupedData(data);
gData.Properties.VariableUnits = {'','hour','milligram/liter','milligram/liter'};
sbiotrellis(gData,'ID','Time',{'CentralConc','PeripheralConc'},'Marker','+',...
            'LineStyle','none');
```



Create Model

Create a two-compartment model with an infusion dose.

```
pkmd                 = PKModelDesign;
pkc1                 = addCompartment(pkmd,'Central');
pkc1.DosingType      = 'Infusion';
pkc1.EliminationType = 'linear-clearance';
pkc1.HasResponseVariable = true;
pkc2                 = addCompartment(pkmd,'Peripheral');
model                = construct(pkmd);
configset            = getconfigset(model);
configset.CompileOptions.UnitConversion = true;
dose             = sbiodose('dose','TargetName','Drug_Central');
```

```
dose.StartTime   = 0;
dose.Amount      = 100;
dose.Rate        = 50;
dose.AmountUnits = 'milligram';
dose.TimeUnits   = 'hour';
dose.RateUnits   = 'milligram/hour';
responseMap = {'Drug_Central = CentralConc','Drug_Peripheral = PeripheralConc'};
```

Define Parameters to Estimate

Use the `estimatedInfo` object to define the estimated parameters.

```
paramsToEstimate   = {'log(Central)','log(Peripheral)','Q12','Cl_Central'};
estimatedParam     = estimatedInfo(paramsToEstimate,'InitialValue',[1 1 1 1],...
                                   'Bounds',[0 10]);
```

Define the Options for Hybrid Optimization

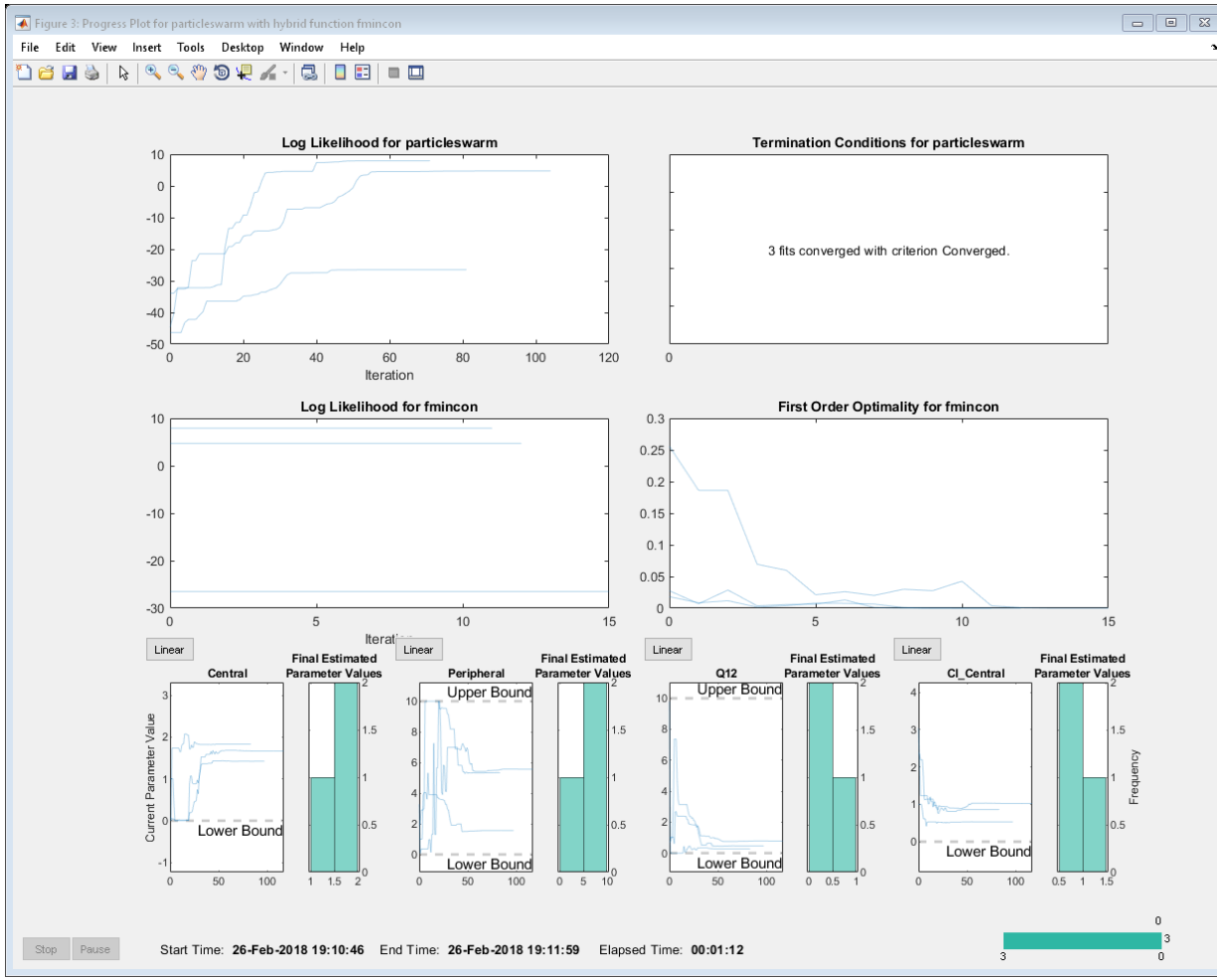
Define the options for the global solver and the hybrid solver. Because the parameters are bounded, make sure you use a compatible hybrid function for a constrained optimization, such as `fmincon`. Use `optimset` to define the options for `fminsearch`. Use `optimoptions` to define the options for `fminunc`, `patternsearch`, and `fmincon`.

```
rng('default');
globalMethod = 'particleswarm';
options = optimoptions(globalMethod);
hybridMethod = 'fmincon';
hybridopts = optimoptions(hybridMethod,'Display','none');
options = optimoptions(options,'HybridFcn',{hybridMethod,hybridopts});
```

Fit Data

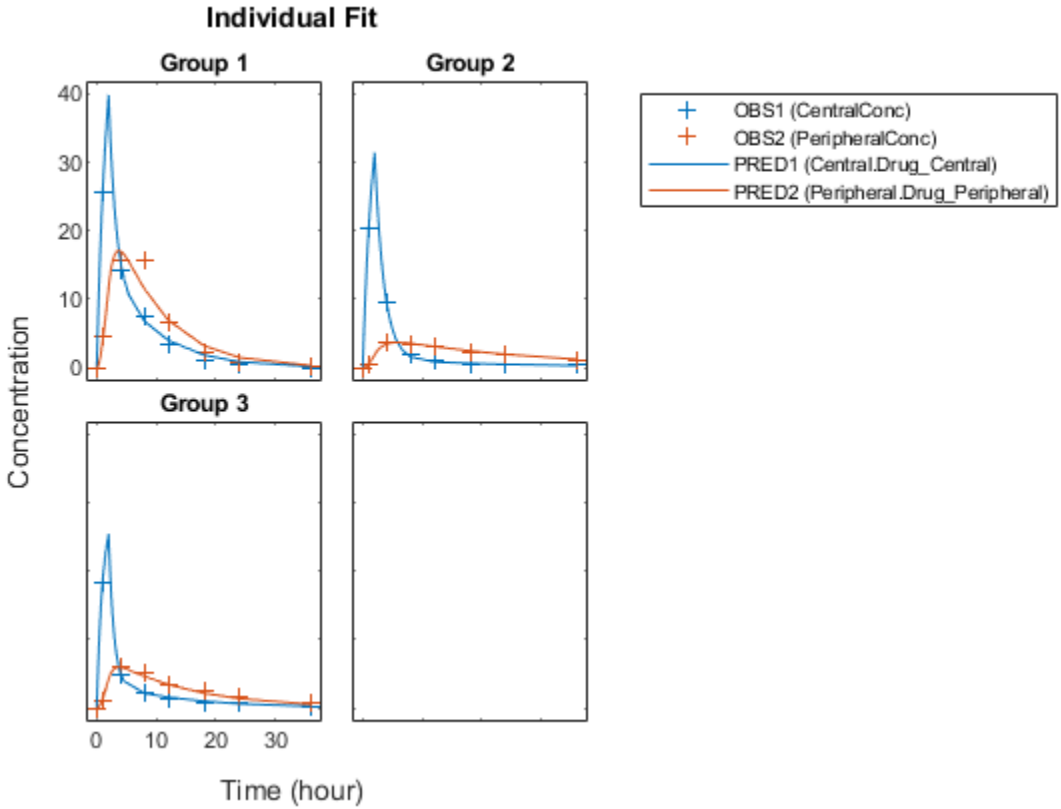
Estimate model parameters. Turn on `ProgressPlot` to see the live feedback on the status of fitting. The first row of plots are the quality measure plots for the global solver. The second row plots are for the hybrid function. For details, see “Progress Plot” on page 4-61.

```
unpooledFit =  sbiofit(model,gData,responseMap,estimatedParam,dose,globalMethod,...
                       options,'Pooled',false,'ProgressPlot',true);
```



Plot Results

```
plot(unpooledFit);
```



See Also

`sbiofit`

More About

- “Fit a One-Compartment Model to an Individual's PK Profile” on page 4-70
- “Fit a Two-Compartment Model to PK Profiles of Multiple Individuals” on page 4-96
- “Nonlinear Regression” on page 4-50

Fit a Two-Compartment Model to PK Profiles of Multiple Individuals

This example shows how to estimate pharmacokinetic parameters of multiple individuals using a two-compartment model.

Suppose you have drug plasma concentration data from three individuals that you want to use to estimate corresponding pharmacokinetic parameters, namely the volume of central and peripheral compartment (`Central`, `Peripheral`), the clearance rate (`Cl_Central`), and intercompartmental clearance (`Q12`). Assume the drug concentration versus the time

profile follows the biexponential decline $C_t = Ae^{-at} + Be^{-bt}$, where C_t is the drug concentration at time t, a and b are slopes for corresponding exponential declines.

The synthetic data set contains drug plasma concentration data measured in both central and peripheral compartments. The data was generated using a two-compartment model with an infusion dose and first-order elimination. These parameters were used for each individual.

	`Central`	`Peripheral`	`Q12`	`Cl_Central`
Individual 1	1.90	0.68	0.24	0.57
Individual 2	2.10	6.05	0.36	0.95
Individual 3	1.70	4.21	0.46	0.95

The data is stored as a table with variables `ID`, `Time`, `CentralConc`, and `PeripheralConc`. It represents the time course of plasma concentrations measured at eight different time points for both central and peripheral compartments after an infusion dose.

```
clear all
load(fullfile(matlabroot,'examples','simbio','data10_32R.mat'))
```

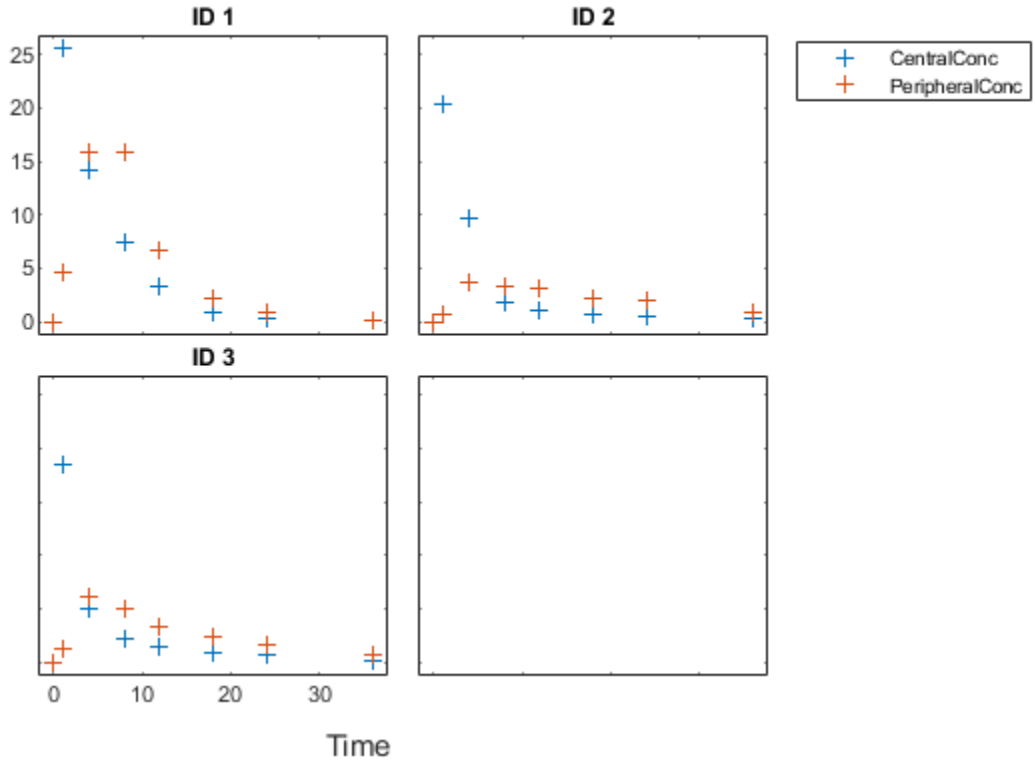
Convert the data set to a `groupedData` object which is the required data format for the fitting function `sbiofit` for later use. A `groupedData` object also lets you set independent variable and group variable names (if they exist). Set the units of the `ID`, `Time`, `CentralConc`, and `PeripheralConc` variables. The units are optional and only required for the `UnitConversion` feature, which automatically converts matching physical quantities to one consistent unit system.

```
gData = groupedData(data);
gData.Properties.VariableUnits = {'','hour','milligram/liter','milligram/liter'};
gData.Properties
```

```
ans =

  struct with fields:

               Description: ''
                  UserData: []
             DimensionNames: {'Row'  'Variables'}
              VariableNames: {'ID'  'Time'  'CentralConc'  'PeripheralConc'}
      VariableDescriptions: {}
             VariableUnits: {1x4 cell}
        VariableContinuity: []
                  RowNames: {}
         GroupVariableName: 'ID'
    IndependentVariableName: 'Time'
```

Create a trellis plot that shows the PK profiles of three individuals.

```
sbiotrellis(gData,'ID','Time',{'CentralConc','PeripheralConc'},...
            'Marker','+','LineStyle','none');
```



Use the built-in PK library to construct a two-compartment model with infusion dosing and first-order elimination where the elimination rate depends on the clearance and volume of central compartment. Use the configset object to turn on unit conversion.

```
pkmd                 = PKModelDesign;
pkc1                 = addCompartment(pkmd,'Central');
pkc1.DosingType      = 'Infusion';
pkc1.EliminationType = 'linear-clearance';
pkc1.HasResponseVariable = true;
pkc2                 = addCompartment(pkmd,'Peripheral');
model                = construct(pkmd);
configset            = getconfigset(model);
configset.CompileOptions.UnitConversion = true;
```

Assume every individual receives an infusion dose at time = 0, with a total infusion amount of 100 mg at a rate of 50 mg/hour. For details on setting up different dosing strategies, see “Doses” on page 2-42.

```
dose             = sbiodose('dose','TargetName','Drug_Central');
dose.StartTime   = 0;
dose.Amount      = 100;
dose.Rate        = 50;
dose.AmountUnits = 'milligram';
dose.TimeUnits   = 'hour';
dose.RateUnits   = 'milligram/hour';
```

The data contains measured plasma concentrations in the central and peripheral compartments. Map these variables to the appropriate model species, which are `Drug_Central` and `Drug_Peripheral`.

```
responseMap = {'Drug_Central = CentralConc','Drug_Peripheral = PeripheralConc'};
```

The parameters to estimate in this model are the volumes of central and peripheral compartments (`Central` and `Peripheral`), intercompartmental clearance `Q12`, and clearance rate `Cl_Central`. In this case, specify log-transform for `Central` and `Peripheral` since they are constrained to be positive. The `estimatedInfo` object lets you specify parameter transforms, initial values, and parameter bounds (optional).

```
paramsToEstimate   = {'log(Central)','log(Peripheral)','Q12','Cl_Central'};
estimatedParam     = estimatedInfo(paramsToEstimate,'InitialValue',[1 1 1 1]);
```

Fit the model to all of the data pooled together, that is, estimate one set of parameters for all individuals. The default estimation method that `sbiofit` uses will change depending on which toolboxes are available. To see which estimation function `sbiofit` used for the fitting, check the `EstimationFunction` property of the corresponding results object.

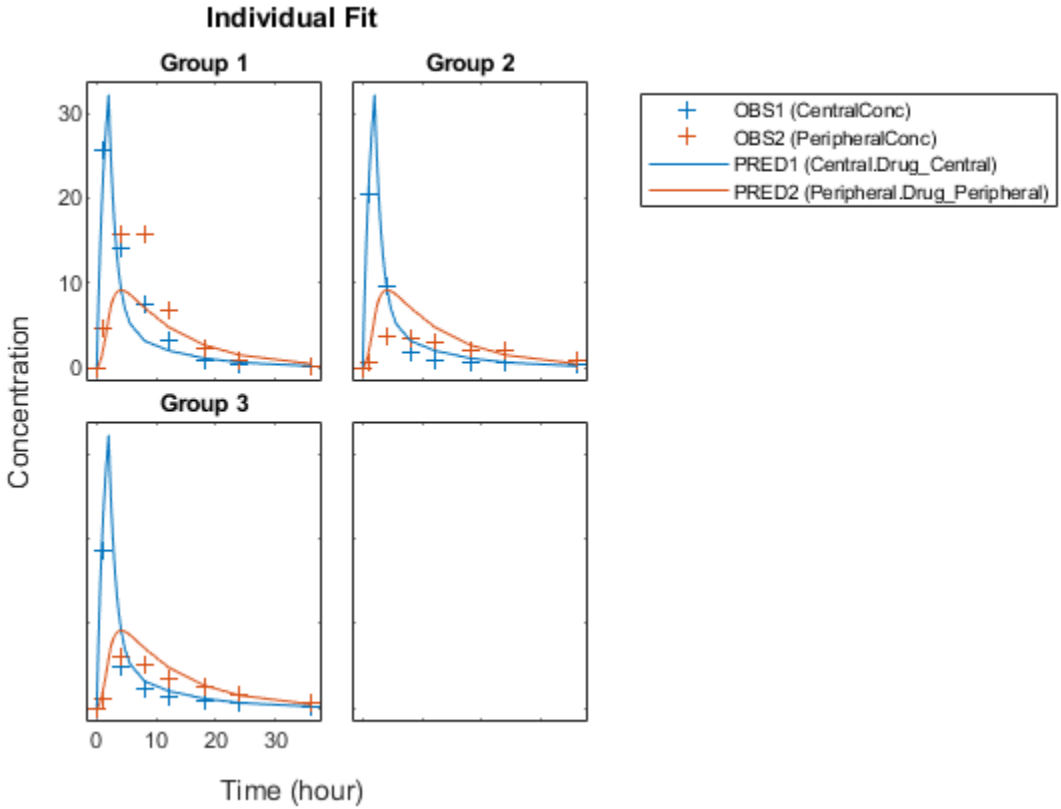
```
pooledFit = sbiofit(model,gData,responseMap,estimatedParam,dose,'Pooled',true)
```

```
pooledFit =

  OptimResults with properties:

                  ExitFlag: 3
                    Output: [1x1 struct]
                 GroupName: []
                      Beta: [4x3 table]
        ParameterEstimates: [4x3 table]
```

```
                        J: [24x4x2 double]
                     COVB: [4x4 double]
          CovarianceMatrix: [4x4 double]
                        R: [24x2 double]
                      MSE: 6.6220
                      SSE: 291.3688
                  Weights: []
             LogLikelihood: -111.3904
                      AIC: 230.7808
                      BIC: 238.2656
                      DFE: 44
            DependentFiles: {1x3 cell}
    EstimatedParameterNames: {'Central'  'Peripheral'  'Q12'  'Cl_Central'}
             ErrorModelInfo: [1x3 table]
         EstimationFunction: 'lsqnonlin'
```

Plot the fitted results versus the original data. Although three separate plots were generated, the data was fitted using the same set of parameters (that is, all three individuals had the same fitted line).

```
plot(pooledFit);
```

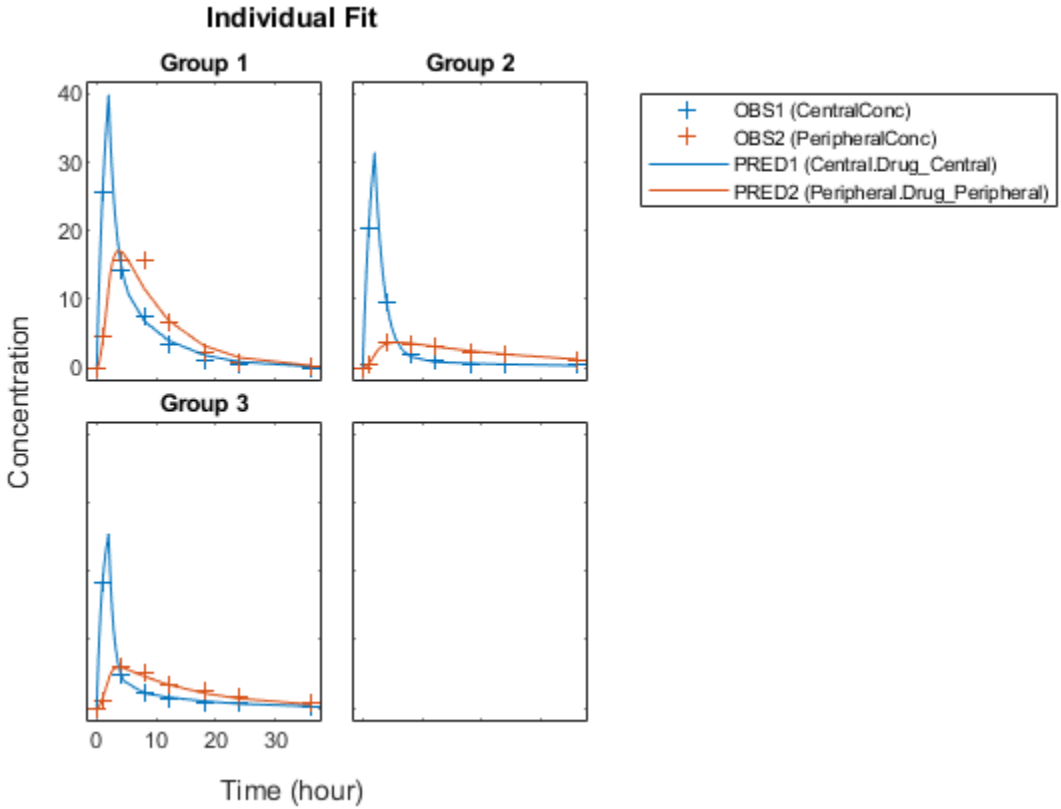


Estimate one set of parameters for each individual and see if there is any improvement in the parameter estimates. In this example, since there are three individuals, three sets of parameters are estimated.

```
unpooledFit =  sbiofit(model,gData,responseMap,estimatedParam,dose,'Pooled',false);
```

Plot the fitted results versus the original data. Each individual was fitted differently (that is, each fitted line is unique to each individual) and each line appeared to fit well to individual data.

```
plot(unpooledFit);
```



Display the fitted results of the first individual. The MSE was lower than that of the pooled fit. This is also true for the other two individuals.

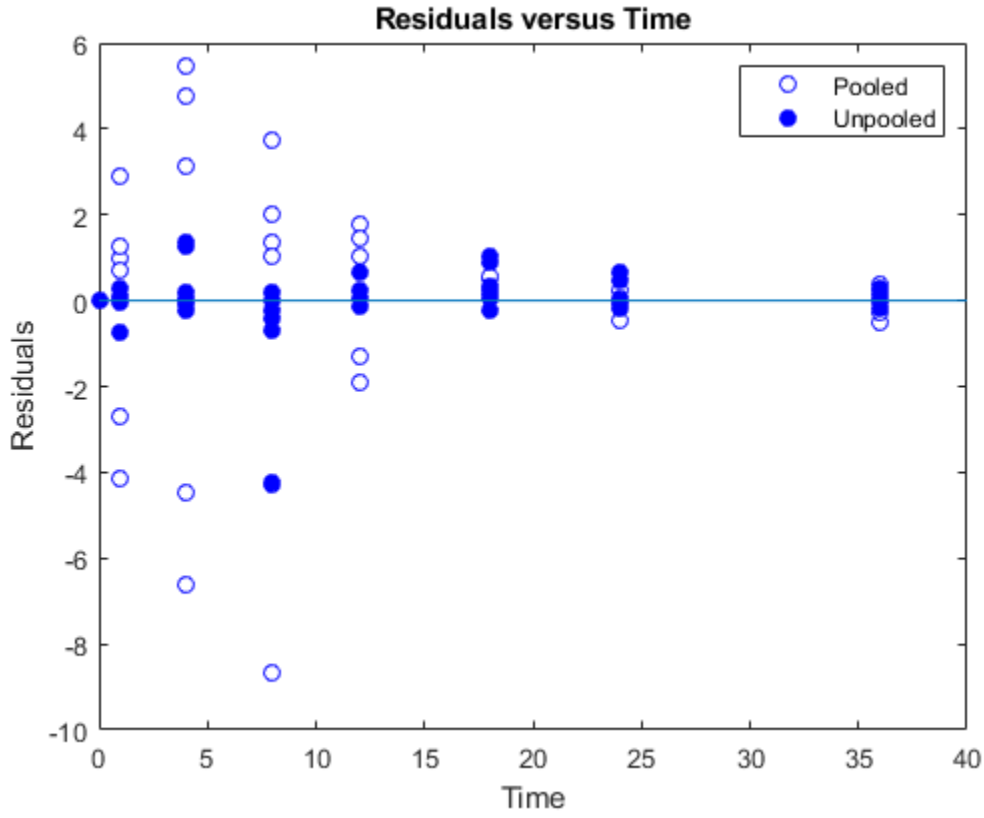
```
unpooledFit(1)
```

```
ans =

  OptimResults with properties:

                ExitFlag: 3
                  Output: [1x1 struct]
               GroupName: 1
                    Beta: [4x3 table]
```

```
          ParameterEstimates: [4x3 table]
                           J: [8x4x2 double]
                        COVB: [4x4 double]
            CovarianceMatrix: [4x4 double]
                           R: [8x2 double]
                         MSE: 2.1380
                         SSE: 25.6559
                     Weights: []
                LogLikelihood: -26.4805
                         AIC: 60.9610
                         BIC: 64.0514
                         DFE: 12
               DependentFiles: {1x3 cell}
    EstimatedParameterNames: {'Central'  'Peripheral'  'Q12'  'Cl_Central'}
               ErrorModelInfo: [1x3 table]
           EstimationFunction: 'lsqnonlin'
```

Generate a plot of the residuals over time to compare the pooled and unpooled fit results. The figure indicates unpooled fit residuals are smaller than those of pooled fit as expected. In addition to comparing residuals, other rigorous crieteria can be used to compare the fitted results.

```
t = [gData.Time;gData.Time];
res_pooled = vertcat(pooledFit.R);
res_pooled = res_pooled(:);
res_unpooled = vertcat(unpooledFit.R);
res_unpooled = res_unpooled(:);
plot(t,res_pooled,'o','MarkerFaceColor','w','markerEdgeColor','b')
hold on
plot(t,res_unpooled,'o','MarkerFaceColor','b','markerEdgeColor','b')
refl = refline(0,0); % A reference line representing a zero residual
title('Residuals versus Time');
xlabel('Time');
ylabel('Residuals');
legend({'Pooled','Unpooled'});
```



This example showed how to perform pooled and unpooled estimations using `sbiofit`. As illustrated, the unpooled fit accounts for variations due to the specific subjects in the study, and, in this case, the model fits better to the data. However, the pooled fit returns population-wide parameters. If you want to estimate population-wide parameters while considering individual variations, use `sbiofitmixed`.

See Also

`sbiofit`

More About

- “Fit a One-Compartment Model to an Individual's PK Profile” on page 4-70
- “Estimate Category-Specific PK Parameters for Multiple Individuals” on page 4-78
- “Nonlinear Regression” on page 4-50

Accelerating Model Simulations and Analyses

In this section...
“What Is Acceleration?” on page 4-106
“When to Accelerate” on page 4-106
“Prerequisites for Accelerating Simulations and Analyses” on page 4-106
“Accelerate Simulations Programmatically” on page 4-107
“Accelerate Simulations using SimBiology Desktop” on page 4-109
“Troubleshooting Accelerated Simulations” on page 4-109

What Is Acceleration?

Normally, when simulating or analyzing a model in SimBiology, the model is expressed in MATLAB code. You can accelerate the simulation by converting the model to compiled C code, which executes faster. Because this compilation step has a small time overhead, acceleration is not recommended for individual simulations of small models. However, for large models, or for repeated simulations during analysis, acceleration can provide a significant speed increase that outweighs the small time overhead.

When to Accelerate

The functionality to accelerate simulations performs optimally under the following conditions:

- Running repeated simulations with different initial conditions
- Running very long simulations (for example, simulations that take longer than a minute to run)

Prerequisites for Accelerating Simulations and Analyses

To prepare your models for accelerated simulations, install and set up a compiler:

1. Install a C compiler (if one is not already installed on your system). For a current list of supported compilers, see Supported and Compatible Compilers.
2. Ensure that any user-defined functions in your model can be used for code generation from MATLAB, so they can convert to compiled C. For more information, see

Language, Function, and Object support for C and C++ code generation (MATLAB Coder) or contact MathWorks Technical Support.

Note

- On 32-bit Windows platforms, the LCC compiler is automatically installed. However, for better performance of the acceleration functionality, you may want to install a supported compiler other than LCC, and it will be selected automatically.
- On 64-bit Windows platforms, if you have not installed another compiler, SimBiology uses the LCC64 compiler for model accelerations. If you have installed another supported compiler, it will be selected automatically.
- Microsoft® Visual Studio® 2010 run-time libraries must be available on any computer running accelerated models generated using Microsoft Windows SDK.
- If you plan to redistribute your accelerated models to other MATLAB users, be sure they have the same run-time libraries.

Accelerate Simulations Programmatically

Use `sbioaccelerate` if you are accelerating a SimBiology model. For a `SimFunction object` and an exported model (`SimBiology.export.Model`), use the corresponding `accelerate` method.

Using sbioaccelerate

Follow the two-step process for acceleration.

1. Run `sbioaccelerate` to prepare your model for accelerated simulations. Use the same input arguments that you plan to use with `sbiosimulate` in the next step. For example:

   ```
   sbioaccelerate(model,configset,doses);
   ```

   For a very large model, this step may take a minute or longer to complete.

2. Run `sbiosimulate` with the same input arguments that you used with `sbioaccelerate`. For example:

   ```
   simdata = sbiosimulate(model,configset,doses);
   ```

If you pass in an array of doses to `sbioaccelerate`, you can simulate the model using any subset of these doses and do not need to run acceleration again.

For illustrated examples, see the following.

- “Prepare a Model for Accelerated Simulation”
- “Accelerate Simulation With Array of Doses”

Using accelerate

A `SimFunction` object is automatically accelerated at the first function execution. Hence it is not necessary to accelerate the model before you create the object. However, manually accelerate using the `accelerate` method of the object if you want it accelerated in your deployment applications.

For exported model, see `accelerate`.

When to Rerun Acceleration

If you make any modifications to the model, such as changes to reactions or adding events, you need to rerun the acceleration, before running simulations.

However, there are exceptions. You do *not* need to accelerate again if you are making the changes to:

- Any variants
- InitialAmount property of species
- Capacity property of compartments
- Value property of parameters
- StopTime property of configset
- OutputTimes property of SolverOptions
- Active, Amount, and Rate properties of `ScheduleDose` and `RepeatDose`
- Time property of `ScheduleDose`
- Interval, RepeatCount, and StartTime properties of `RepeatDose`
- Notes, Tag, and UserData properties of any applicable objects

Accelerate Simulations using SimBiology Desktop

Each built-in task in the SimBiology desktop lets you specify a model to analyze, and you can choose whether to accelerate the model. For details, see “Configuring Model-Related Settings” on page 1-70.

Troubleshooting Accelerated Simulations

If you have user-defined functions, do not use persistent variables in these functions. Persistent variables are not compatible with the functionality used for accelerating simulations.

If you specify user-defined functions in SimBiology expressions, you might see the following warning if your code is not compatible with code generation from MATLAB:

```
The SimBiology Expression and any user-defined functions
could not be accelerated. Please check that these expressions
and any user-defined functions are supported for code generation
as described in the Code Generation from MATLAB documentation.
```

where *Expression* is any of the following:

- Reaction rate/rule expression
- Repeated assignment rule expression
- Event trigger expression
- Event function expression

For more information, see Language, Function, and Object support for C and C++ code generation (MATLAB Coder) or contact MathWorks Technical Support.

See Also

`accelerate` | `SimBiology.export.Model` | `SimFunction object` | `accelerate` | `sbioaccelerate`

More About

- “SimBiology Desktop”

- “Configuring Tasks” on page 1-70

Noncompartmental Analysis

Noncompartmental analysis (NCA) lets you compute pharmacokinetic (PK) parameters of a drug from the time course of measured drug concentrations. This approach does not require the assumption of a specific compartmental model. NCA is often used to determine the degree of exposure following administration of a drug, such as AUC, and other PK parameters, such as the clearance and the terminal half-life.

Data

SimBiology lets you calculate NCA parameters from concentration–time data. The data must contain a time column, a concentration column, and a dose column that defines dose amounts. Three types of drug administration routes are supported: IV bolus, IV infusion, and Extravascular. You can have a column for each type. For infusion doses, an infusion rate column is also needed.

If you have data containing multiple groups of observations, you can define a group column. If needed, you can use two levels of hierarchy to specify grouping. Specify the outer level of grouping using the group column, and specify the inner level (subgroups) in the ID column. Consider data that contains three groups, where each group contains four patients. The group column labels the three groups, and the ID column labels each patient.

Dosing

Single-dosing data contains a single dose amount for each individual. Multiple-dosing data has several doses at different times for each individual. There are common parameters calculated for either type of dosing data, and parameters that are specific to single or multiple dosing.

Common Parameters for Single and Multiple Dosing

SimBiology computes some common parameters for single- or multiple-dosing data. This figure represents the concentration-time profile after a single dose. For multiple dosing, the same principles apply, except that SimBiology uses the first period, that is, after the first dose and before the second.

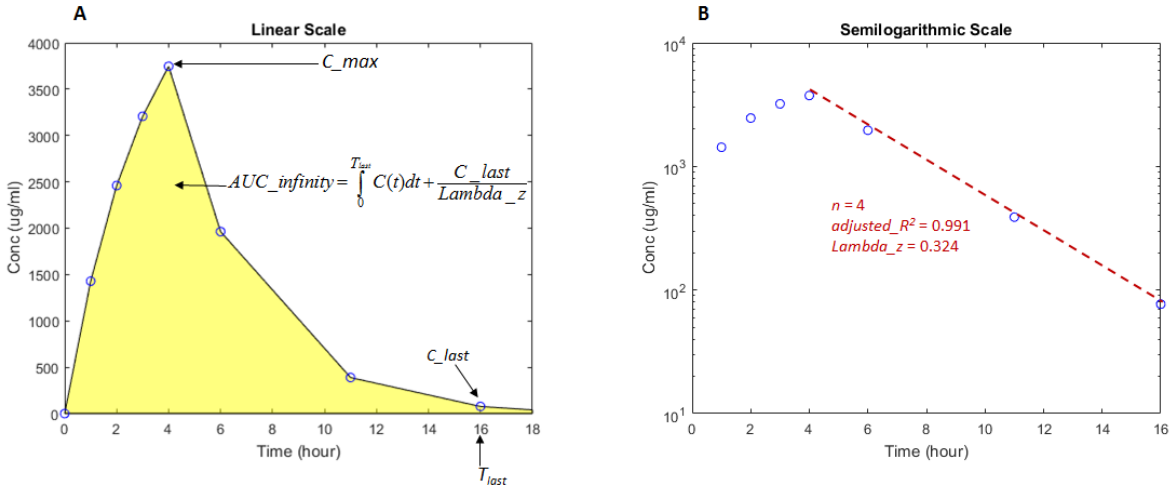


Figure **A** shows concentration–time data in a linear scale and illustrates how the AUC from time 0 to infinity is calculated. Figure **B** shows the same data in a semilogarithmic scale. To compute the terminal rate constant (*Lambda_z*), SimBiology performs a set of linear regressions of the log-transformed data using each of the last *n* points (`n = 3, 4, 5, ...`) from the terminal portion of the curve. *Lambda_z* is chosen from the regression that uses the most points and has the maximum *adjusted_R*^{2}.

This table describes the common parameters for single and multiple dosing.

Parameter	Description
AUC_0_last	Area under the measured concentration–time curve from time = 0 to the last time point. $AUC_0_last = \int_0^{Tlast} C(t)dt$, where *C(t)* is the plasma concentration at time *t*. SimBiology uses the linear trapezoidal method (MATLAB) to calculate the AUC.

Parameter	Description
AUC_infinity	Total area under the concentration–time curve extrapolating to *Inf* using the terminal rate constant *Lambda_z*. $AUC_infinity = AUC_0_last + \frac{C_last}{Lambda_z}$, where *C_last* is the last observed concentration and *Lambda_z* is the terminal rate constant.
AUC_infinity_dose	$AUC_infinity_dose = \frac{AUC_infinity}{DM}$.
AUCx_y	Partial AUC computed for a custom time range, where x and y are the start and end times, respectively.
AUC_extrap_percent	Fraction of total *AUC_infinity* obtained from extrapolation. $AUC_extrap_percent = \frac{AUC_infinity - AUC_0_last}{AUC_infinity} * 100$.
Lambda_z	To calculate the terminal rate constant (*Lambda_z*), SimBiology performs a set of linear regressions of the log(concentration)–time data using each of the last *n* points (`n = 3, 4, 5, ...`) from the terminal portion of the curve, that is, points satisfying the conditions: $(Time \geq T_{\max}) \& (Conc \leq C_{\max})$. A minimum of three points is required to determine *Lambda_z*. *Lambda_z* is chosen from the regression that uses the most points and has the maximum *adjusted_R²* among all regressions. $adjusted_R^2 = 1 - \frac{(1-R^2)*(n-1)}{n-2}$
R2	Coefficient of determination for the `linear regressions` used in the *Lambda_z* calculation.

Parameter	Description
Num_points	Number of data points from the profile used in the determination of *Lambda_z*.
CL	Total drug clearance. $Cl = \frac{DM}{AUC_infinity}$, where *DM* is the dose amount.
C_0	Extrapolated concentration at time = 0, computed using a regression of the first two data points in a profile. This parameter is for IV Bolus doses only.
C_max	Maximum observed concentration.
C_max_Dose	$C_\max_Dose = \frac{C_max}{DM}$.
C_max_x_y	Maximum observed concentration within a given time range, specified by the start time *x* and the end time *y*. This parameter is computed when you specify a custom time range in the **Cmax Time Range** box in the SimBiology desktop or set the `C_max_ranges` property of the options object created by `sbioncaoptions`.
MRT	Mean residence time. $MRT = \frac{AUMC}{AUC_infinity}$.
Tlast	Time of the last observed concentration value above the lower limit of quantization (LOQ).
T_half	Terminal half-life of the drug. $T_half = \frac{\ln(2)}{Lambda_z}$.
T_max	*T_max* is the time point at which the maximum concentration (*C_max*) is observed.

Parameter	Description
T_max_x_y	Time point at which maximum concentration is observed within a given time range, specified by the start time *x* and the end time *y*. This parameter is computed when you specify a custom time range in the **Cmax Time Range** box in the SimBiology desktop or set the `C_max_ranges` property of the options object created by `sbioncaoptions`.
V_ss	Apparent volume of distribution at equilibrium. This parameter is for IV Bolus doses only. $V_ss = \frac{DM * AUMC}{AUC_infinity^2}$.
V_z	Volume of distribution during the terminal phase. $V_z = \frac{DM}{AUC_infinity * Lambda_z}$.
DM	Administered dose amount. For multiple dosing, the last administered dose is reported.
doseSchedule	Single- or multiple-dosing data.
administrationRoute	Dose administration route. Supported routes are `IVBolus`, `IVInfusion`, `ExtraVascular`.

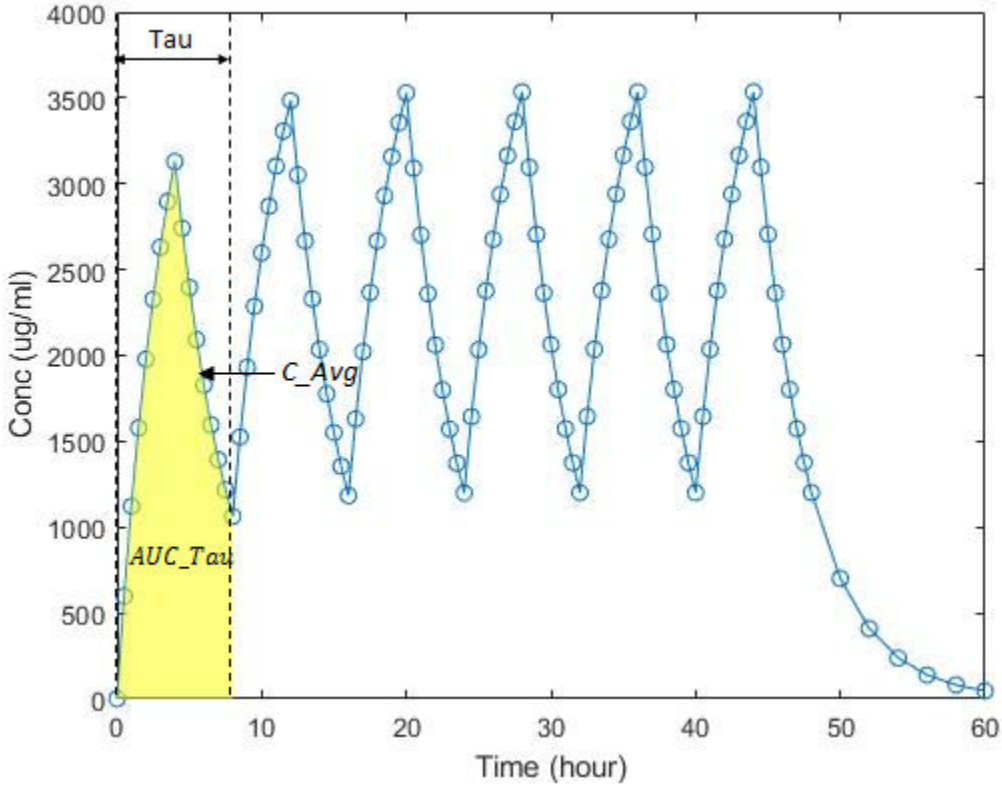
Parameters for Single Dosing

In addition to the common parameters, SimBiology reports parameters for single-dosing data.

Parameter	Description
AUMC_0_last	Area under the first moment of the concentration–time curve from time 0 to the last time point *Tlast*. $AUMC_0_last = \int_0^{Tlast} t * C(t)dt$.
AUMC	Total area under the first moment of the concentration–time curve extrapolating to *Inf* using *Lambda_z*. $AUMC = AUMC_0_last + \frac{C_last}{Lambda_z^2} + \frac{Tlast * C_last}{Lambda_z}$.
AUMC_extrap_percent	Fraction of total *AUMC* obtained from extrapolation. $AUMC_extrap_percent = \frac{AUMC - AUMC_0_last}{AUMC} * 100$.

Parameters for Multiple Dosing

This figure shows the concentration-time profile after multiple doses. SimBiology uses the first period, that is, after the first dose and before the second, to compute the following NCA parameters for multiple-dosing data, in addition to the common parameters listed previously.



Parameter	Description
AUC_Tau	Area under the concentration–time curve during a dosing period of length *Tau*. SimBiology uses the first period, that is, after the first dose and before the second. $AUC_Tau = \int_{T_first_period}^{T_first_period+Tau} C(t)dt$.
Tau	Dosing interval.

Parameter	Description
C_Avg	Average concentration over one period. $C_Avg = \frac{AUC_Tau}{Tau}$.
C_min	Minimum observed concentration during the first period, that is, `C_min = C(T_min)`.
PTF_percent	Peak trough fluctuation over one dosing interval at steady state. $PTF_Percent = \frac{C_\max - C_\min}{C_Avg} * 100$.
Accumulation_Index	Theoretical accumulation ratio. $Accumulation_Index = \frac{1}{1 - e^{-Lambda_z * Tau}}$.
T_min	Time at which the minimum concentration is reached in a dosing interval.

Sparse Sampling

To calculate PK parameters, measured concentrations at multiple time points for each individual is needed after the drug administration. Under certain circumstances, it is not feasible or not practical to obtain such longitudinal data on a single subject. In such cases, concentration data is collected from multiple individuals at each time point and then averaged to calculate NCA parameters for each group instead. SimBiology performs such sparse sampling by taking the average of the dependent variable for all individuals at the same time point. It then returns the values of NCA parameters for each group. Time values for each measurement across individuals (IDs) within a group must be identical.

Calculating NCA Parameters

You can calculate NCA parameters using the `sbionca` function in the command line or using the SimBiology desktop.

Using sbionca

`sbionca` provides command line functionality to compute NCA parameters. Define the data classification options and parameter calculation options using an option object created by `sbioncaoptions`. For an example, see “Compute NCA Parameters from Concentration-Time Data”.

Using SimBiology Desktop

After you import the data, select **Open > NCA** on the **Define Plot** tab. If your data has a grouping column, specify it using **Group Data Column**. Use **ID Data Column** to specify the inner level of grouping. For details, see “Data” on page 4-111. Specify the dosing data column (**IV Bolus Dose Data Column** or **Extravascular Dose Data Column**, or both if you have two columns for both types). **Lower limit of quantization (LOQ)** is a threshold below which the values of dependent variables are truncated to zero.

Lambda Time Range lets you specify a custom time range to compute the terminal rate constant (Lambda_z). The time range applies to all groups; you cannot specify a different time range for each group.

Partial AUC lets you specify a custom time range to compute the partial AUC bounded by the start and end times. You can specify a different time range for each group.

Cmax Time Range lets you specify a custom time range to report the maximum observed concentration within the range (*C_max*) and the time (*T_max*) when it is observed. You can specify a different time range for each group.

To export the calculated statistics, select **Export Statistics** from the context menu of the NCA table. By default, the data is exported as a table (MATLAB). To convert it to a `dataset`, use `table2dataset`.

See Also

`sbionca` | `sbioncaoptions`

More About

- “SimBiology Desktop”
- “Import Data from a NONMEM-Formatted File Using the SimBiology Desktop” on page 5-15

5

Pharmacokinetic Modeling

- “Pharmacokinetic Modeling Functionality” on page 5-2
- “Importing Data — Supported Files and Data Types” on page 5-6
- “Importing Data” on page 5-12
- “Import Data from a NONMEM-Formatted File Using the SimBiology Desktop” on page 5-15
- “Create Pharmacokinetic Models” on page 5-24
- “Perform Data Fitting with PKPD Models” on page 5-36

Pharmacokinetic Modeling Functionality

In this section...
“Overview” on page 5-2
“How SimBiology Supports Pharmacokinetic Modeling” on page 5-2
“Pharmacokinetic Modeling Examples” on page 5-4
“Acknowledgements: Tobramycin Data Set” on page 5-4

Overview

SimBiology software extends the MATLAB computing environment for analyzing pharmacokinetic (PK) data using models. The software lets you do the following:

- Create models — Use a model construction wizard. Alternatively, extend any model with pharmacodynamic (PD) model components, or build higher fidelity models. See “Model” on page 5-3 for more information.
- Fit data — Fit nonlinear, mixed-effects models to data, and estimate the fixed and random effects, or fit the data using nonlinear least squares. For more information, see “Analyze Data Using Models” on page 5-3.
- Generate diagnostic plots — For more information, see “Analyze Data Using Models” on page 5-3.

The software lets you work with different model structures, thus letting you try multiple models to see which one produces the best results.

How SimBiology Supports Pharmacokinetic Modeling

Import and Work with Data

You can import tabular data into the SimBiology desktop or the MATLAB Workspace. The supported file types are `.xls`, `.csv`, and `.txt`. You can specify that the data is in a NONMEM formatted file. The import process interprets the columns according to the NONMEM definitions. For details, see “Importing Data” on page 5-12.

From the SimBiology desktop, you can filter the raw data to suppress outliers, visualize data using common plots (such as `plot`, `semilog`, `scatter`, or `stairs`), and perform basic statistical analysis. You also can use functions to process and visualize the data at

the command line. For an example on importing NONMEM data, see “Import Data from a NONMEM-Formatted File Using the SimBiology Desktop” on page 5-15.

Model

SimBiology provides an extensible modeling environment. You can do any of the following:

- Create a PK model using a model construction wizard to specify the number of compartments, the route of administration, and the type of elimination.
- Extend any model with pharmacodynamic (PD) model components, or build higher fidelity models.
- Build or load your own SimBiology, or SBML model.

For more information, see “What is a Model?” on page 2-2.

Analyze Data Using Models

Perform both individual and population fits to grouped longitudinal data:

- Individual fit — Fit data using nonlinear least-squares method, specify parameter transformations, estimate parameters, and calculate residuals and the estimated coefficient covariance matrix. For a command line workflow, see “Fitting Workflow for sbiofit” on page 4-55. For the SimBiology desktop, see “Fit Data” on page 1-71.
- Population fit — Fit data, specify parameter transformations, and estimate the fixed effects and the random sources of variation on parameters using nonlinear mixed-effects models. For a command line workflow, see “Nonlinear Mixed-Effects Modeling Workflow” on page 4-44. For the desktop, see “Fit Data” on page 1-71.
- Population fit using a stochastic algorithm — Fit data, specify parameter transformations, and estimate the fixed effects and the random sources of variation on parameters, using the Stochastic Approximation Expectation-Maximization (SAEM) algorithm. SAEM is more robust with respect to starting values. This functionality relaxes assumption of constant error variance. Specify `nlmefitsa` as the estimation function name when you run `sbiofitmixed` or in the Fit Data task of the desktop.

In addition, you can turn on the ProgressPlot on page 4-61 option to get the live feedback on the status of parameter estimation.

Pharmacokinetic Modeling Examples

The following examples show how to estimate pharmacokinetic parameters at the command line.

- “Modeling the Population Pharmacokinetics of Phenobarbital in Neonates”
- “Fit a One-Compartment Model to an Individual's PK Profile” on page 4-70
- “Fit a Two-Compartment Model to PK Profiles of Multiple Individuals” on page 4-96
- “Estimate Category-Specific PK Parameters for Multiple Individuals” on page 4-78
- “Perform Hybrid Optimization Using sbiofit” on page 4-91

For a desktop example, see “Estimate Pharmacokinetic Parameters Using SimBiology Desktop”.

Acknowledgements: Tobramycin Data Set

Acknowledgements for data in the `tobramycin.txt` file are located in the `/matlab/toolbox/simbio/simbiodemos` folder. Data set is provided by Dr. Leon Aarons (`laarons@fs1.pa.man.ac.uk`).

The data in the `tobramycin.txt` file were downloaded from the Web site of the Resource Facility for Population Kinetics `http://depts.washington.edu/rfpk/service/datasets/index.html` (no longer active). Funding source: NIH/NIBIB grant P41-EB01975.

The original data set was modified as follows:

- Header comments were removed.
- The file was converted to a tab-delimited format.
- Missing values in the `HT` column were denoted with `"."` instead of `100000000.000`.

References

[1] Original Publication: Aarons L, Vozeh S, Wenk M, Weiss P, and Follath F. “Population pharmacokinetics of tobramycin.” *Br J Clin Pharmacol.* 1989 Sep;28(3):305–14.

See Also

`sbiofit` | `sbiofitmixed`

More About

- “Nonlinear Regression”
- “Nonlinear Mixed-Effects Modeling”

Importing Data — Supported Files and Data Types

In this section...
“Supported Files and Data Types” on page 5-6
“Support for Importing NONMEM Formatted Files” on page 5-6
“Creating a Data File with SimBiology Definitions” on page 5-11

Supported Files and Data Types

You can import tabular data to the SimBiology desktop or to the MATLAB Workspace. The supported file types are Excel files (.xls, .xlsx), text files (.csv, .txt), and SAS XPORT files (.xpt). You can also specify that the data is in a NONMEM formatted file. The import process interprets the columns according to the NONMEM definitions. For more information see “Support for Importing NONMEM Formatted Files” on page 5-6.

From the SimBiology desktop, you can filter the raw data to suppress outliers, visualize data using common plots (such as `plot`, `semilog`, `scatter`, or `stairs`), and perform basic statistical analysis. You also can use functions to process and visualize the data at the command line.

Note If your data set contains dosing information that is infusion data, the data set must contain the rate and not an infusion duration.

Unit Conversion

Regardless of whether unit conversion functionality is on or off, dosing in the data file must be expressed in amounts (or as `amount/time` for infusion rate). By default **Unit Conversion** is off, so you must ensure that units for the data are consistent with each other. If you want to turn on unit conversion, see “Unit Conversion for Imported Data” on page 5-34 .

Support for Importing NONMEM Formatted Files

You can specify that the data is in a NONMEM formatted file. The following table highlights the interpretation of this data in SimBiology software.

<table>
<tr><th>Column Header</th><th>Interpretation</th></tr>
<tr><td>ID</td><td>Text (character vector), numeric, or categorical values that identify the record or group. The import process assumes that contiguous data with the same value contains data from one individual. If the data contains non-contiguous references to the same value, the import process assigns the second ID encountered an indexed valued derived from the group first encountered. For example, if the ID columns contains <code>[1 1 1 2 2 2 1 1 1]</code>, the IDs assigned are <code>1, 2, 1_1</code>.</td></tr>
<tr><td>TIME</td><td>Monotonically increasing positive values within each group, indicating time of observation or dose or text (character vector). The data file can specify clock (<code>2:30</code> as a character vector) or decimal values (<code>6.25</code>). The import process assigns a value of <code>0</code> to the first <code>TIME</code> value in the data file. The import process assigns subsequent values relative to the first value.

The following table is an example of how the import process interprets the clock values as decimal values.

<table><tr><th>Original Clock Values</th><th>Imported Values</th></tr><tr><td>10:00</td><td>0</td></tr><tr><td>10:30</td><td>0.5</td></tr><tr><td>11</td><td>1</td></tr><tr><td>12:30</td><td>2.5</td></tr></table>
If the data file also contains a <code>DATE</code> column, the import process uses it with the <code>TIME</code> column in calculating the relative <code>TIME</code> values. The column cannot contain <code>Inf</code>.</td></tr>
</table>

Column Header	Interpretation
`DATE`, `DAT1`, `DAT2`, or `DAT3`	Defines the day of the observation or the dose. The column can contain the month as a number (`9`) or a character vector (`Sep`). Specify date in the following formats: • `DATE` — The column can specify `month/day/year` or `month-day-year`. If you specify two numbers, the import process assumes they are month and day. You can use either `/` or `-` as a separator. • `DAT1` — The column can specify `day/month/year` or `day-month-year`. If you specify two numbers, the import process assumes they are day and month. • `DAT2` — The column can specify `year/month/day` or `year-month-day`. If you specify two numbers, the import process assumes they are month and day. • `DAT3` — The column can specify `year/day/month` or `year-day-month`. If you specify two numbers, the import process assumes they are day and month. **Note** • If you specify only one number, the import process assumes it is the day. • You can omit the year or specify 1, 2, 3, or 4 digits. If you specify two-digit years, it is assumed to be in the 1900s. • If the data has the `DAT1`, `DAT2`, or `DAT3` column, set the `DateLabel` property of a `NMFileDef` object accordingly using `sbionmfiledef`. Then specify the object as the second input argument when you run `sbionmimport`.
`DV`	Numeric value of an observation. Column cannot contain `Inf` or `–Inf`.

Column Header	Interpretation
`MDV`	Defines whether a row describes an observation: • Row contains 0 — Observation event • Row contains 1 — Not an observation event
`EVID`	Defines the type of event described for the row in the record: • 0 — Observation event; row contains an observed value. • 1 — Dose event; row describes a dose. • 2 — Other event; row describes some other event such as measurement of a covariate. If a column contains values for dose, but `EVID` is not `1`, the import process ignores the value. You see a warning and the value is ignored. If `EVID` is set to 2, then only those specified row data are imported as covariate data. However, if you have an `EVID` column as well as one or more covariate columns, but do not specify a value of `2` anywhere in the `EVID` column, then SimBiology imports all the row data as covariate values. The import process does not support values `3` and `4`. You see a warning and the value is ignored.

Column Header	Interpretation
CMT	Indicates which compartment is used for observation value or for dose received. The interpretation also depends on EVID: • Observation event (EVID = 0) — CMT column indicates which compartment was used for observation value. • Dose Event (EVID = 1) — CMT column indicates which compartment received the dose. **Note** SimBiology numbers compartments starting with 1, while NONMEM numbers them starting with 0. For instance, if a NONMEM data file contains doses and measurements for CMT = 0, SimBiology generates data columns named Dose1 and Response1 respectively.
AMT	Positive number indicating dose. 0 or NaN specifies no dose administered. The column cannot contain Inf.
RATE	Positive number indicating rate of infusion. 0 specifies an infinite rate (equivalent to a bolus dose), and NaN specifies no rate. The column cannot contain Inf.
II	Positive number defining the time between doses.
ADDL	When the data specifies a number of identical serial doses at specific intervals (defined by II), ADDL specifies the number of doses in the series excluding the initial dose. If the data specifies II but not ADDL, then SimBiology assumes that the dosing occurs for the duration of that data record.

Unsupported NONMEM Definitions

The import process does not support (and therefore ignores) the rows containing the following values or definitions:

- EVID values 3 and 4
- SS column for specifying steady state doses
- PCMT column to define whether to compute a prediction for the row

- `CALL` column for calling the ERROR or the PK subroutine
- If rate is specified as being less than zero, it is assumed to be zero

Creating a Data File with SimBiology Definitions

If you are creating a file containing population data that you want to later import into SimBiology, create the data file with the following columns:

- Group column — Specify text, numeric, or categorical values. The rows in the file that have the same Group column value are for the same individual.
- Time column — Specify monotonically increasing positive values within each group that define the time of the dose, observation and/or covariate measurements.
- Zero or more dosing columns — Create one dosing column for each compartment being dosed. In each column, specify positive values representing doses in amount that are added to a species. Use `0` or `NaN` to specify that no dose was applied at the specified time. This is useful for times when an observation was recorded but no dose was applied.
- Zero, or more rate columns — Specify positive values. Zero specifies an infinite rate and `NaN` specifies that no rate applies. The rate column is associated with a dosing column and defines the rate at which the dose is administered.
- Zero or more observation columns — Specify numeric values or `NaN`s. You can only specify one observation value at a particular time for each group. `NaN` values define that no observation was recorded at the specified time. This is useful for times when a dose was applied but no observation was recorded.
- Zero or more covariate columns — Specify numeric values or `NaN`s. Each value defines the covariate value at the specified time. `NaN` values define that no covariate observation was recorded at the specified time.

  If you set an `EVID` value of `2` for some rows, then SimBiology imports only those rows as covariate data. If you do not mention an `EVID` value of `2` anywhere and have one or more covariate columns, then SimBiology imports all the row data as covariate data.

Importing Data

In this section...
“Import Data from Files” on page 5-12
“Importing Data from NONMEM-Formatted Files” on page 5-13
“Other Resources for Importing Data” on page 5-14

Import Data from Files

Use the `dataset` function to import tabular data with named columns into an array that you can use in fitting and analysis at the command line. Use this function when you want to import the data without NONMEM interpretation of column headers. The `dataset` function lets you specify parameter/value pair arguments in which you can specify options such as the type of delimiter, and whether the first row contains header names. For more information, see `dataset`.

To prepare the data file for import, remove any comments that are present at the beginning of the file.

Examples:

```
% text files
data = dataset('file', 'tobramycin.txt')
% text files with . in place of missing values
data = dataset('file', 'tobramycin.txt', 'TreatAsEmpty', '.')

% For Excel files
data = dataset('xlsfile', 'tobramycin.xls')
```

You can also construct the `dataset` array from variables in the MATLAB Workspace.

```
% Create a 10x2 array
x = rand(10,2);
% Construct a dataset array containing x
data = dataset({x(:, 1), 'Column1'}, {x(:,2), 'Column2'})
```

If you import the data as separate variables containing doubles, you can construct the `dataset` array by concatenating the variables.

```
% Create 2 10x1 vectors
x = rand(10,1);
```

```
y = rand(10,1);
% Construct a dataset array containing x and y
data = dataset({x, 'Column1'}, {y, 'Column2'})
```

After you finish analyzing your data, you can export any new variables in the MATLAB Workspace to a variety of file formats.

Importing Data from NONMEM-Formatted Files

Use the `sbionmimport` function to import data from NONMEM formatted files. To import the data without NONMEM interpretation of column headers, see “Import Data from Files” on page 5-12.

To prepare the data file for import, remove any comments that are present at the beginning of the file and select one of the following methods to import your data:

- If the data file contains only the column header values shown in “Support for Importing NONMEM Formatted Files” on page 5-6, use the syntax shown in the following example:

  ```
  filename = 'C:\work\datafiles\dose.xls';
  ds = sbionmimport(filename);
  ```

- If the data file has column header labels different from the table shown in “Support for Importing NONMEM Formatted Files” on page 5-6 and you want to apply NONMEM interpretation of headers:

  1. Create a NONMEM file definition object. This object lets you define what the column headers in the data file mean in SimBiology. In the following example, the column containing response values is `CP`, whereas in NONMEM formatted files the column is labelled `DV`.

     To use the tobramycin data set [1] on page 5-4, create a NONMEM file definition object and define the following:

     ```
     def = sbionmfiledef;
     def.DoseLabel = 'DOSE';
     def.GroupLabel = 'ID';
     def.TimeLabel = 'TIME';
     def.DependentVariableLabel = 'CP';
     def.MissingDependentVariableLabel = 'MDV';
     def.EventIDLabel = 'EVID';
     def.ContinuousCovariateLabels = {'WT', 'HT', 'AGE', 'SEX', 'CLCR'};
     ```

Your file can contain any name for column headings. See `sbionmfiledef` for the list of properties you can configure in the NONMEM file definition object.

2. Use the `sbionmimport` function to import your data file with the column header definitions as specified in the NONMEM file definition object. For example, browse to *`matlabroot`*`/toolbox/simbio/simbiodemos/` (where *`matlabroot`* is the folder where MATLAB is installed).

   ```
   [data, pkDataObject] = sbionmimport('tobramycin.txt', def, ...
      'TreatAsEmpty', '.');
   ```

   This example shows you how to obtain the PKData object, `PKDataObj`, while importing, since you will use the PKData object in fitting the model later.

   The `sbionmimport` function accepts property-name-value pairs accepted by `dataset`. For example, if the data set does not contain column headers, use `'ReadVarNames', false` to specify that `sbionmimport` should read values from the first row of the file.

For information about creating a model to fit the data, see “Create a Pharmacokinetic Model Using the Command Line” on page 5-26.

Other Resources for Importing Data

For detailed information about supported data formats and the functions for importing data into the MATLAB Workspace, see the “Methods for Importing Data” (MATLAB).

You also can import data using the MATLAB Import Wizard (see “Import Images, Audio, and Video Interactively” (MATLAB). Use the Import Wizard, to import data as text files (such as `.txt` and `.dat`), MAT-files, and spreadsheet files, (such as `.xls`).

The MATLAB Import Wizard processes the data source. The wizard recognizes data delimiters, as well as row or column headers, to facilitate the process of data selection and importation into the MATLAB Workspace. You can import the data to the SimBiology desktop from the MATLAB Workspace.

Import Data from a NONMEM-Formatted File Using the SimBiology Desktop

This example shows how to import data from a NONMEM-formatted file. The data can be in any of the following supported file formats: .xlsx, .xls, .csv, and .txt. Note: Before importing any NONMEM formatted data from a file, remove any comments that are present at the beginning of the file.

Load Sample Data

Open the SimBiology desktop by typing `simbiology` in the MATLAB Command Window or clicking **SimBiology** on the **Apps** tab.

On the **Home** tab, select **Add Data > Load Data from File**.

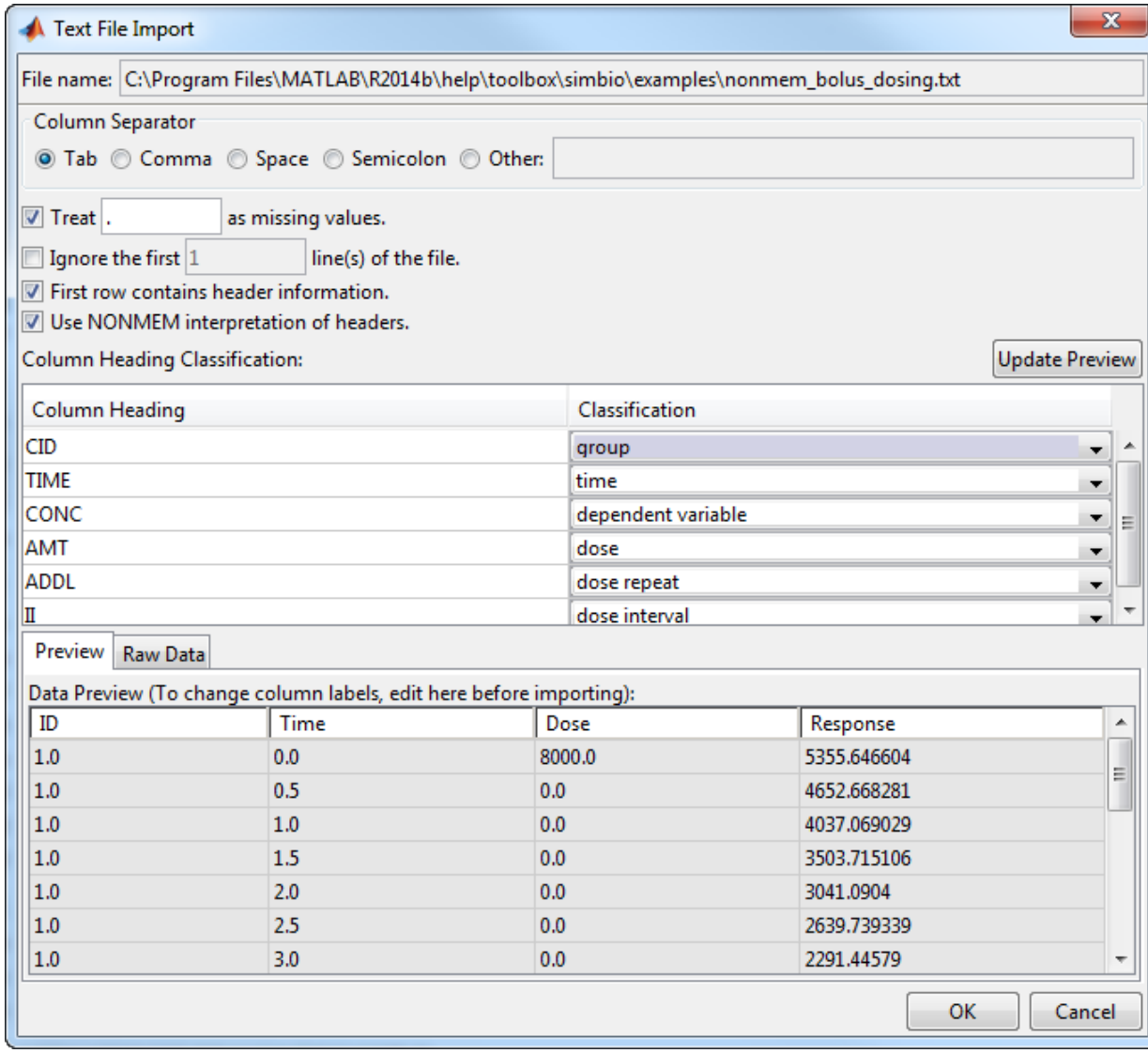
Navigate to the folder *`matlabroot`*`\help\toolbox\simbio\examples`, where *`matlabroot`* is the folder where MATLAB is installed, and open a sample NONMEM-formatted file named `nonmem_bolus_dosing.txt`. This file contains synthetically generated data for 20 individuals who received bolus doses every 8 hours for 5 times, and drug plasma concentrations were recorded every half hour for 60 hours.

Note If you are using a Macintosh platform, press **Command+Shift+G** in the File Browser dialog box, and enter the full path to the folder.

Configure NONMEM Data Settings

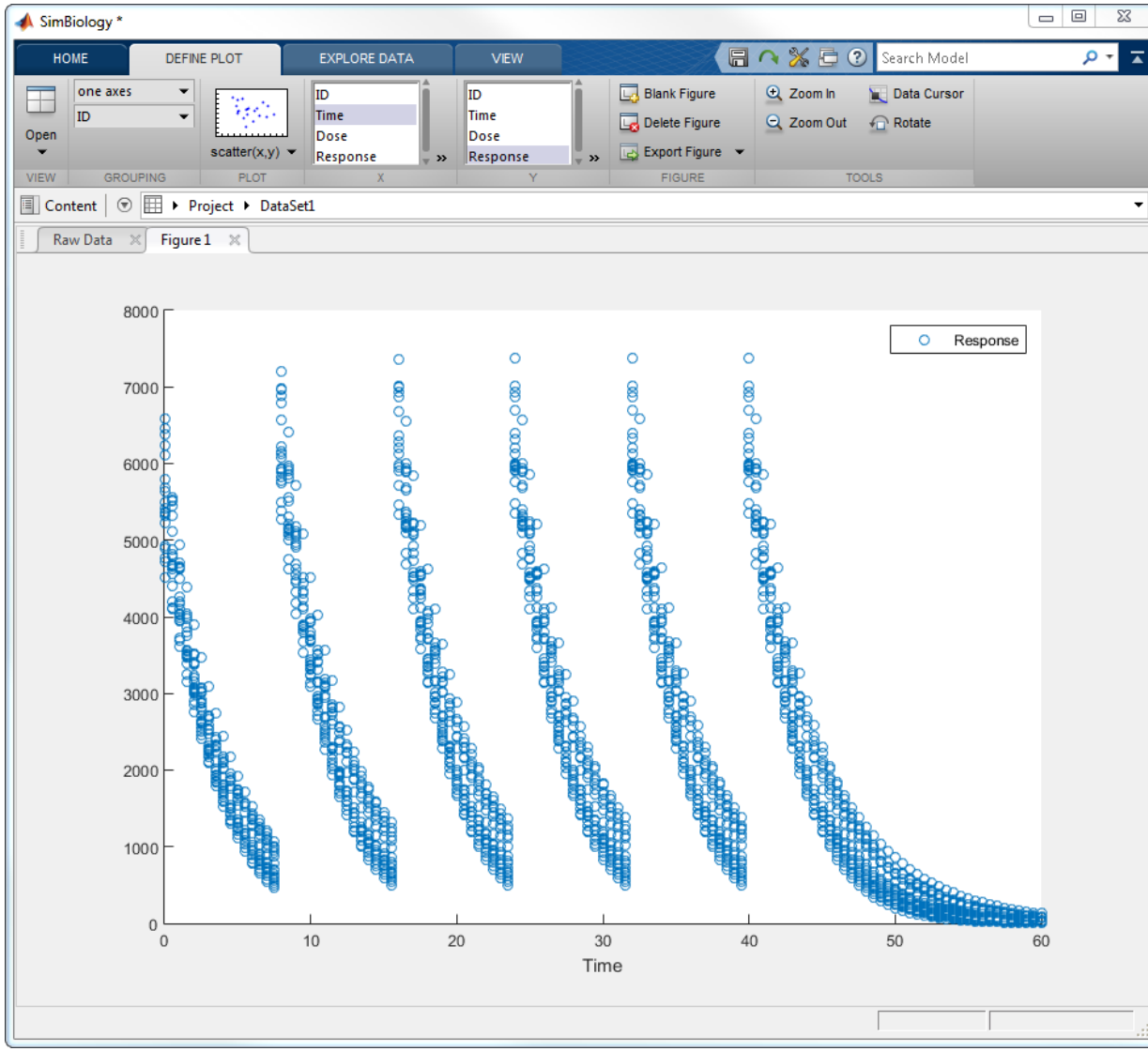
From the **Text File Import** dialog box, select **Tab** as **Column Separator**, and select **Use NONMEM interpretation of headers**.

The **CID** column heading corresponds to groups of patients. Select `group` to identify it as a group column, and click **Update Preview**.



Note The import dialog maps NONMEM column headings to appropriate data classification categories for SimBiology to interpret the data. If there is any ambiguity, a warning message is shown at the bottom of the window.

Click **OK** to load the data, and SimBiology generates a scatter plot of time versus response for all individuals as shown next.

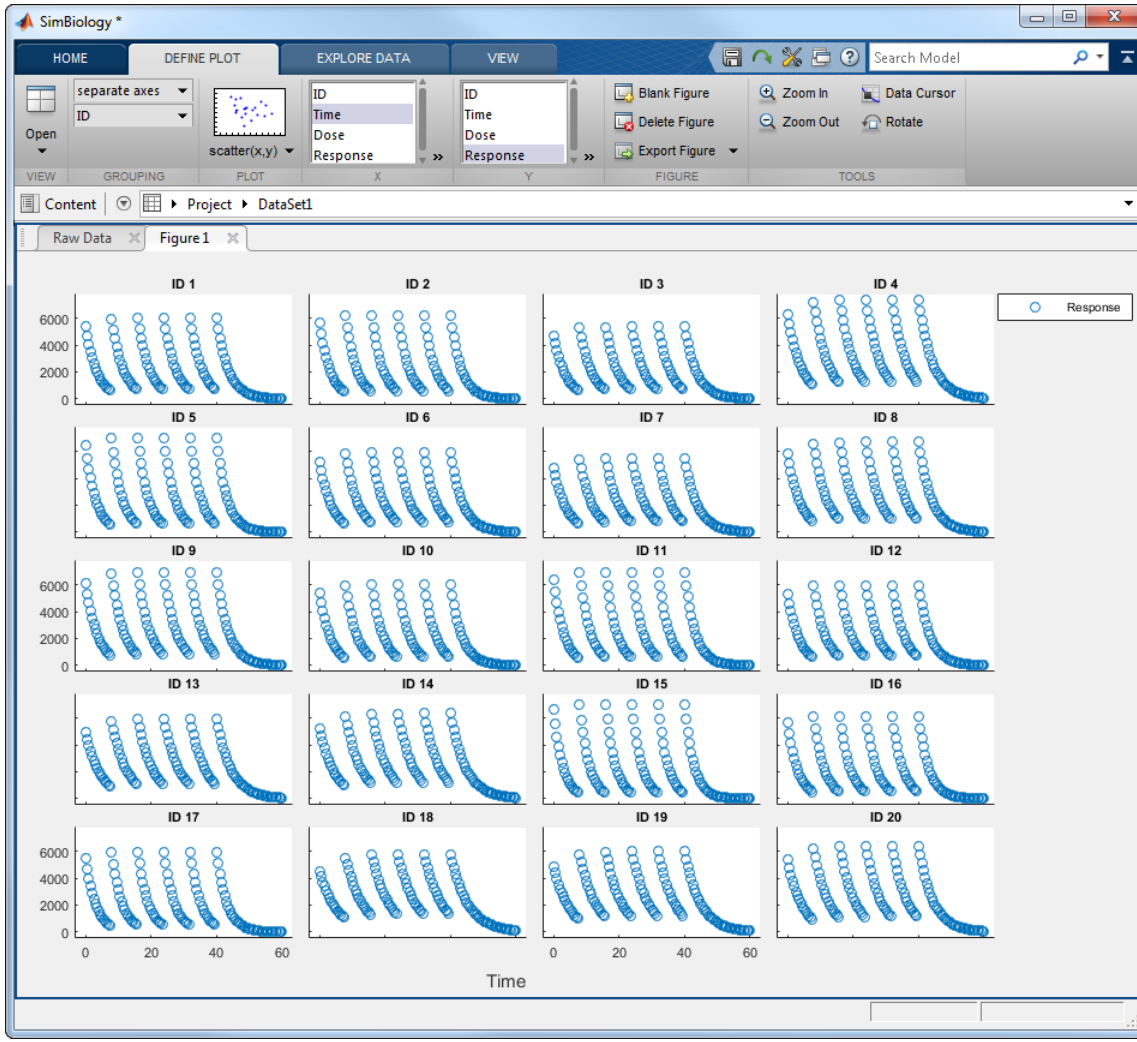


View Raw Data

Click the **Raw Data** tab at the bottom of the plot to see a tabular format of the data. Notice that SimBiology has generated an updated **Dose** column and a **Response** column, which is the CONC column of the original NONMEM data. You can assign appropriate units using the drop-down list under each column.

Define Plot Settings

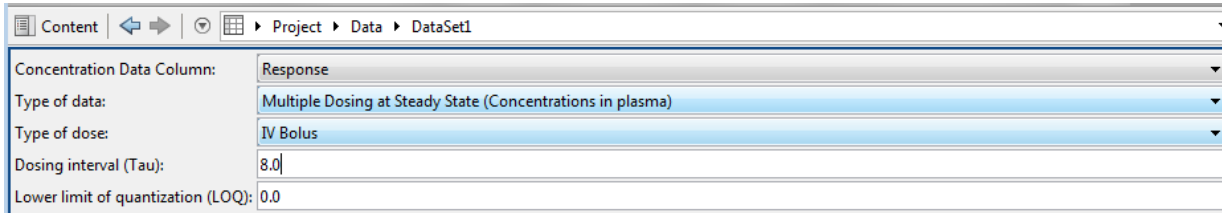
Instead of having shown in one axes for all patients, you can have separate axes for each individual. Go back to the **Figure 1** tab. On the **Define Plot** tab, in the **Grouping** section, select **separate axes** instead of **one axes**. SimBiology then generates separate axes for all individuals.



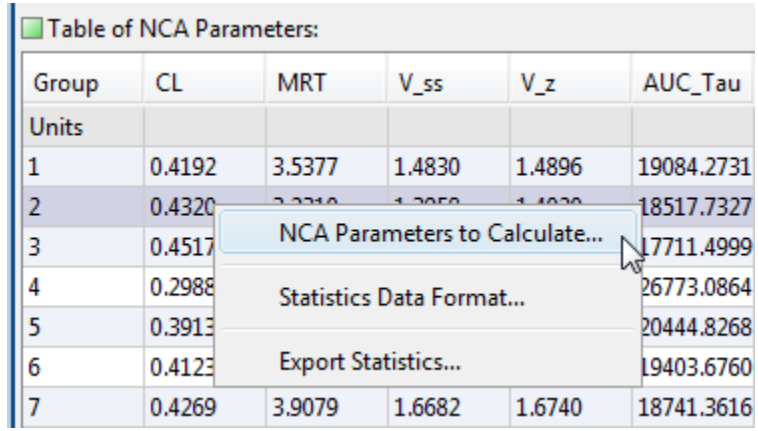
Non-compartmental Analysis (NCA) Parameters

SimBiology lets you calculate NCA parameters from concentration-time data, including AUC, clearance, and terminal half life. For details, see NCA on page 4-111.

To view NCA parameters, select **Open > NCA**. Since the imported data contains multiple doses, change the **Type of data** to **Multiple Dosing at Steady State (Concentrations in plasma)**, and enter `8.0` for **Dosing interval (Tau)**.



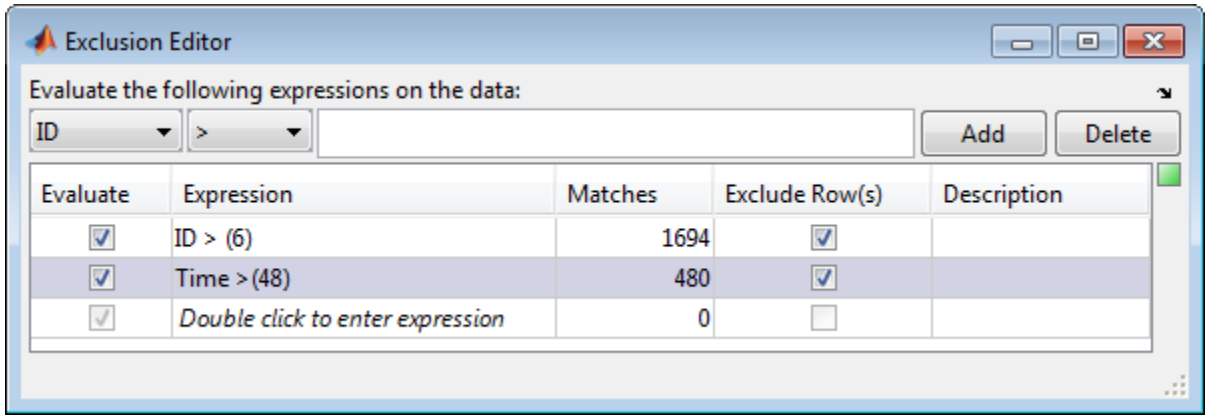
The **Table of NCA Parameters** is then updated. In order to see the description of each parameter and select which parameter to include in the calculation, right-click anywhere on the table, and select **NCA Parameters to Calculate**.



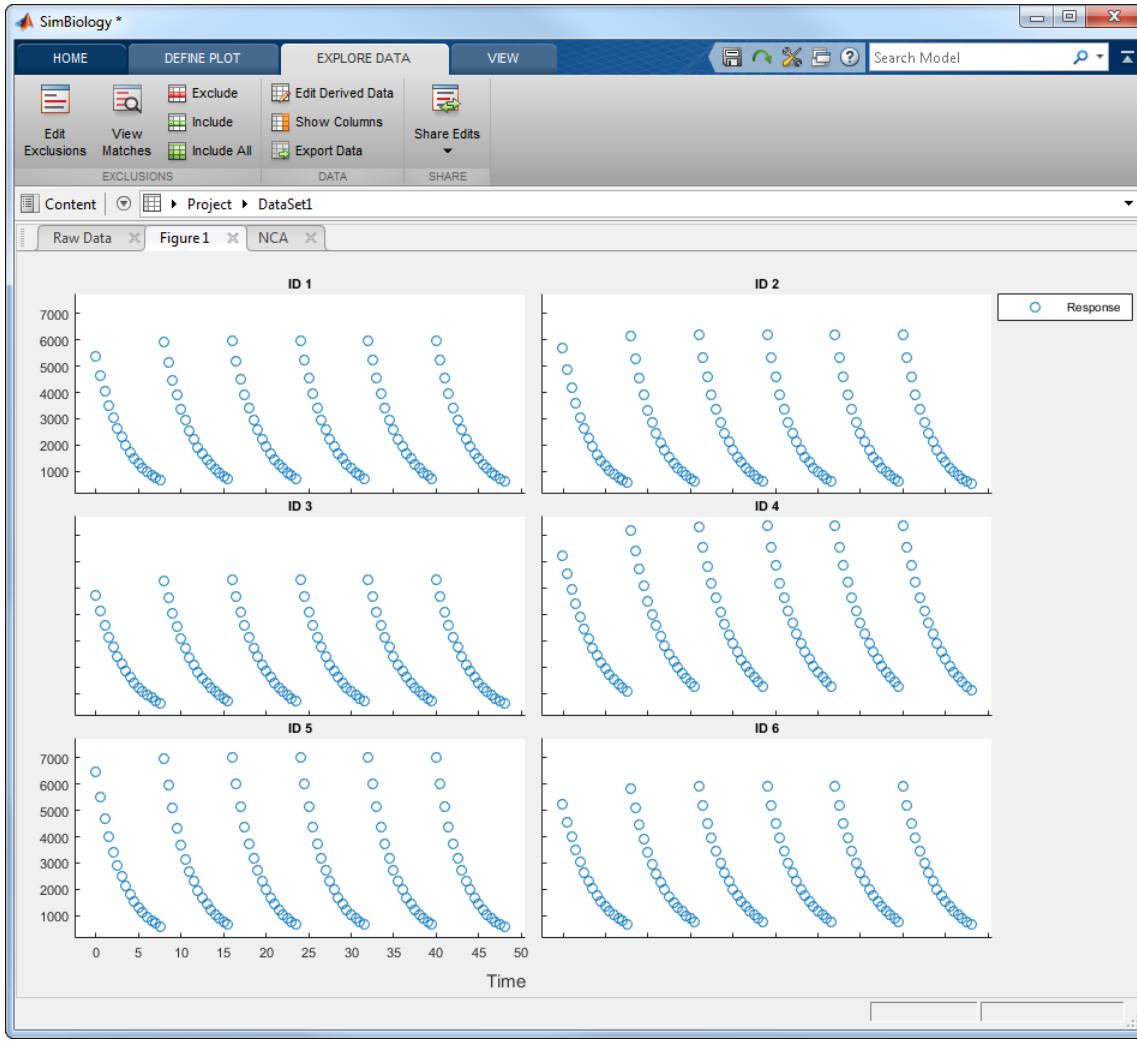
You can also export the parameters as a dataset or separate variables to the MATLAB Workspace using the **Export Statistics** option.

Filter Data

You can filter the data and select which data to plot by defining data exclusion rules. On the **Explore Data** tab, click **Edit Exclusions**. To exclusively view the response data of the first six individuals for day 1 and day 2, add two exclusions: `ID > (6)`, and `Time > (48)`. Select **Exclude Row(s)** for each expression as shown next.



Go back to the **Figure 1** tab, and it now shows only the first six individual data for the first two days.


SimBiology *
HOME
DEFINE PLOT
EXPLORE DATA
VIEW
Search Model
Edit Exclusions
View Matches
Exclude
Include
Include All
EXCLUSIONS
Edit Derived Data
Show Columns
Export Data
DATA
Share Edits
SHARE
Content
Project
DataSet1
Raw Data
Figure 1
NCA
ID 1
ID 2
ID 3
ID 4
ID 5
ID 6
Response
7000
6000
5000
4000
3000
2000
1000
0
5
10
15
20
25
30
35
40
45
50
Time

If necessary, you can add additional data exclusion expressions and share them via **Share Edits** icon on the **Explore Data** tab.

See Also

More About

- “Support for Importing NONMEM Formatted Files” on page 5-6
- “Noncompartmental Analysis” on page 4-111

Create Pharmacokinetic Models

In this section...
“Ways to Create or Import Pharmacokinetic Model” on page 5-24
“How SimBiology Models Represent Pharmacokinetic Models” on page 5-24
“Create a Pharmacokinetic Model Using the Command Line” on page 5-26
“Dosing Types” on page 5-28
“Elimination Types” on page 5-31
“Intercompartmental Clearance” on page 5-33
“Unit Conversion for Imported Data” on page 5-34

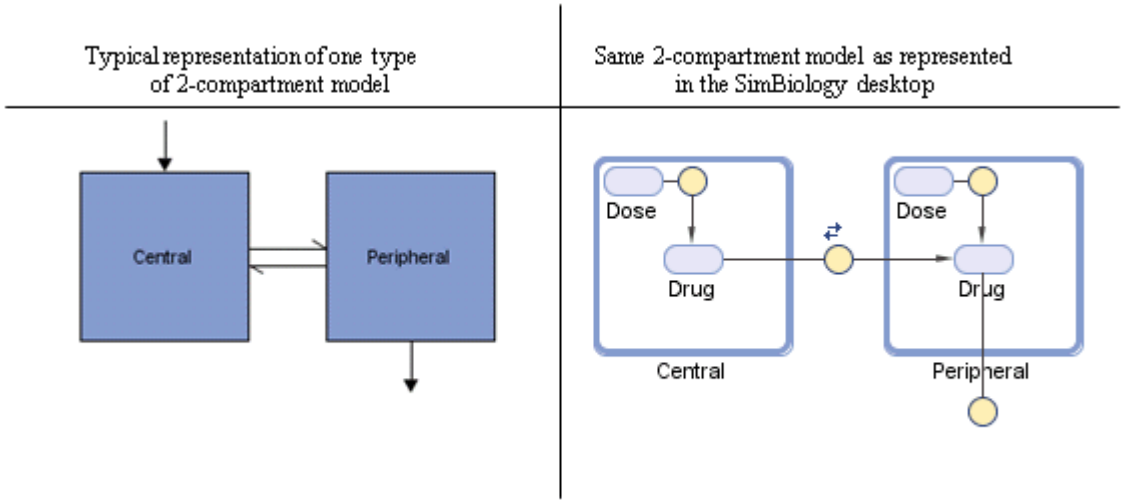
Ways to Create or Import Pharmacokinetic Model

To start modeling, you can:

- Create a PK model using a model construction wizard that lets you specify the number of compartments, the route of administration, and the type of elimination.
- Extend any model to build higher fidelity models.
- Build or load your own model. Load a SimBiology project or SBML model.

How SimBiology Models Represent Pharmacokinetic Models

The following figure compares a model as typically represented in pharmacokinetics with the same model shown in the SimBiology model diagram. For this comparison, assume that you are modeling administration of a drug using a two-compartment model with any dosing input and linear elimination kinetics. (The model structure remains the same with any dosing type.)



Comparison of views in PK

Note the following:

- SimBiology represents the concentration or amount of a drug in a given compartment or volume by a species *object* contained within the compartment.
- SimBiology represents the exchange or flow of the drug between compartments and the elimination of the drug by reactions.
- SimBiology represents intercompartmental clearance by a parameter (`Q`) which specifies the clearance between the compartments.
- SimBiology drives the dosing schedule with a combination of species (`Drug` and/or `Dose`) and reactions (`Dose -> Drug`), depending on whether the administration into the compartment follows bolus, zero-order, infusion, or first-order dosing kinetics. For more information on the components added and parameters estimated, see “Dosing Types” on page 5-28.

You can also view this model as a regression function, `y = f(k,u)`, where `y` is the predicted value, given values of an input `u`, and parameter values `k`. In SimBiology the model represents `f`, and the model is used to generate a regression function if `y`, `k`, and `u` are identified in the model.

Create a Pharmacokinetic Model Using the Command Line

To create a PK model with the specified number of compartments, dosing type, and method of elimination:

1. Create a `PKModelDesign` object. The `PKModelDesign` object lets you specify the number of compartments, route of administration, and method of elimination, which SimBiology uses to construct the model object with the necessary compartments, species, reactions, and rules.

   ```
   pkm = PKModelDesign;
   ```

2. Add a compartment specifying the compartment name, and optionally, the type of dosing, and the method of elimination. Also specify whether the data contains a response variable measured in this compartment and whether the dose(s) have time lags. For example, if using the tobramycin data set [1] on page 5-4, specify a compartment named `Central`, with `Bolus` for the `DosingType` property, `linear-clearance` for the `EliminationType` property, and `true` for the `HasResponseVariable` property.

   ```
   pkc1 = addCompartment(pkm, 'Central', 'DosingType', 'Bolus', ...
                         'EliminationType', 'linear-clearance', ...
                         'HasResponseVariable', true);
   ```

   For a description of other `DosingType` and `EliminationType` property values, see "Dosing Types" on page 5-28 and "Elimination Types" on page 5-31.

   For a description of the `HasResponseVariable` property, see `HasResponseVariable`. At least one compartment in a model must have a response. Although SimBiology supports multiple responses per compartment, when adding compartments to a `PKModelDesign` object, you are limited to one response per compartment.

   Note To add a compartment that has a time lag associated with any dose that targets it, set the `HasLag` property to `true`:

   ```
   pkc_lag = addCompartment(pkm, 'Central', 'DosingType', 'Bolus', ...
                            'EliminationType', 'linear-clearance', ...
                            'HasResponseVariable', true, 'HasLag', true);
   ```

   Or after adding a compartment, set its `HasLag` property to `true`:

   ```
   pkc1.HasLag = true;
   ```

   ---

3 Optionally, add a second compartment named `Peripheral`, with no dosing, no elimination, and no time lag. Set the `HasResponseVariable` property to `true`. If you are using the tobramycin data set [1] on page 5-4, skip this step and use only one compartment.

```
pkc2 = addCompartment(pkm, 'Peripheral', 'HasResponseVariable', true);
```

The model construction process adds the necessary parameters, including a parameter representing intercompartmental clearance `Q`. You can add more compartments by repeating this step. The addition of each compartment creates a chain of compartments in the order of compartment addition, with a bidirectional flow of the drug between compartments in the model.

Use the handle to the compartment (`pkc1` or `pkc2`), to change compartment properties.

4 Construct a SimBiology model object.

```
[modelObj, PKModelMapObj] = pkm.construct
```

The `construct` method returns a SimBiology model object (`modelObj`) and a `PKModelMap` object (`PKModelMapObj`) that contains the mapping of the model components to the elements of the regression function.

Note If you change the `PKModelDesign` object, you must create a new model object using the `construct` method. Changes to the `PKModelDesign` do not automatically propagate to a previously constructed model object.

5 Perform parameter fitting as shown in “Perform Data Fitting with PKPD Models” on page 5-36.

The model object and the `PKModelMap` object are input arguments for the `sbionlmefit`, `sbionlmefitsa` and `sbionlinfit` functions used in parameter fitting.

For information on ...	See ...
Dosing types	“Dosing Types” on page 5-28
Elimination types	“Elimination Types” on page 5-31
Parameter fitting	“Perform Data Fitting with PKPD Models” on page 5-36

For information on ...	See ...
Simulating the model and a description of configuration sets	“Model Simulation” on page 4-2

Dosing Types

When creating models, SimBiology creates the following model components for each compartment in the model, regardless of the dosing type:

- Two species (`Drug_CompartmentName` and `Dose_CompartmentName`) for each compartment.
- A reaction (`Dose_CompartmentName -> Drug_CompartmentName`) for each compartment, governed by mass action kinetics.
- A parameter (`ka_CompartmentName`) for each compartment, representing the absorption rate of the drug when absorption follows first-order kinetics. This is the forward rate parameter for the `Dose_CompartmentName -> Drug_CompartmentName` reaction.
- A parameter (`Tk0_CompartmentName`) for each compartment, representing the duration of drug absorption when absorption follows zero-order kinetics.
- A parameter (`TLag_CompartmentName`) for each compartment, representing the time lag for any dose that targets that compartment and also that is specified as having a time lag.

For dosing types that have a fixed infusion or absorption duration (`infusion` and `zero-order`), you can use overlapping doses. The doses are additive.

The following table describes the dosing types, the default parameters to estimate, and lists the model components created and used for dosing.

Dosing Type	Description	SimBiology Model Components Used	Default Parameters to Estimate
`''`(empty character vector)	No dose	The species (`Drug_CompartmentName`) in each compartment	None

Dosing Type	Description	SimBiology Model Components Used	Default Parameters to Estimate
SimBiology desktop — `bolus` Command line — `Bolus`	Assumes that the drug amount is increased instantly at the dose time. In the SimBiology model, the initial concentration of the drug is based on dose amount and volume of the compartment containing the drug.	The species (`Drug_CompartmentName`) in each compartment	None
SimBiology desktop — `infusion` Command line — `Infusion`	Assumes that the infused drug amount increases at a constant known absorption (or infusion) rate over a known duration. The imported data set must contain the rate and not an infusion duration. SimBiology uses this information to change the species concentration at the constant rate over the duration specified in the data set.	The species (`Drug_CompartmentName`) in each compartment	None
SimBiology desktop — `zero-order` Command line — `ZeroOrder`	Assumes that the drug is added at a constant rate over fixed, but unknown duration.	• The species `Drug_CompartmentName` in each compartment • The parameter (`Tk0_CompartmentName`) in each compartment that has zero-order dosing. This parameter represents the duration of drug absorption	`Tk0_CompartmentName` (absorption duration)

Dosing Type	Description	SimBiology Model Components Used	Default Parameters to Estimate
SimBiology desktop — `first-order` Command line — `FirstOrder`	Assumes that the rate at which the drug is absorbed is not constant. In the SimBiology model, absorption rate is assumed to be governed by mass-action kinetics.	• A species (`Dose_CompartmentName`) representing the dose amount before it is absorbed • A species (`Drug_CompartmentName`) for each compartment • A parameter (`ka_CompartmentName`) representing the absorption rate of the drug • A `MassAction` `reaction` (`Dose_CompartmentName –> Drug_CompartmentName`) with forward rate parameter (`ka_CompartmentName`)	`ka_CompartmentName` (absorption rate)

Elimination Types

Elimination Type	Description	SimBiology Model Components Created	Default Parameters to Estimate
SimBiology desktop — `Linear {Elimination Rate, Volume}` Command line — `'linear'`	Assumes simple mass-action kinetics in the elimination of the drug. In the SimBiology model, elimination is specified by mass-action kinetics with the elimination rate constant specified by the forward rate parameter (`ke`).	• A parameter representing the elimination rate (`ke_CompartmentName`) • A `MassAction reaction (drug –> null)` with forward rate parameter (`ke_CompartmentName`) specific to the compartment	• Compartment volume (`Capacity` property) • Elimination rate constant (`ke_CompartmentName`) • Inter-compartmental clearance (`Q`) when there is more than one compartment. See “Intercompartmental Clearance” on page 5-33.

Elimination Type	Description	SimBiology Model Components Created	Default Parameters to Estimate
SimBiology desktop — `Linear {Clearance, Volume}` Command line — `'linear-clearance'`	Assumes simple mass-action kinetics in the elimination of the drug. In the SimBiology model, similar to `Linear {Elimination Rate, Volume}`. But, in addition, this option lets you specify the model in terms of clearance (`Cl`) where, `Cl = ke * volume`).	• A parameter representing the clearance (`Cl_CompartmentName`) • A parameter representing the elimination rate constant (`ke_CompartmentName`) • An `InitialAssignment` rule that initializes `ke_CompartmentName` based on the initial values for `Cl_CompartmentName` and compartment volume • A `MassAction reaction (drug –> null)` with forward rate parameter (`ke_CompartmentName`)	• Compartment volume (`Capacity` property) • Clearance (`Cl_CompartmentName`) • Inter-compartmental clearance (`Q`) when there is more than one compartment. See “Intercompartmental Clearance” on page 5-33.

Elimination Type	Description	SimBiology Model Components Created	Default Parameters to Estimate
SimBiology desktop — `Enzymatic (Michaelis-Menten)` Command line — `'enzymatic'`	Assumes that elimination is governed by Michaelis-Menten kinetics.	• Parameter representing the Michaelis constant, (`Km_CompartmentName`) • A parameter for maximum velocity (`Vm_CompartmentName` • A reaction with Michaelis-Menten kinetics (`drug -> null`), with kinetic law parameters `Vm_CompartmentName` and `Km_CompartmentName`	• Compartment volume (`Capacity` property) • Parameter (`Km_CompartmentName`) • Parameter (`Vm_CompartmentName`) • Inter-compartmental clearance (`Q`) when there is more than one compartment. See “Intercompartmental Clearance” on page 5-33

Intercompartmental Clearance

The compartments created when you generate a SimBiology model form a chain and each pair of linked compartments are connected by a transport reaction similar to linear elimination. The addition of two compartments, `C1` and `C2`, generates a reversible mass-action reaction `C1.Drug_C1 <-> C2.Drug`. The forward rate parameter is the compartmental clearance, Q_{12}, divided by the volume of `C1`. The reverse rate parameter is Q_{12}, divided by the volume of `C2`.

The process of adding each pair of compartments in the chain C_m and C_n generates the following model components:

- A parameter Q_{mn} representing the compartmental clearance between those two compartments. This parameter is added to the list of parameters to be estimated (`Estimated` property of `PKModelMap` object).
- A parameter (`kmn`) representing the rate of transfer of the drug from `Cm` to `Cn`, where $k_{mn} = Q_{mn}/V_m$.

- A parameter (`knm`) representing the rate of `Cn` to `Cm`, where $k_{nm} = Q_{mn}/V_n$.
- A reversible mass-action reaction between the two compartments, `Cm.Drug_Cm <-> Cn.Drug_Cn`, with forward rate parameter `kmn`, and reverse rate parameter `knm`.
- An initial assignment rule that initializes the value of the parameter `kmn`, based on the initial values for `Cm` and `Qmn`.
- An initial assignment rule that initializes the value of the parameter `knm`, based on the initial values for `Cn` and `Qmn`.

Unit Conversion for Imported Data

Unit conversion converts the matching physical quantities to one consistent unit system in order to resolve them. This conversion is in preparation for correct simulation, but SimBiology returns the physical quantities in the model in units that you specify.

Regardless of whether unit conversion is `on` or `off`, you must express dosing data in amount. By default, **Unit Conversion** is `off`, so you must ensure that units for the data and the model are consistent with one another.

If **Unit Conversion** is `on`, you must specify units. If using the SimBiology desktop, specify units in the **Raw Data** tab, when data is selected in the **Project Explorer**. If using the command line, specify units in the `PKData` object.

Parameters in the model have default units. If unit conversion is `on`, you can change the units as long as the dimensions are consistent. These default units, which you might use to specify the values for the initial guess, are as follows.

Physical Quantity or Model Parameter	Unit
Central or peripheral compartment volume (*Central* or *Peripheral*)	`liter`
First-order elimination rate (*ke*)	`1/second`
Michaelis constant (*Km*)	`milligram/liter`
Maximum reaction-velocity, Michaelis-Menten kinetics (*Vm*)	`milligram/second`
Clearance (*Cl*)	`liter/second`
Absorption duration (*Tk0*)	`second`
Absorption rate (*ka*)	`1/second`

Use the configuration settings options to turn unit conversion `on` or `off`. For details, see “Model Simulation” on page 4-2.

For details on dimensional analysis for reaction rates, see “How Reaction Rates Are Evaluated” on page 2-14.

Perform Data Fitting with PKPD Models

SimBiology lets you estimate model parameters by fitting the model to experimental time-course data, using either nonlinear regression or mixed-effects (NLME) techniques. You can perform both individual and population fits to grouped data.

- Individual fit — Fit data using nonlinear regression (least-squares) methods, specify parameter transformations, estimate parameters, and calculate residuals and the estimated coefficient covariance matrix. For a command line workflow, see "Fitting Workflow for sbiofit" on page 4-55. For the SimBiology desktop, see "Fit Data" on page 1-71.
- Population fit — Fit data, specify parameter transformations, and estimate the fixed effects and the random sources of variation on parameters using nonlinear mixed-effects models. For a command line workflow, see "Nonlinear Mixed-Effects Modeling Workflow" on page 4-44. For the desktop, see "Fit Data" on page 1-71.
- Population fit using a stochastic algorithm — Fit data, specify parameter transformations, and estimate the fixed effects and the random sources of variation on parameters, using the Stochastic Approximation Expectation-Maximization (SAEM) algorithm. SAEM is more robust with respect to starting values. This functionality relaxes assumption of constant error variance. Specify `nlmefitsa` as the estimation function name when you run `sbiofitmixed` or in the Fit Data task of the desktop.

In addition, you can turn on the ProgressPlot on page 4-61 option to get the live feedback on the status of parameter estimation.

See Also

`sbiofit` | `sbiofitmixed`

More About

- "Nonlinear Regression" on page 4-50
- "Nonlinear Mixed-Effects Modeling" on page 4-42
- "Configuring Tasks" on page 1-70

A

Creating Reaction Rates

Define Reaction Rates with Mass Action Kinetics

Use mass action kinetics to define zero-order, first-order, second-order, and reversible reactions.

Definition of Mass Action Kinetics

Mass action describes the behavior of reactants and products in an elementary chemical reaction. Mass action kinetics describes this behavior as an equation where the velocity or rate of a chemical reaction is directly proportional to the concentration of the reactants.

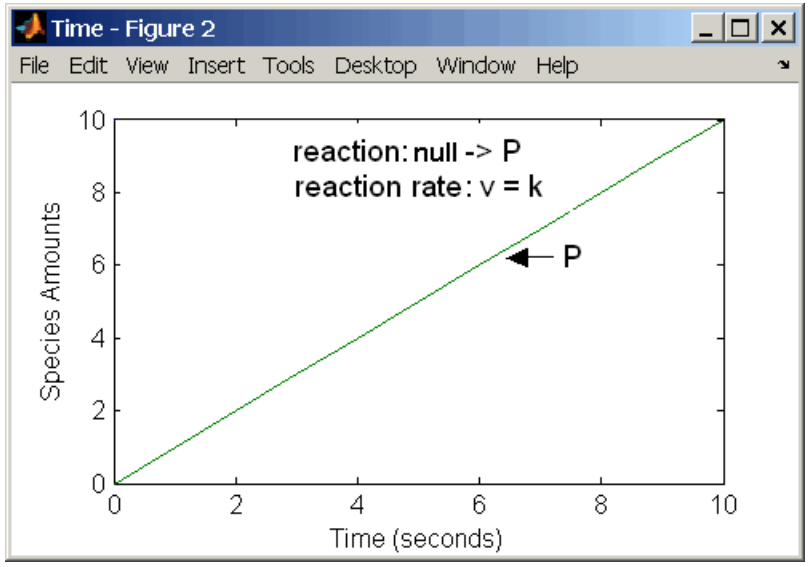
Zero-Order Reactions

With a zero-order reaction, the reaction rate does not depend on the concentration of reactants. Examples of zero-order reactions are synthesis from a `null` species, and modeling a source species that is added to the system at a specified rate.

```
     reaction: null -> P
reaction rate: k mole/second
      species: P =  0 mole
   parameters: k =  1 mole/second
```

Note When specifying a `null` species, the reaction rate must be defined in units of amount per unit time not concentration per unit time.

Entering the reaction above into the software and simulating produces the following result:



Zero-Order Mass Action Kinetics

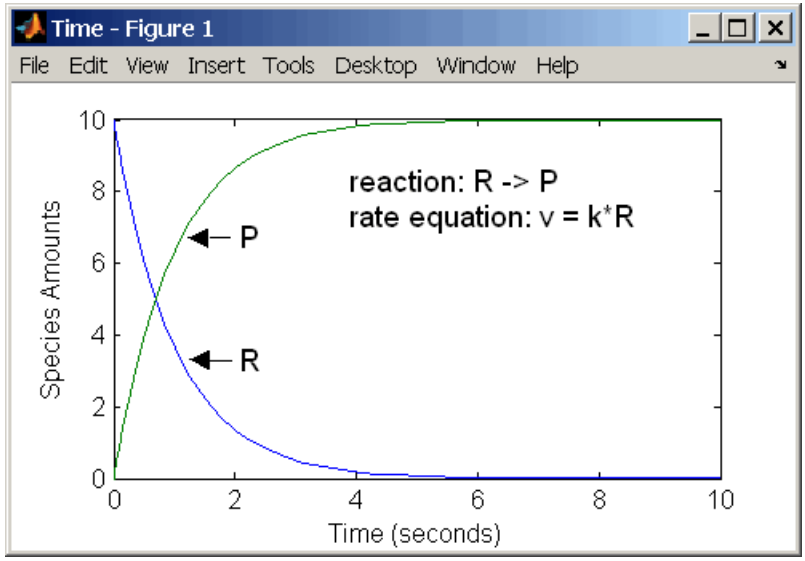
Note If the amount of a reactant with zero-order kinetics reaches zero before the end of a simulation, then the amount of reactant can go below zero regardless of the solver or tolerances you set.

First-Order Reactions

With a first-order reaction, the reaction rate is proportional to the concentration of a single reactant. An example of a first-order reaction is radioactive decay.

```
     reaction: R -> P
reaction rate: k*R mole/(liter*second)
      species: R = 10 mole/liter
               P =  0 mole/liter
   parameters: k =  1 1/second
```

Entering the reaction above into the software and simulating produces the following results:



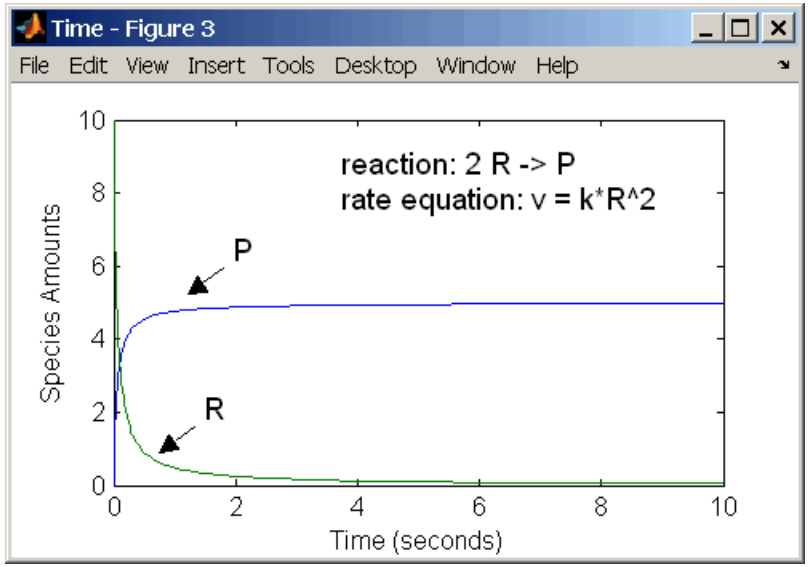
First-Order Mass Action Kinetics

Second-Order Reactions

A second-order reaction has a reaction rate that is proportional to the square or the concentration of a single reactant or proportional to two reactants. Notice the space between the reactant coefficient and the name of the reactant. Without the space, 2R would be considered the name of a species.

```
     reaction: 2 R -> P
reaction rate: k*R^2 mole/(liter*second)
      species: R = 10 mole/liter
               P =  0 mole/liter
   parameters: k =  1 liter/(mole*second)
```

Entering the reaction above into the software and simulating produces the following results:

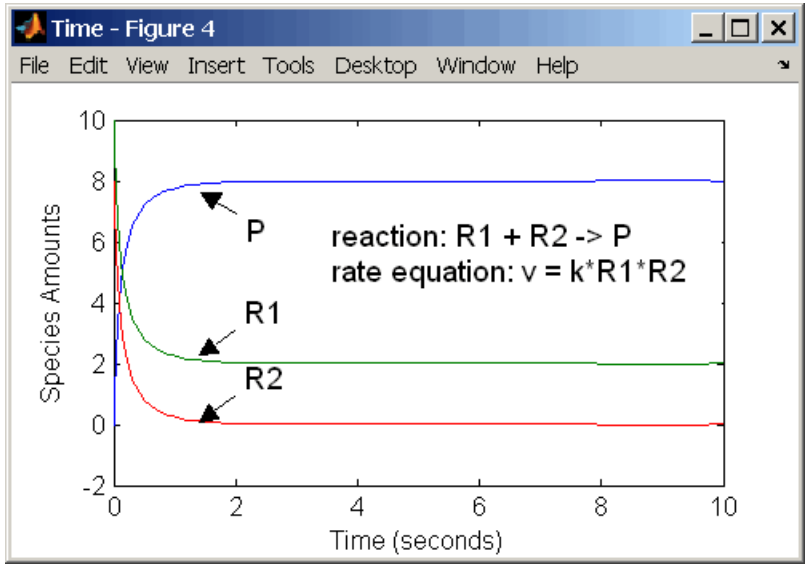


Second-Order Kinetics with Single Reactant

With two reactants, the reaction rate depends on the concentration of two of the reactants.

```
     reaction: R1 + R2 -> P
reaction rate: k*R1*R2 mole/(liter*second)
      species: R1 = 10 mole/liter
               R2 =  8 mole/liter
                P =  0 mole/liter
   parameters:  k =  1 liter/(mole*second)
```

Enter the reaction above into the software and simulating produces the following results. There is a difference in the final values because the initial amount of one of the reactants is lower than the other. After the first reactant is used up, the reaction stops.



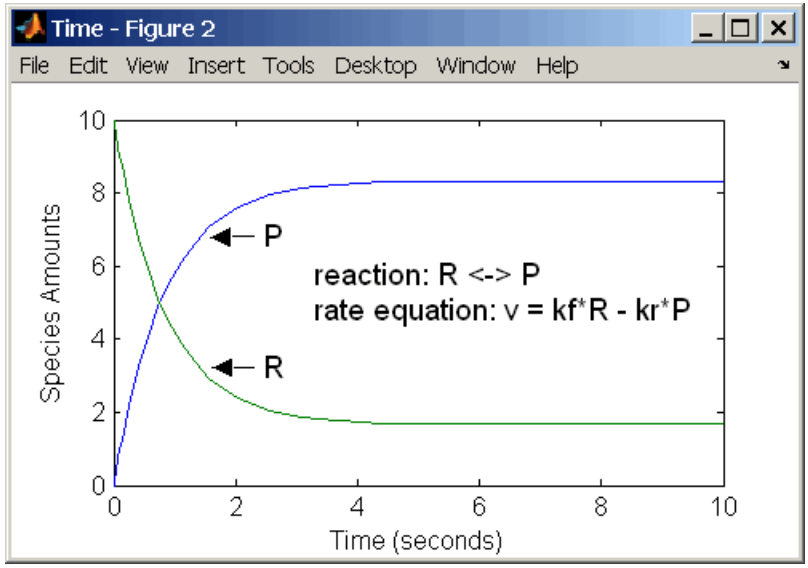
Second-Order Kinetics with Two Reactants

Reversible Mass Action

You can model reversible reactions with two separate reactions or with one reaction. With a single reversible reaction, the reaction rates for the forward and reverse reactions are combined into one expression. Notice the angle brackets before and after the hyphen to represent a reversible reaction.

```
     reaction: R <-> P
reaction rate: kf*R - kr*P mole/(liter*second)
      species: R = 10   mole/liter
               P =  0   mole/liter
   parameters: kf = 1   1/second
               kr = 0.2 1/second
```

Entering the reaction above into the software and simulating produces the following results. At equilibrium when the rate of the forward reaction equals the reverse reaction, `v = kf*R - kr*P = 0` and `P/R = kf/kr`.


Time - Figure 2
File Edit View Insert Tools Desktop Window Help
Species Amounts
Time (seconds)
P
R
reaction: R <-> P
rate equation: v = kf*R - kr*P
0
2
4
6
8
10

Define Reaction Rates with Enzyme Kinetics

Use differential equations, mass action kinetics, or Michaelis-Menten kinetics to define enzyme reactions.

Simple Model for Single Substrate Catalyzed Reactions

A simple model for enzyme-catalyzed reactions starts a substrate S reversibly binding with an enzyme E. Some of the substrate in the substrate/enzyme complex is converted to product P with the release of the enzyme.

$$S + E \underset{k1r}{\overset{k1}{\rightleftharpoons}} ES \xrightarrow{k2} E + P$$

$$v1 = k1[S][E], \quad v1r = k1r[ES], \quad v2 = k2[ES]$$

This simple model can be defined with

- Differential rate equations. See “Enzyme Reactions with Differential Rate Equations” on page A-8.
- Reactions with mass action kinetics. See “Enzyme Reactions with Mass Action Kinetics” on page A-10.
- Reactions with Henri-Michaelis-Menten kinetics. See “Enzyme Reactions with Irreversible Henri-Michaelis-Menten Kinetics” on page A-11.

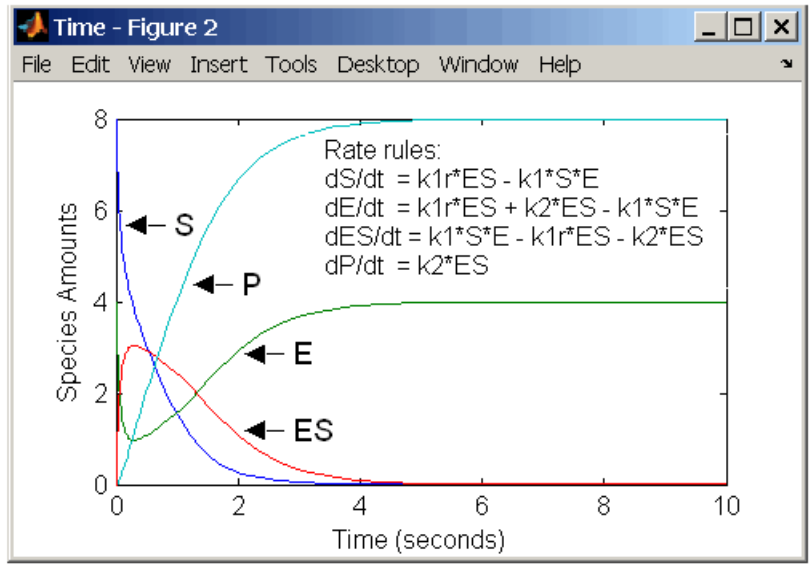
Enzyme Reactions with Differential Rate Equations

The reactions for a single-substrate enzyme reaction mechanism (see “Simple Model for Single Substrate Catalyzed Reactions” on page A-8) can be described with differential rate equations. You can enter the differential rate equations into the software as rate rules.

```
     reactions: none
 reaction rate: none
    rate rules: dS/dt  = k1r*ES - k1*S*E
                dE/dt  = k1r*ES + k2*ES - k1*S*E
                dES/dt = k1*S*E - k1r*ES - k2*ES
                dP/dt  = k2*ES
       species: S =  8   mole
                E =  4   mole
```

```
              ES =  0   mole
               P =  0   mole
    parameters:  k1 = 2   1/(mole*second)
                k1r = 1   1/second
                 k2 = 1.5 1/second
```

Remember that the rate rule `dS/dt = f(x)` is written in a SimBiology rate rule expression as `S = f(x)`. For more information about rate rules see “Rate Rules” on page 2-19.



Alternatively, you could remove the rate rule for `ES`, add a new species `Etotal` for the total amount of enzyme, and add an algebraic rule `0 = Etotal - E - ES`, where the initial amounts for `Etotal` and `E` are equal.

```
      reactions: none
  reaction rate: none
     rate rules: dS/dt = k1r*ES - k1*S*E
                 dE/dt = k1r*ES + k2*ES - k1*S*E
                 dP/dt = k2*ES
 algebraic rule: 0 = Etotal - E - ES
        species: S =  8   mole
                 E =  4   mole
                ES =  0   mole
```

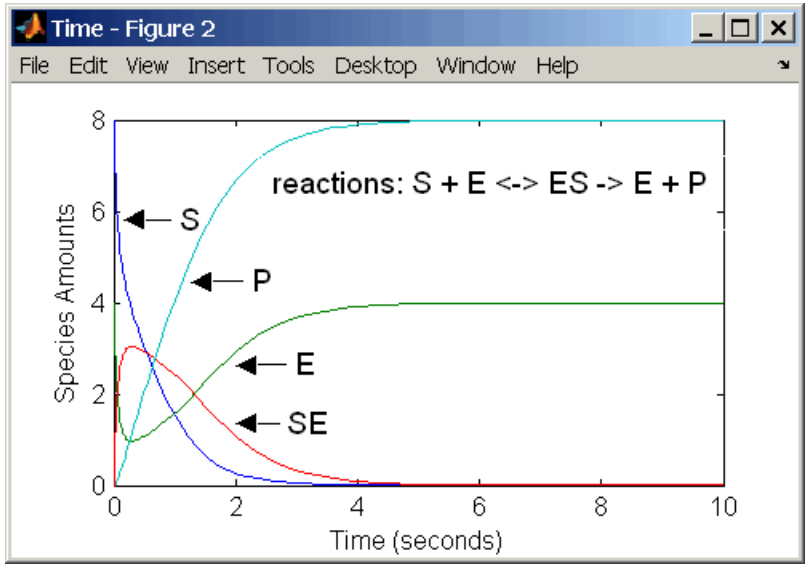
```
              P =  0   mole
         Etotal =  4   mole
     parameters: k1 = 2   1/(mole*second)
                k1r = 1   1/second
                 k2 = 1.5 1/second
```

Enzyme Reactions with Mass Action Kinetics

Determining the differential rate equations for the reactions in a model is a time-consuming process. A better way is to enter the reactions for a single substrate enzyme reaction mechanism directly into the software. The following example using models an enzyme catalyzed reaction with mass action kinetics. For a description of the reaction model, see “Simple Model for Single Substrate Catalyzed Reactions” on page A-8.

```
     reaction: S + E -> ES
reaction rate: k1*S*E (binding)

     reaction: ES -> S + E
reaction rate: k1r*ES (unbinding)

     reaction: ES -> E + P
reaction rate: k2*ES (transformation)
      species: S =  8   mole
               E =  4   mole
              ES =  0   mole
               P =  0   mole
   parameters: k1  = 2   1/(mole*second)
               k1r = 1   1/second
               k2  = 1.5 1/second
```

The results for a simulation using reactions are identical to the results from using differential rate equations.



Enzyme Reactions with Irreversible Henri-Michaelis-Menten Kinetics

Representing an enzyme-catalyzed reaction with mass action kinetics requires you to know the rate constants `k1`, `k1r`, and `k2`. However, these rate constants are rarely reported in the literature. It is more common to give the rate constants for Henri-Michaelis-Menten kinetics with the maximum velocity `Vm=k2*E` and the constant `Km = (k1r + k2)/k1`. The reaction rate for a single substrate enzyme reaction using Henri-Michaelis-Menten kinetics is given below. For information about the model, see “Simple Model for Single Substrate Catalyzed Reactions” on page A-8.

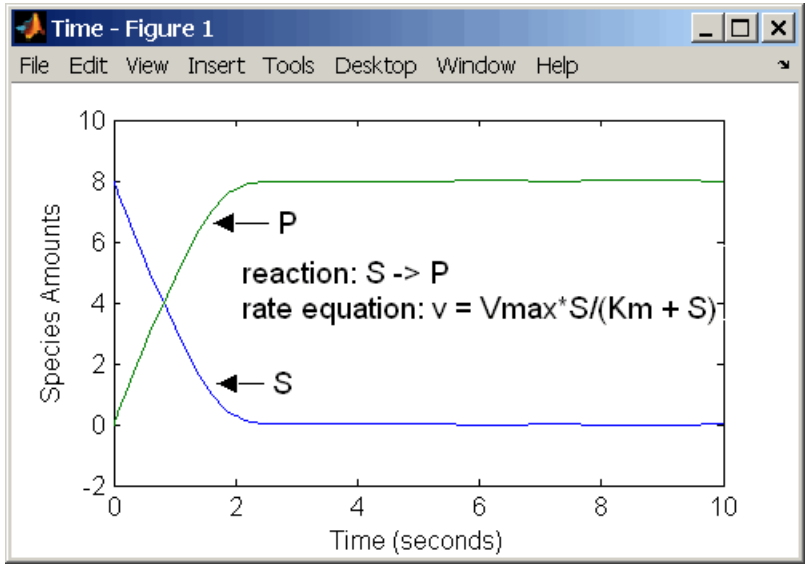
$$v = \frac{Vmax[S]}{Km + [S]}$$

The following example models an enzyme catalyzed reaction using Henri-Michaelis-Menten kinetics with a single reaction and reaction rate equation. Enter the reaction defined below into the software and simulate.

```
     reaction: S -> P
reaction rate: Vmax*S/(Km + S)
```

```
    species:    S =  8    mole
                P =  0    mole
parameters: Vmax =  6    mole/second
               Km =  1.25 mole
```

The results show a plot slightly different from the plot using mass action kinetics. The differences are due to assumptions made when deriving the Michaelis-Menten rate equation.



B

Models Used in Examples

Minimal Cascade Model for a Mitotic Oscillator

Albert Goldbeter modified a model with enzyme cascades [Goldbeter and Koshland 1981 on page B-17] to fit cell cycle data from studies with embryonic cells [Goldbeter 1991 on page B-17]. 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.

In this section...
“Goldbeter Model” on page B-2
“SimBiology Model with Rate Rules” on page B-5
“SimBiology Model with Reactions” on page B-7
“References” on page B-17

Goldbeter Model

- “About the Goldbeter Model” on page B-2
- “Reaction Descriptions and Model Assumptions” on page B-3
- “Mathematical Model” on page B-4

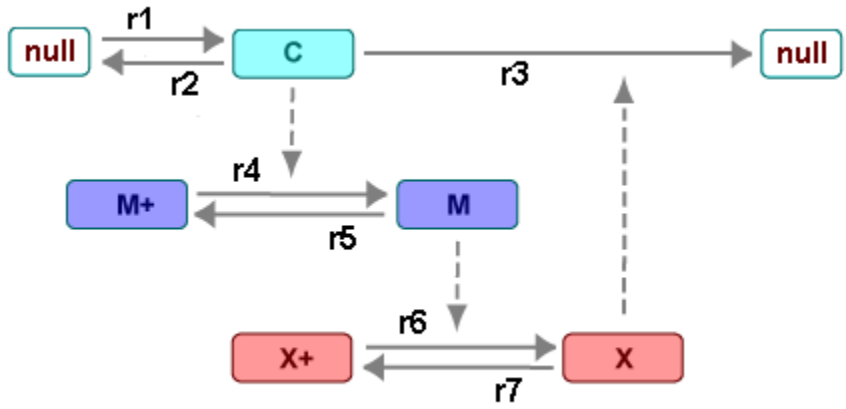
About the Goldbeter Model

Albert Goldbeter created a simple cell division model from studies with embryonic cells [Goldbeter 1991 on page B-17]. 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 on page B-17].

- 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 on page B-17].

- 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 on page B-17] for cyclin (C).

$$\frac{dC}{dt} = v_i - v_d X \frac{C}{K_d + C} - k_d C$$

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+`).

$$\frac{dM}{dt} = V_1 \frac{(1-M)}{K_1 + (1-M)} - V_2 \frac{M}{K_2 + M}$$

$$V_1 = \frac{VM_1[C]}{K_c + [C]}$$

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+`).

$$\frac{dX}{dt} = V_3 \frac{(1-X)}{K_3 + (1-X)} - V_4 \frac{X}{K_4 + X}$$

$$V_3 = VM_3[M]$$

SimBiology Model with Rate Rules

- "SimBiology Model with Rules" on page B-5
- "SimBiology Simulation with Rules" on page B-6

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 B-4:

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

The following equations are the rate and `repeatedAssignment` rules for "Differential Rate Equation 2, Kinase (M)" on page B-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 `repeatedAssignment` rules for “Differential Rate Equation 3, Protease (X)” on page B-4:

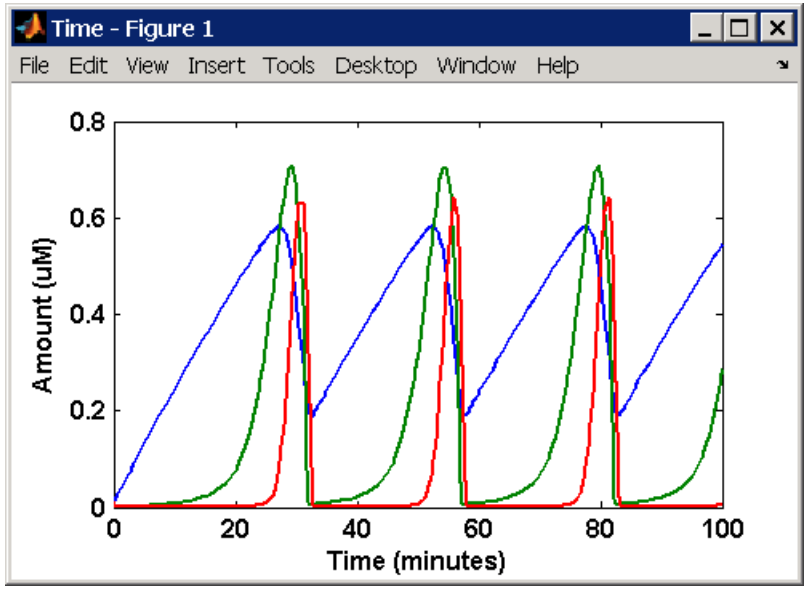
```
dX/dt = (V3*Xplus)/(K3 + Xplus) - (V4*X)/(K4 + X)
V3 = VM3*M
Xplus = Xt - X
```

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 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.

SimBiology Simulation with Rules

This is a simulation of Goldbeter's minimal mitotic oscillator using differential rate and algebraic equations. Simulate with the `sundials` solver and plot species `C`, `M`, and `X`. For a description of the model, see “SimBiology Model with Rules” on page B-5.



SimBiology Model with Reactions

- "Converting Differential Rate Equations to Reactions" on page B-7
- "Calculating Initial Values for Reactions" on page B-9
- "SimBiology Simulation with Reactions" on page B-16

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 B-4.

$$\frac{dC}{dt} = v_i - v_d X \frac{C}{K_d + C} - k_d C$$

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 B-4.

$$\frac{dM}{dt} = V_1 \frac{(1 - M)}{K_1 + (1 - M)} - V_2 \frac{M}{K_2 + M}$$

$$V_1 = \frac{VM_1[C]}{K_c + [C]}$$

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)
 repeatedAssignment 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.

$$\frac{dX}{dt} = V_3 \frac{(1-X)}{K_3 + (1-X)} - V_4 \frac{X}{K_4 + X}$$

```
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)
  repeatedAssignment 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
		12044	D. molecule
species	`C` (substrate)	0.01	A. uM

R3		Value	Units
		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
`repeatedAssignment` 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)
		5.0000e-020	C. mole/second
		30110	D. molecule/second
parameter	`Kc`	0.5	A. uM
		5.0000e-7	B. mole/liter

R4		Value	Units
		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

R5		Value	Units
		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
repeatedAssignment rule	V3 = VM3*M	----	

R6		Value	Units
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
		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

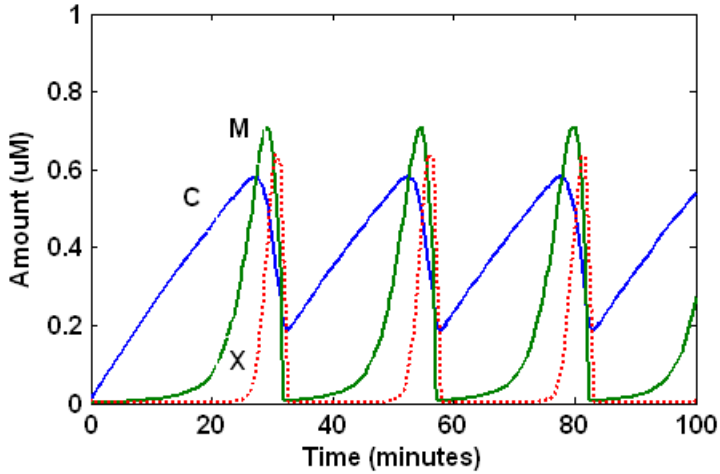
Reaction 7 Protease Deactivation

R7		Value	Units
reaction	X -> X_plus	----	----
reaction rate	(V4*X)/(K4 + X)	----	A. uM/minute

R7		Value	Units
		----	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
		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 `sundials` solver and plot species `C`, `M`, and `X`. For a description of the model, see “SimBiology Model with Reactions” on page B-7.



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.

Model of the Yeast Heterotrimeric G Protein Cycle

In this section...
“Background on G Protein Cycles” on page B-18
“Modeling a G Protein Cycle” on page B-19
“References” on page B-22

Background on G Protein Cycles

- “G Proteins” on page B-18
- “G Proteins and Pheromone Response” on page B-19

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 on page B-22].

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

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.

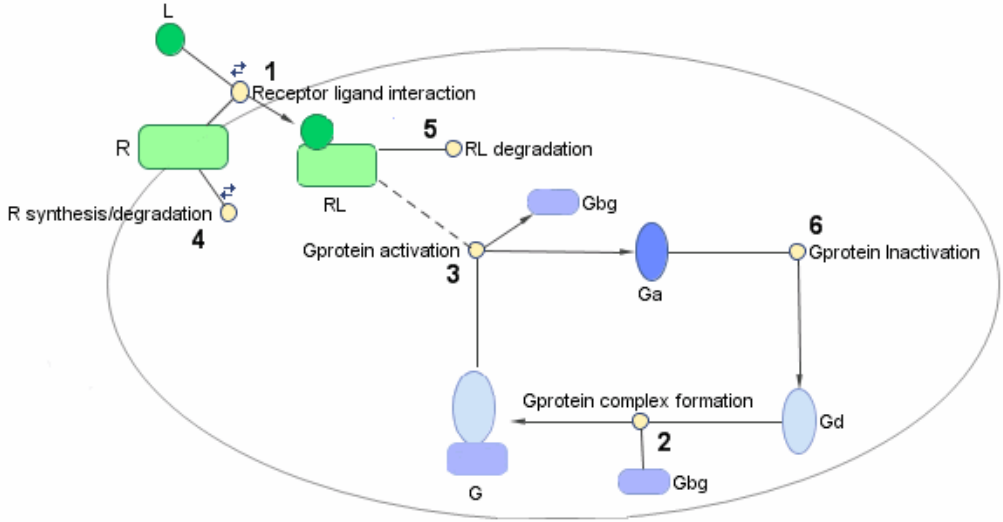
Modeling a G Protein Cycle

- “Reactions Overview” on page B-19
- “Assumptions, Experimental Data, and Units in the G Protein Model” on page B-21

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 on page B-22] 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`	`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`
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 B-5.

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 on page B-22]. The following values were obtained by comparing the coordinates of the standards with those of the unknowns in the figure.

Time	Fraction of Active Ga (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.

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.

See Also

Related Examples

- “Parameter Scanning, Parameter Estimation, and Sensitivity Analysis in the Yeast Heterotrimeric G Protein Cycle”

Model of 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 on page B-56]. 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.

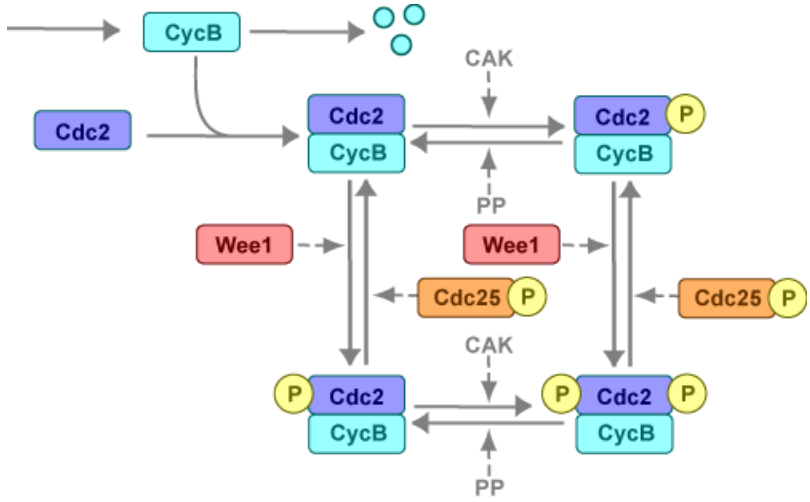
In this section...
“M-Phase Control Model” on page B-23
“M-Phase Control Equations” on page B-25
“SimBiology Model with Rate and Algebraic Rules” on page B-33
“SimBiology Model with Reactions and Algebraic Rules” on page B-39
“References” on page B-56

M-Phase Control Model

- “Synthesis Reactions” on page B-23
- “Regulation Reactions with Active MPF” on page B-24

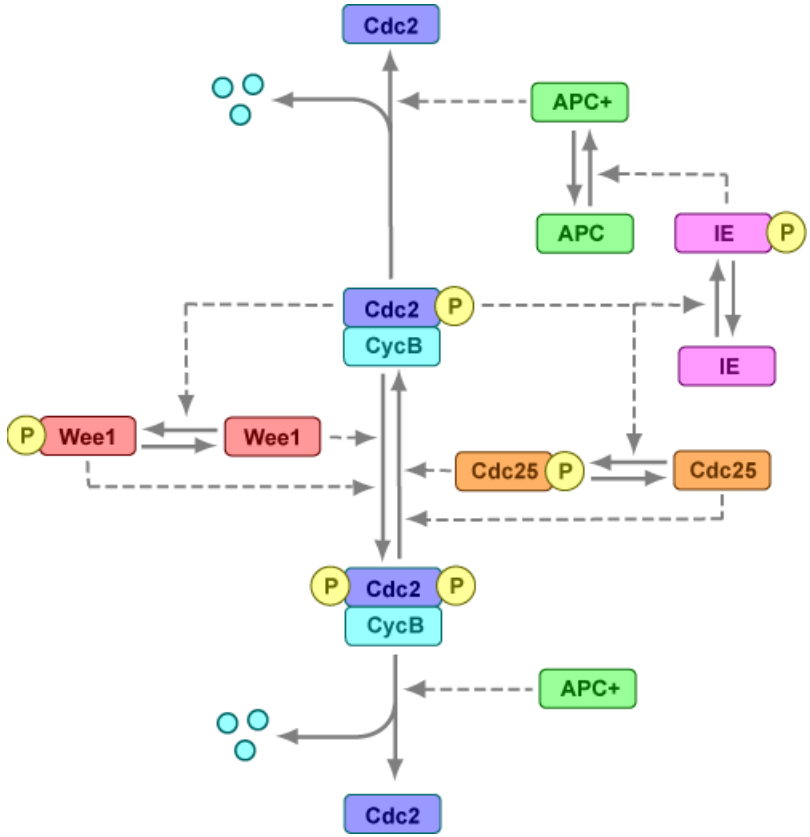
Synthesis Reactions

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.



M-Phase Control Equations

- “About the Rate Equations in This Example” on page B-26
- “Converting Differential Equations to Reactions” on page B-26
- “Equation 1, Cyclin B” on page B-27
- “Equation 2, M-Phase Promoting Factor” on page B-27
- “Equation 3, Inhibited M-Phase Promoting Factor” on page B-28
- “Equation 4, Inhibited and Activated M-Phase Promoting Factor” on page B-29
- “Equation 5, Activated M-Phase Promoting Factor” on page B-29
- “Equation 11, Cell Division Control 25” on page B-30

- "Equation 12, Wee1 Activation/Deactivation" on page B-31
- "Equation 13, Intermediate Enzyme Activation/Deactivation" on page B-31
- "Equation 14, APC Activation/Deactivation" on page B-32
- "Equation 17, Rate Parameter K2" on page B-32
- "Equation 18, Rate Parameter Kcdc25" on page B-33
- "Equation 19, Rate Parameter Kwee1" on page B-33

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 on page B-56]. 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.

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

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 on page B-56].

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

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 B-33.

```
Rule 1 on page B-34  [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 B-39.

```
Reaction 1 on page B-40   AA -> CycB           v = k1
Reaction 2 on page B-41   CycB -> AA           v = K2*[CycB]
Reaction 3 on page B-42   Cdc2 + CycB -> MPF   v = k3*[Cdc2]*[CycB]
```

Equation 2, M-Phase Promoting Factor

Differential rate equation for M-phase promoting factor (MPF) [Marlovits 1998 on page B-56]. Note that the parameter name `kcakr` [Marlovits et al. 1998 on page B-56] is changed to `kpp` [Borisuk 1998 on page B-56] in the following reaction equations. MPF is a heterodimer of cdc2 kinase and cyclin B.

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

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 B-33.

```
Rule 2 on page B-35 MPF = kpp*MPFp - (Kwee1 + kcak + K2)*MPF + Kcdc25*pMPF + k3*Cdc2*Cy
```

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 B-39. A reaction name in parentheses denotes a reaction repeated in another differential rate equation.

```
(Reaction 3 on page B-42) Cdc2 + CycB -> MPF     v = k3*[Cdc2]*[CycB]
Reaction 4 on page B-43   MPF -> Cdc2 + AA       v = K2*[MPF]
Reaction 5 on page B-44   MPFp -> MPF            v = kpp*[MPFp]
Reaction 6 on page B-45   MPF -> MPFp            v = kcak*[MPF]
Reaction 7 on page B-46   pMPF -> MPF            v = Kcdc25*[pMPF]
Reaction 8 on page B-47   MPF -> pMPF            v = Kwee1*[MPF]
```

Equation 3, Inhibited M-Phase Promoting Factor

Differential rate equation for inhibited M-phase promoting factor (pMPF) [Marlovits 1998 on page B-56].

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

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 B-33.

```
Rule 3 on page B-36 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 B-39.

```
Reaction 11 on page B-49  pMPF -> Cdc2 + AA      v = K2*[pMPF]
Reaction 12 on page B-49  pMPFp -> pMPF          v = kpp*[pMPFp]
Reaction 13 on page B-49  pMPF -> pMPFp          v = kcak*[pMPF]
(Reaction 8 on page B-47) MPF -> pMPF            v = Kwee1*[MPF]
(Reaction 7 on page B-46) 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 on page B-56].

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

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 B-33.

```
Rule 4 on page B-37 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 B-39.

```
Reaction 15 on page B-50   pMPFp -> Cdc2 + AA       v = K2*[pMPFp]
(Reaction 13 on page B-49) pMPF -> pMPFp           v = kcak*[pMPF]
(Reaction 12 on page B-49) pMPFp -> pMPF           v = kpp*[pMPFp]
Reaction 16 on page B-50   MPFp -> pMPFp           v = Kwee1*[MPFp]
Reaction 17 on page B-50   pMPFp -> MPFp           v = Kcdc25*[pMPFp]
```

Equation 5, Activated M-Phase Promoting Factor

Differential rate equation for activated M-phase promoting factor (MPFp) [Marlovits 1998 on page B-56].

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

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 B-33.

```
Rule 5 on page B-38  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 B-39.

```
Reaction 19 on page B-51    MPFp -> MPF + AA        v = K2*[MPFp]
(Reaction 6 on page B-45)  MPF -> MPFp             v = kcak*[MPF]
(Reaction 5 on page B-44)  MPFp -> MPF             v = kpp*[MPFp]
(Reaction 17 on page B-50) pMPFp -> MPFp           v = Kcdc25*[pMPFp]
(Reaction 16 on page B-50) MPFp -> pMPFp           v = Kwee1*[MPFp]
```

Equation 11, Cell Division Control 25

Differential rate equation for activating and deactivating Cdc25 [Marlovits 1998 on page B-56].

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

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 B-33. Note that since there isn't a rate rule for `Cdc25`, its amount is written as `(TotalCdc25 - Cdc25p)`.

```
Rule 11 on page B-38 Cdc25p = (k25*MPFp*(TotalCdc25 - Cdc25p))/(Km25 + (TotalCdc25 - Cd
```

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 B-39.

```
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 on page B-56]. 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 B-33.

```
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 B-39.

```
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 on page B-56]. 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 B-33.

```
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 B-39.

```
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 on page B-56].

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

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 B-33.

```
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 B-39.

```
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 on page B-56]. 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 B-33.

```
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 B-39. `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 on page B-56]. 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 B-33.

```
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 B-39.

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

Equation 19, Rate Parameter Kwee1

Algebraic equation to define the rate parameter [Marlovits 1998 on page B-56]. 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 B-33.

```
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 B-39.

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

SimBiology Model with Rate and Algebraic Rules

- "Overview" on page B-34

- "Writing Differential Rate Equations as Rate Rules" on page B-34
- "Species" on page B-35
- "Parameters" on page B-36
- "Rate Rule 1, Cyclin B (CycB)" on page B-37
- "Rate Rule 2, M-Phase Promoting Factor (MPF)" on page B-37
- "Rate Rule 3, Inhibited M-Phase Promoting Factor (pMPF)" on page B-37
- "Rate Rule 4, Activated but Inhibited M-Phase Promoting Factor (pMPFp)" on page B-37
- "Rate Rule 5, Activated M-Phase Promoting Factor (MPFp)" on page B-38
- "Rate Rule 11, Activated Cdc25 (Cdc25p)" on page B-38
- "Rate Rule 12, Inhibited Wee1 (Wee1p)" on page B-38
- "Rate Rule 13, Activated Intermediate Enzyme (IEp)" on page B-38
- "Rate Rule 14, Activated APC (APCa)" on page B-38
- "Algebraic Rule 17, Rate Parameter K2" on page B-38
- "Algebraic Rule 18, Rate Parameter Kcdc25" on page B-39
- "Algebraic Rule 19, Rate Parameter Kwee1" on page B-39

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 on page B-56]. For a list and description of the equations, see "M-Phase Control Equations" on page B-25.

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 B-27.

```
 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 B-38. 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 B-27.

```
 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 B-38, "Algebraic Rule 18, Rate Parameter Kcdc25" on page B-39, and "Algebraic Rule 19, Rate Parameter Kwee1" on page B-39.

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

The rate rule is from "Equation 3, Inhibited M-Phase Promoting Factor" on page B-28.

```
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 B-29.

```
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 B-29.

```
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 B-30.

```
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 B-31.

```
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 B-31.

```
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 B-32.

```
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 B-32.

```
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 B-39.

```
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 B-33.

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

SimBiology Model with Reactions and Algebraic Rules

- "Overview" on page B-40
- "Reaction 1, Synthesis of Cyclin B" on page B-40
- "Reaction 2, Degradation of Cyclin B" on page B-41
- "Reaction 3, Dimerization of Cyclin B with Cdc2 Kinase" on page B-42
- "Reaction 4, Degradation of Cyclin B on MPF" on page B-43
- "Reaction 5, Deactivation of Active MPF" on page B-44
- "Reaction 6, Activation of MPF" on page B-45
- "Reaction 7, Remove Inhibiting Phosphate from Inhibited MPF" on page B-46
- "Reaction 8, Inhibition of MPF by Phosphorylation" on page B-47
- "Reaction 11, Degradation of Cyclin B on Inhibited MPF" on page B-49
- "Reaction 12, Deactivation of MPF to Inhibited MPF" on page B-49
- "Reaction 13, Activation of Inhibited MPF" on page B-49
- "Reaction 15, Degradation of Cyclin B on Active but Inhibited MPF" on page B-50
- "Reaction 16, Inhibit MPF by Phosphorylation" on page B-50
- "Reaction 17, Remove Inhibiting Phosphate from Activated MPF" on page B-50

- "Reaction 19, Degradation of Cyclin B on Activated MPF" on page B-51
- "Reaction 36, Activation of Cdc25 by Activated MPF" on page B-51
- "Reaction 37, Deactivation of Cdc25" on page B-51
- "Reaction 38, Deactivation of Wee1 by Active MPF" on page B-52
- "Reaction 39, Activation of Wee1" on page B-52
- "Reaction 40, Activation of Intermediate Enzyme by Active MPF" on page B-52
- "Reaction 41, Deactivation of IE" on page B-52
- "Reaction 42, APC Activation by IEp" on page B-53
- "Reaction 43, APC Deactivation" on page B-53
- "Block Diagram of the M-Phase Control Model with Reactions" on page B-53

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 on page B-56]. For a list and description of the equations, see "M-Phase Control Equations" on page B-25.

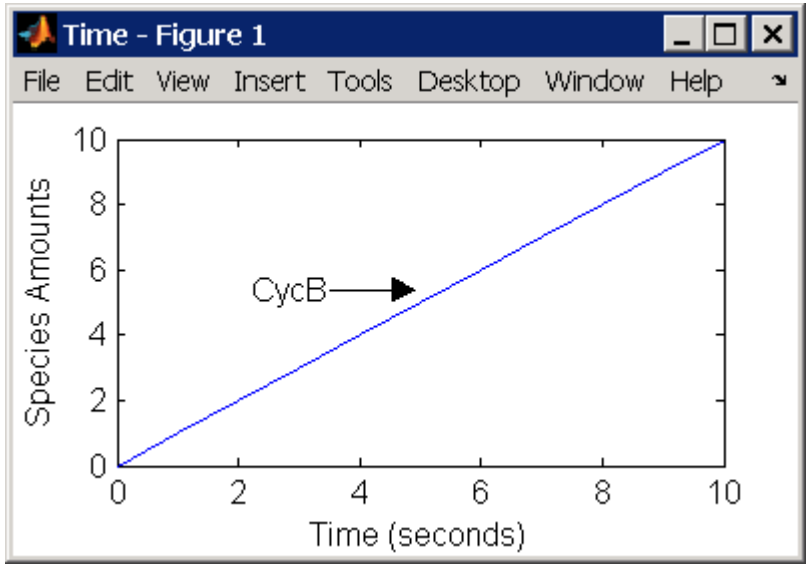
A basic model includes reactions 1 to 8, 11 to 13, 15 to 17, 19, and 36 to 43 with algebraic rules from equations 17, 18, and 19.

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 `sundials` solver.



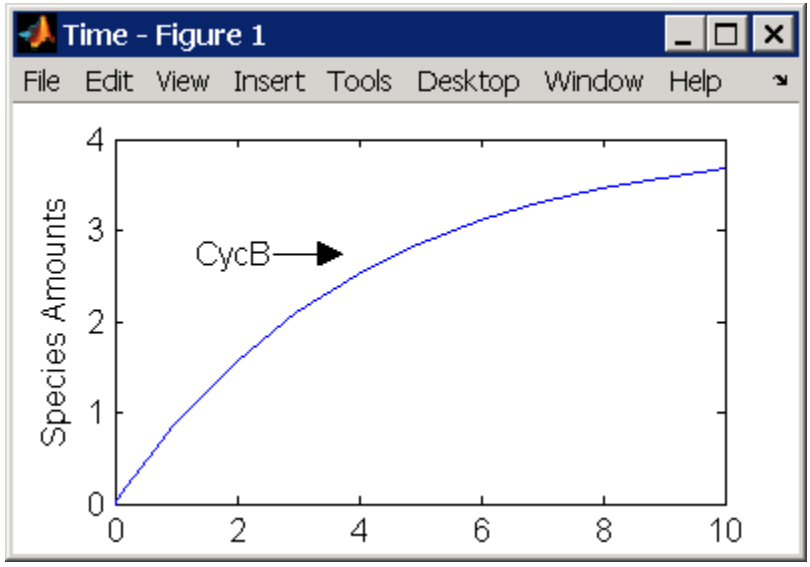
Reaction 2, Degradation of Cyclin B

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`.

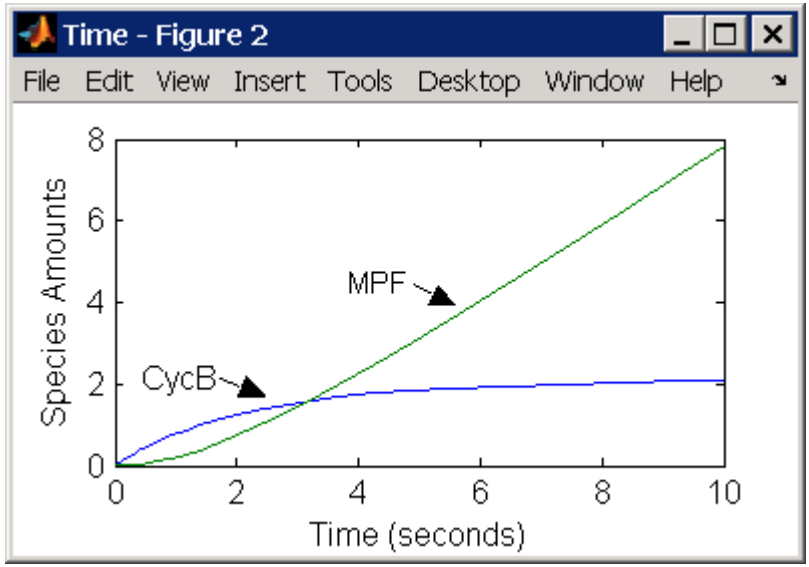


Reaction 3, Dimerization of Cyclin B with Cdc2 Kinase

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`.

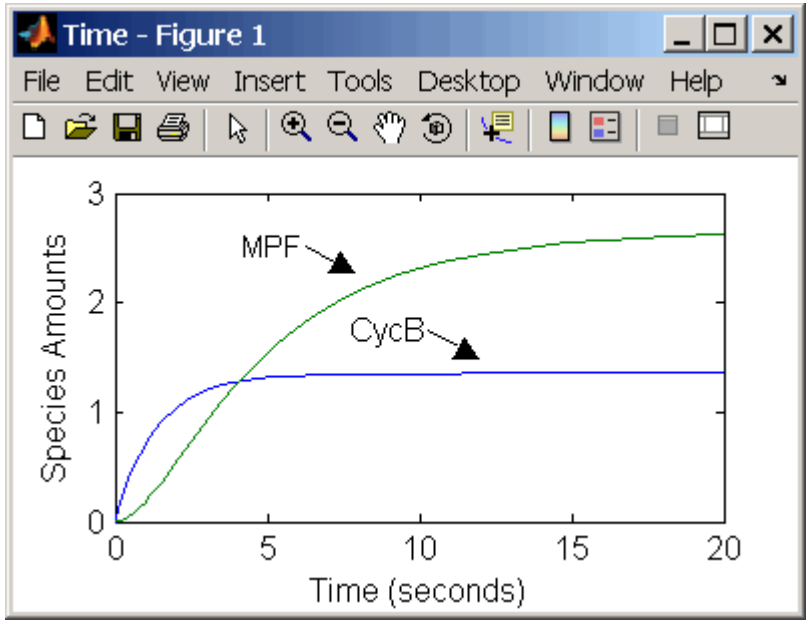


Reaction 4, Degradation of Cyclin B on MPF

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.



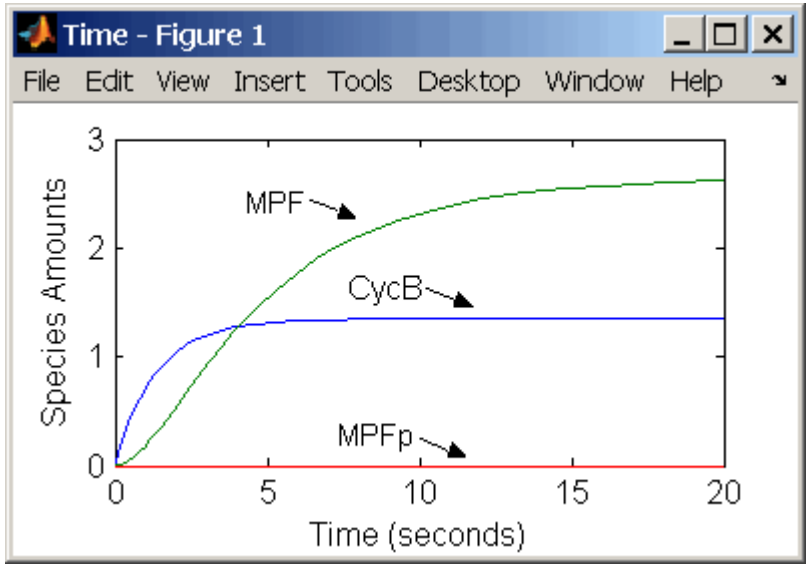
Reaction 5, Deactivation of Active MPF

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.



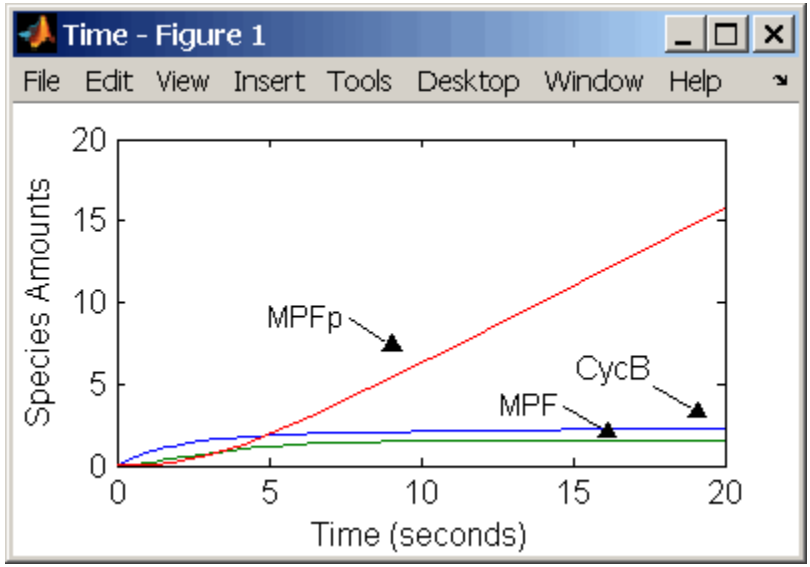
Reaction 6, Activation of MPF

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.

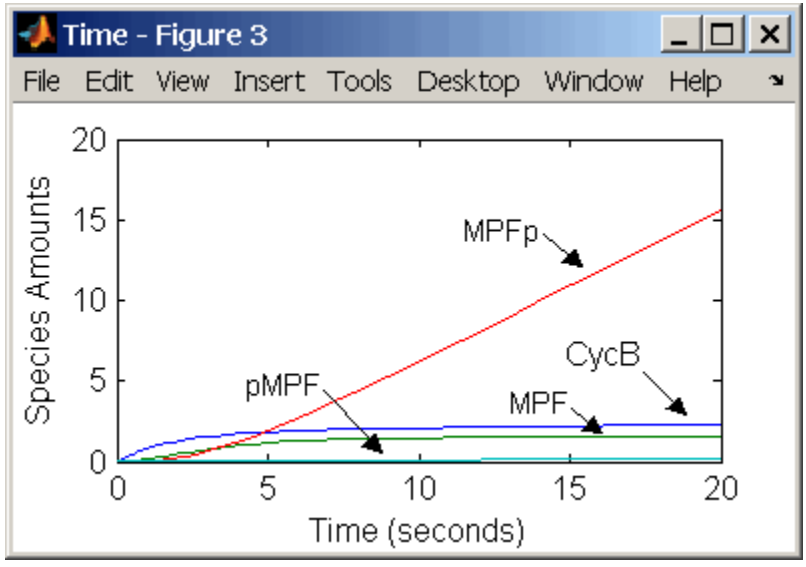
Reaction 8, Inhibition of MPF by Phosphorylation

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.

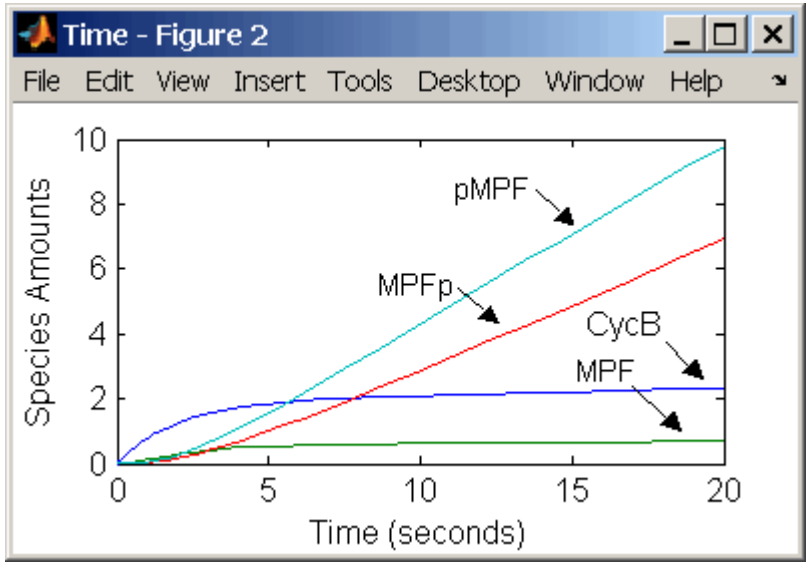
```
     reaction: MPF -> pMPF
reaction rate: Kwee1*[MPF]
   parameters: Kwee1 = 0.0 1/minute or 0.01 1/minute, variable by
                                                    algebraic rule
               Vwee1i = 0.01 1/(nM*minute)
               Vwee1a = 1.0 1/(nM*minute)
      species: MPF = 0 nM
               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 on page B-44.

```
     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 on page B-45.

```
     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).

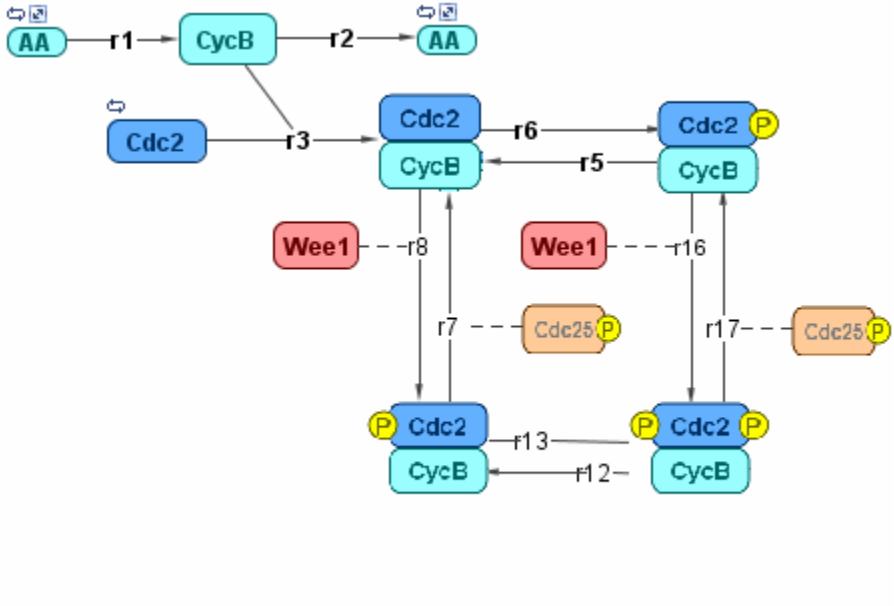
```
     reaction: APCi + IEp -> APCa + IEp
reaction rate: (kap*[IEp]*[APCi])/(Kmap + [APCi])
   parameters: kap = 0.13 1/minute
               Kmap = 0.01 nM
     species : APCi = 1 nM
               APCa = 0 nM
```

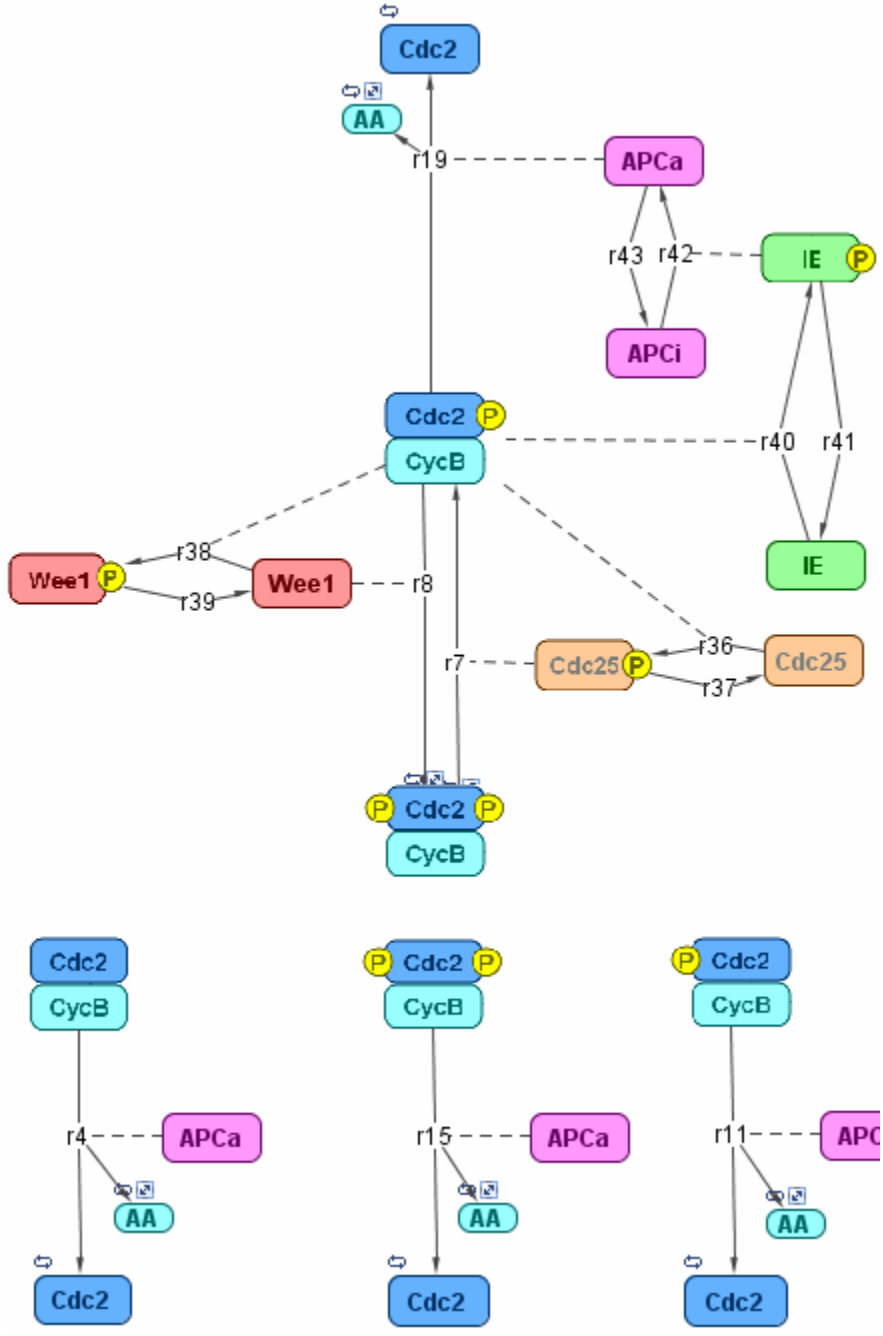
Reaction 43, APC Deactivation

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


AA
r1
CycB
r2
AA
Cdc2
r3
Cdc2
CycB
r6
Cdc2
P
CycB
r5
Wee1
r8
Wee1
r16
r7
Cdc25
P
r17
Cdc25
P
P
Cdc2
r13
P
Cdc2
P
CycB
r12
CycB


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

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.