SimBiology

For Use with MATLAB®

- Computation
- Visualization
- Programming

User's Guide

Version 1

How to Contact The MathWorks:

`www.mathworks.com`	Web
`comp.soft-sys.matlab`	Newsgroup

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

508-647-7000 Phone

508-647-7001 Fax

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

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

SimBiology User's Guide

Revision History

September 2005 Online only New for Version 1.0 (Release 14SP3+)

Contents

1 Modeling

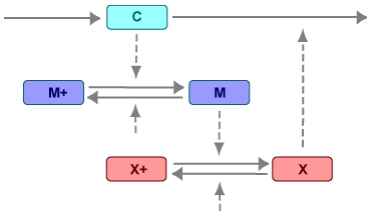
Simulation

2

Index

Modeling

This chapter describes how you can use SimBiology to model biological processes. It begins with the familiar concepts of mass action and enzyme kinetics.



Mass Action Kinetics

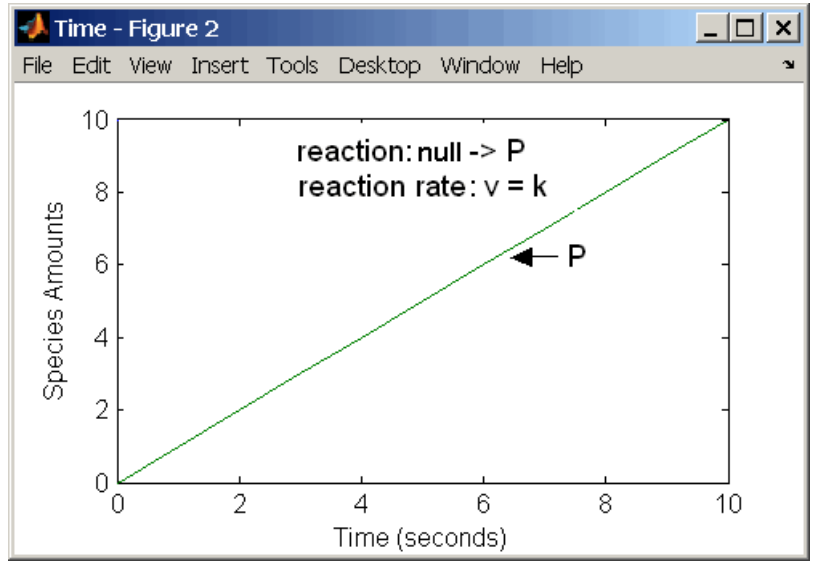
Mass action describes the behavior of reactants and products in an elementary chemical reaction. Mass action kinetics describes this behavior as an equation where the velocity or rate of a chemical reaction is directly proportional to the concentration of the reactants.

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/(liter*second)
      species: R = 10 mole
               P =  0 mole
   parameters: k =  1 mole/(liter*second)
```

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



Zero-Order Mass Action Kinetics

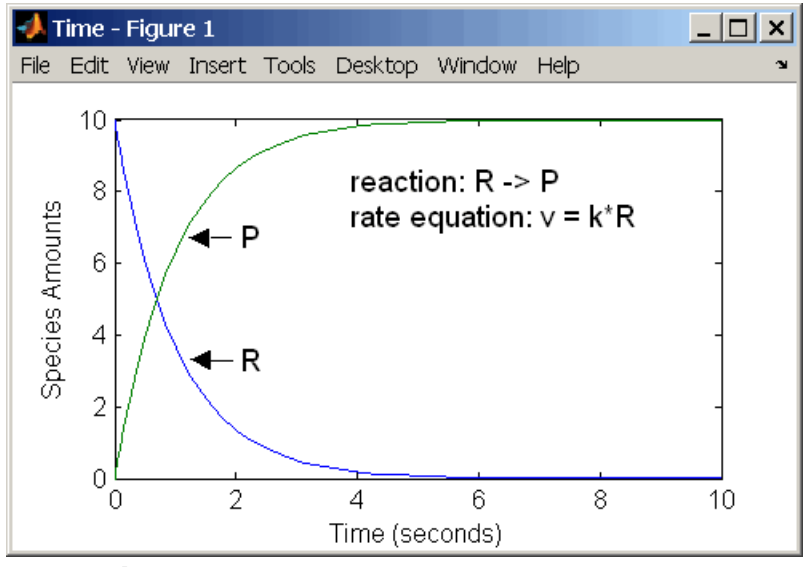
Note If the amount of a reactant with zero-order kinetics reaches zero before the end of a simulation, then the amount of reactant can go below zero regardless of the solver or tolerances you set.

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 SimBiology and simulating produces the following results:



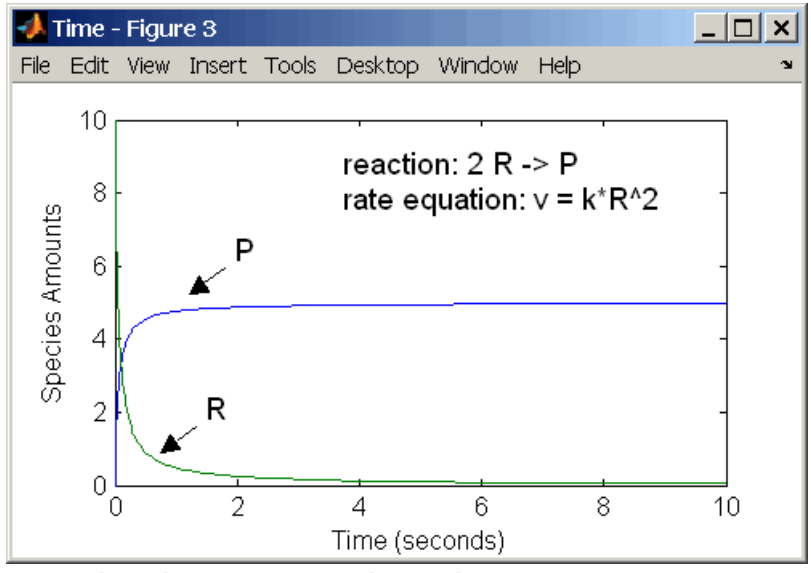
First-Order Mass Action Kinetics

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 SimBiology and simulating produces the following results:



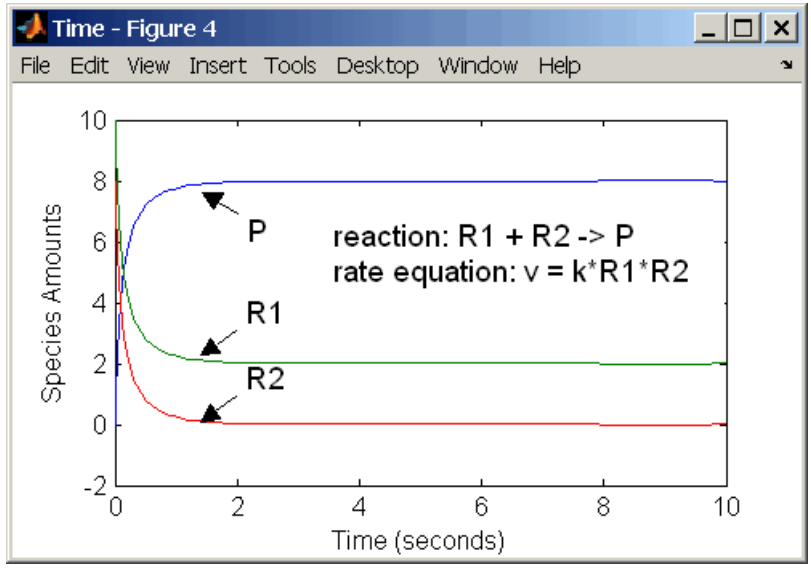
Second-Order Kinetics with Single Reactant

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 SimBiology 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.



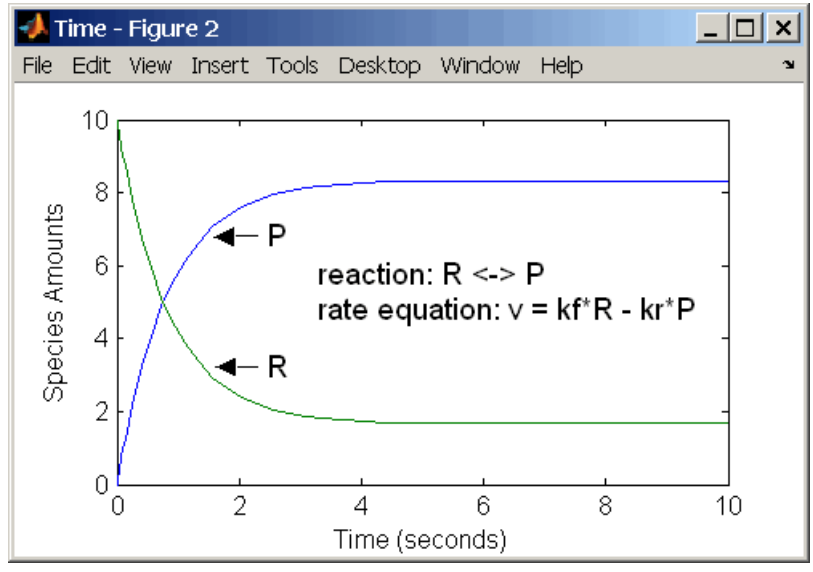
Second-Order Kinetics with Two Reactants

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 SimBiology and simulating produces the following results. At equilibrium when the rate of the forward reaction equals the revers reaction, `v = kf*R - kr*P = 0` and `P/R = kf/kr`.



Enzyme Kinetics

Enzymes can increase the rate of a reaction by using a reaction mechanism or pathway with a lower activation energy. This section describes a common single substrate enzyme model using a mass action mechanism or rate equations derived from mass action mechanisms.

Simple Model for Single Substrate Catalyzed Reactions (p. 1-8)	Model for a single substrate reaction catalyzed irreversibly by an enzyme
Enzyme Reactions with Differential Rate Equations (p. 1-9)	Model reactions with differential rate equations derived from the reactions and reaction rates
Enzyme Reactions with Mass Action Kinetics (p. 1-11)	Model reactions directly with their reaction rate equations
Enzyme Reactions with Irreversible Henri-Michaelis-Menten Kinetics (p. 1-12)	Model a mass action mechanism with a derived kinetic equation

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 1-9.
- Reactions with mass action kinetics, see “Enzyme Reactions with Mass Action Kinetics” on page 1-11.

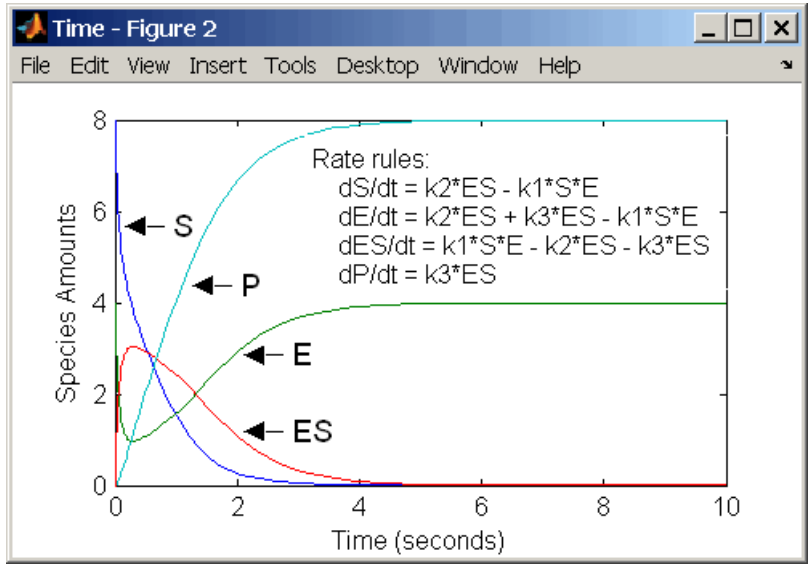
- Reactions with Henri-Michaelis-Menten kinetics, see “Enzyme Reactions with Irreversible Henri-Michaelis-Menten Kinetics” on page 1-12.

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 1-8) can be described with differential rate equations. You can enter the differential rate equations into SimBiology 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)
                 k2r = 1   1/second
                 k2 = 1.5 1/second
```

Remember to enter rate rules using the form `dS/dt = f(x)` as `S = f(x)`.



Alternatively, you could remove the rate rule for `ES`, add a new species `Et` 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
albebric rule: 0 = Et - E - ES
      species: S =  8   mole
               E =  4   mole
              ES =  0   mole
               P =  0   mole
              Et =  4   mole
   parameters: k1 = 2   1/(mole*second)
               k1r = 1   1/second
               k2 = 1.5 1/second
```

Enzyme Reactions with Mass Action Kinetics

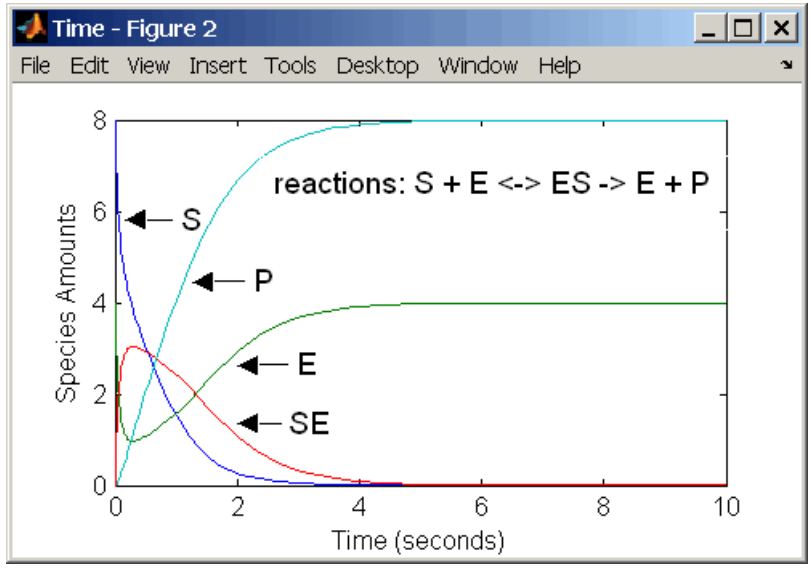
Determining the differential rate equations for the reactions in a model is a time-consuming process. A better way is to enter the reactions for a single substrate enzyme reaction mechanism directly into SimBiology. The following example using models an enzyme catalyzed reaction with mass action kinetics. For a description of the reaction model, see "Simple Model for Single Substrate Catalyzed Reactions" on page 1-8.

```
     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 1-8.

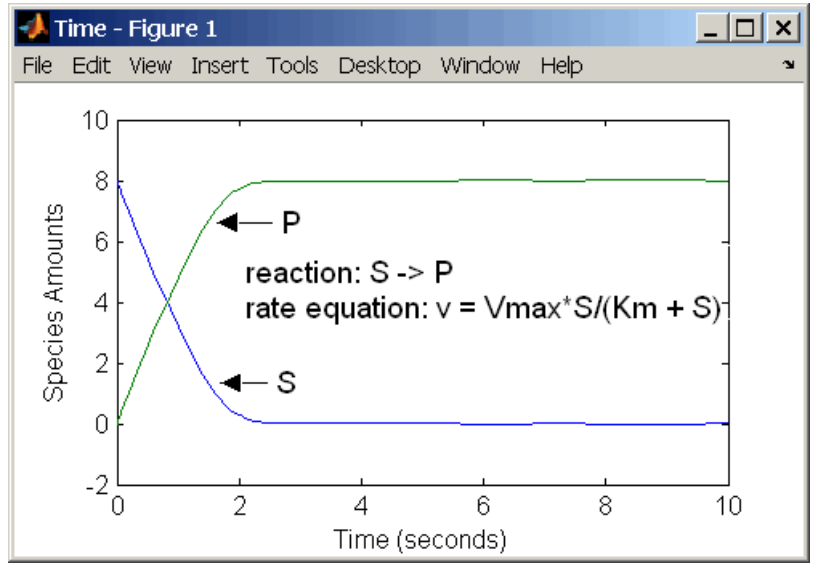
$$v = \frac{Vmax[S]}{Km + [S]}$$

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 SimBiology 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.



Constant Amounts and Boundary Conditions

SimBiology has two properties (constant amount, boundary condition) to specify how the amount of a species changes or does not change during a simulation

Definition of Constant and Boundary Properties (p. 1-14)	Definitions
Constant = NO, Boundary = NO (p. 1-15)	Species modeled in a reaction or a rule, but not both.
Constant = YES, Boundary = NO (p. 1-15)	Constant species that are neither modeled in a reaction or varied by a rule.
Constant = NO, Boundary = YES (p. 1-16)	Species in a reaction, but changed only by a rule
Constant = YES, Boundary = YES (p. 1-17)	Constant species in reactions that adds mass (sources) or removes mass (sinks)
Model Edges (p. 1-18)	Interface between biological system (model) and the environment

Definition of Constant and Boundary Properties

The SBML specification (Level 2, Version 1) added the property BoundaryCondition to the model definition.

Species with BoundaryCondition = Yes — The species amount is either constant or determined by a rule, but in either case the amount is 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.

Species with ConstantAmount = No — The species amount is determined by a reaction or a rule, but not both.

Species with ConstantAmount = Yes — The species amount does not change during a simulation. The species can be in a reaction or rule, but it cannot have a rule that changes its amount.

Constant = NO, Boundary = NO

The value of a species can change, and it can change with either a reaction or rule, but not both.

Constant	Boundary	Reaction	Rule	Changed By
NO	NO	YES	NO	Reaction
NO	NO	NO	YES	Rule

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

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

Constant = YES, Boundary = NO

The value of a species cannot change. When a species has its ConstantValue selected and BoundaryCondition not selected, it acts like a parameter. It cannot be in a reaction and it cannot be varied by a rule.

Constant	Boundary	Reaction	Rule	Changed By
YES	NO	NO	NO	Never

Example — Species **E** is not in the reaction, but it is in 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)
```

Constant = NO, Boundary = YES

The value of a species can change, and it is in a reaction, but a differential rate term from the reaction is not created. The value of the species change with a rule and a differential rate term is created from the rule.

Constant	Boundary	Reaction	Rule	Changed By
NO	YES	YES	YES	Rule

From the SMBL specification (Level 2, Version 1), "By default, when a species is a product or reactant of one or more reactions, its concentration is determined by those reactions. In SBML, it is possible to indicate that a given species' concentration is not determined by the set of reactions even when that species occurs as a product or reactant; i.e., the species is on the boundary of the reaction system but is a component of the rest of the model."

Example — 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 — 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)
```

Constant = YES, Boundary = YES

The value of the species can change. It is in a reaction, but a differential rate term is not created from the reaction. The differential rate term is created from a rule.

Constant	Boundary	Reaction	Rule	Changed By
YES	YES	YES	NO	Never

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

Example — **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 — 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 — The **null** species in SimBiology 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 — **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)
```

Model Edges

As you build complex models from simpler pathways, there are edges in the model that you need to define before simulating the model. Knowing where the model edges are located is important because a species that is initially constant or unregulated can later vary as you add details to your model. The concept of a model edge overlaps with SBML boundaries, but not always.

Model edge — Species with constant amounts that might or might not be modeled in the reaction and reaction rate equations. Examples are cofactors, NAD^+, ATP, and DNA.

Model edge — Enzymes with constant amounts that are not regulated. For example, a Michaelis-Menten rate equation with `Vmax` specified as a parameter assumes that the amount of enzyme catalyzing the reaction remains constant.

$$v = \frac{Vmax*[Substrate]}{Km + [Substrate]}$$

You may want to temporarily model a regulated enzyme in a rate equation. If the amount of enzyme is constant, then this species is a model edge. After adding the reaction(s) that change the amount of the enzyme,

$$v = \frac{k*[\mathbf{Enzyme}]*[Substrate]}{Km + [Substrate]}$$

Model edge — Null or source species that synthesizes another species at a constant rate (zero order reaction). Mass is added to the system.

Model edge — degradation of a species to a null or sink species (first-order reaction). Mass is taken away from the system.

Algebraic Rules

An algebraic rule is a model component that defines the value for a nonconstant parameter or the amount of a species that is determined through a algebraic equation instead of a differential relationship.

What Is an Algebraic Rule?

An algebraic rule is an equation that defines the value of a variable that you may not be able to define with a reaction. Use algebraic rules for defining equity constraints that are not rates of change.

There are two types of rules that are evaluated at each time step during a simulation. The first is a rate rule (see "Rate Rules" on page 1-21) while the second is an algebraic rule. An algebraic rule is defined by the equation

```
0 = f(W) - x
```

The variable `x` can be a species amount or parameter value. The function `f(W)` is an expression that can include other species and parameters. In SimBiology, you enter an algebraic rule using the form

```
f(W) - x
```

Mass Balance Equations

There are some models in the literature that are defined with differential rate equations and algebraic mass balance equations.

A mass balance equation can define the amount of a species and reduce the number of differential rate equations that need to be solved. For example, a common signal transduction pathway can include a reaction `Ei -> Ea` where an enzyme transforms from an active form to an inactive form and back. The amount of inactive enzyme `Ei` is defined by the differential rate equation

`dEi/dt = Vm*Ei/Km + Ei`. If the total amount of the enzyme is known or remains constant, the total amount of enzyme `Ea` can be defined with the algebraic equation `Ea = Et - Ei` instead of a differential equation.

With SimBiology, models are defined by reactions, and the corresponding differential rate equations are calculated for all species. Adding a mass balance equation as an algebraic rule, and setting `Et` to be constant, would overdefine the model and cause a simulation error (the number of equations cannot be greater then the number of independent variables). If want to use a mass balance equation, you have to let `Et` vary, then `Et` is an independent variable that is not defined by a reaction and the simulation works.

Rate Rules

A rule is an model component that defines the value for a parameter or the amount of a species. Use algebraic rules for equations that are not rates of change. Use rate rules for equations that determine the rate of change for a parameter value or species amount.

What Is a Rate Rule? (p. 1-21)	Define the rate of change for a species amount or parameter value.
Rate of Change Is Constant (p. 1-22)	Rate of change does not depend on the changing amount of a species
Rate of Change Is Exponential (p. 1-23)	Rate of change depends on the changing amount of the species
Rate of Change Is Determined by Another Species (p. 1-24)	Rate of changed depends on the changing amount of another species
Differential Rate Equations as Rules (p. 1-25)	Rate of change is defined with a differential rate equation derived from the reactions

What Is a Rate Rule?

A rule is an equation that defines the value for a variable. For species, use rate rules as an alternative to the differential rate expression generated from reactions.

There are two types of rules that are evaluated at each time step during a simulation. The first is an algebraic rule (see “Algebraic Rules” on page 1-19) while the second is a rate rule. A rate rule is defined by the equation

```
dx/dt = f(W)
```

The variable `x` can be a species amount, parameter value, or compartment dimension (volume or area). The function `f(W)` is an expression that can include other species and parameters. In SimBiology, you enter a rate rule using the form

```
x = f(W)
```

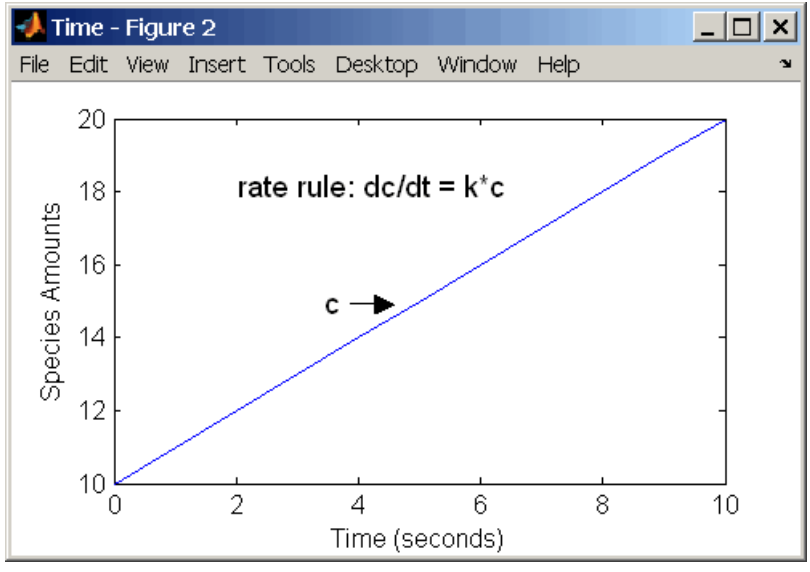
Rate of Change Is Constant

You can increase or decrease the amount or concentration of a species by a constant value using a zero order rule. For example, the species `c` increases by a constant rate `k`. You could also include species and parameters that have their `ConstantAmount` or `ConstantValue` properties selected.

```
      reaction: none
 rate equation: none
     rate rule: dc/dt = k
       species: c = 0 mole
    parameters: k = 1 mole/second
```

The solution is $c = kt + c_0$ where c_0 is the initial amount or concentration of the species `c`.

Enter the rule described above as `c = k`. From the **RuleType** list, select `rate`, enter the values for `c` and `k`, and then simulate.



Alternatively, you could model a constant increase in a species with the reaction `null -> C`.

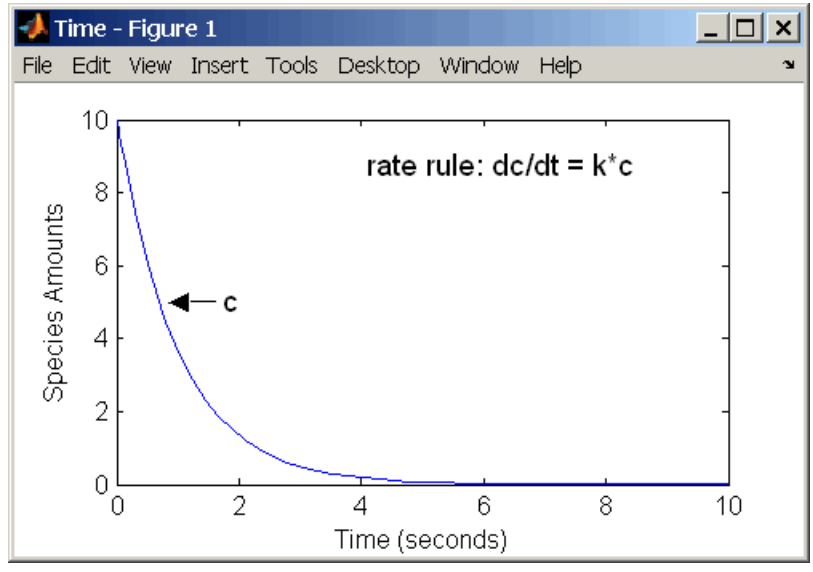
Rate of Change Is Exponential

You can change the amount of a species similar to a first-order reaction using a first-order rate rule. For example, the species `c` decays exponentially. You could also include a parameter with its `ConstantValue` property deselected.

```
      reaction: none
rate equation: none
    rate rule: dc/dt = -k*c
      species: c = 10 mole
   parameters: k =  1 1/second
```

The solution for the rate rule `dc/dt = -k*c` is $c = c_0 e^{-kt}$.

Enter the rate rule described above and simulate with an ODE solver.



Notice that if the amount of a species `c` is determined by a rate rule and `c` is also in a reaction, `c` must have its property for `BoundaryCondition` selected. For example, with a reaction `a -> c` and a rate rule `dc/dt = k*c`, select the `BoundaryCondtion` for `c` so that a differential rate term is not created from the reaction. The amount of `c` is determined solely by a differential rate term from the rate rule.

If the boundary condition is not selected, you will get the following error message:

```
Invalid rule variable 'in a reaction or another rule'.
```

Rate of Change Is Determined by Another Species

A species from one reaction can determine the rate of another reaction if it is in the second reaction rate equation. In a similar way, a species from a reaction can determine the rate of another species if it is in the rate rule that defines that other species.

```
      reaction: a -> b
rate equation: v = -k1*a
    rate rule: dc/dt = k2*a
      species: a = 10 mole
               b =  0 mole
               c =  5 mole
   parameters: k1 = 1 1/second
               k2 = 1 1/second
```

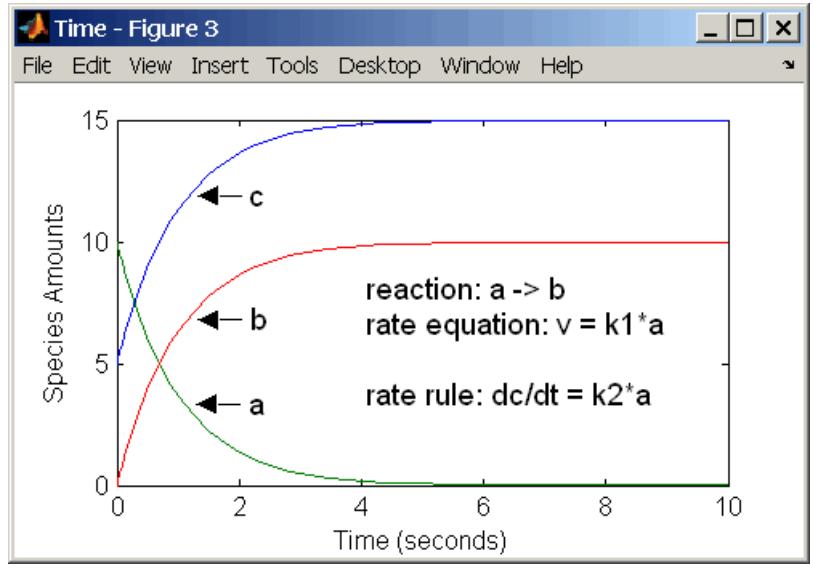
The solution for the species in the reaction are

$$a=a_0e^{-k1t} \quad \text{and} \quad b=a_0(1-e^{-k1t})$$

With the rate rule `dc/dt = k2*a` dependent on the reaction, $dc/dt = k_2(a_0e^{-k1t})$, and the solution is

$$c = c_o + k_2a_o/k_1(1 - e^{-k_1t})$$

Enter the reaction and rule described above and simulate.



Differential Rate Equations as Rules

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`,

$$\frac{dC}{dt} = v_i - v_d X \frac{C}{K_c + C} - k_d C$$

as a rate rule in SimBiology:

```
C = vi - (vd*X*C)/(Kc + C) - kd*C
```

2

Simulation

Simulation Overview

How do solvers work

In order to simulate a model, the model is converted to a set of differential equations. The solver functions are used to compute solutions for those equations at different time intervals, giving the model's states and outputs over a span of time. You can then plot these outputs from your simulation.

The MATLAB ODE solvers are designed to handle *ordinary differential equations*. An ordinary differential equation contains one or more derivatives of a dependent variable y with respect to a single independent variable t, usually referred to as time.

The solver functions implement numerical integration methods for solving initial value problems for ordinary differential equations (ODEs). Beginning at the initial time with initial conditions, they step through the time interval, computing a solution at each time step. If the solution for a time step satisfies the solver's error tolerance criteria, it is a successful step. Otherwise, it is a failed attempt; the solver shrinks the step size and tries again.

Stiff Versus Nonstiff Models

An ordinary differential equation problem is stiff if the solution being sought is varying slowly, but there are nearby solutions that vary rapidly, so the numerical method must take small steps to obtain satisfactory results. The

ODE solvers in MATLAB whose name ends in "s" are for "stiff" problems. Many biological models are numerically stiff because they include species amounts that are changing quickly and others that change slowly.

Stiffness is an efficiency issue. If you don't care how much time a computation takes, you need not be concerned about stiffness. Nonstiff methods can solve stiff problems; they just take a long time to do it.

As an illustration, imagine trying to find the quickest descent through a canyon. An explicit algorithm, which is normally used for nonstiff models, would sample the local gradient to find the descent direction. But following the gradient on either side of the trail will send you bouncing back and forth from wall to wall — the descent will be found but it will take a long time. An implicit algorithm used for stiff models can anticipate where each step is taking you, keep you on the trail with fewer steps, and so save time. Using a stiff solver for a stiff problem can save thousands of solver steps and function evaluations compared to a non-stiff solver.

Methods intended to solve stiff problems efficiently do more work per step, but can take much bigger steps. Stiff methods are implicit. At each step they use MATLAB matrix operations to solve a system of simultaneous linear equations that helps predict the evolution of the solution.

Not all difficult problems are stiff, but all stiff problems are difficult for solvers not specifically designed for them. Solvers for stiff problems can be used exactly like the other solvers.

For an illustrative code example you can run to plot the effects of numerical stiffness on different solvers, see MATLAB News & Notes - May 2003 Cleve's Corner: Stiff Differential Equations..

Selecting a Solver

Choice of solver depends on the problem and time available for computation. There are trade-offs to be made between speed and accuracy. In general, ode45 is the best function to apply as a "first try" for most problems, or ode15s if you suspect that a problem is stiff. As you find out more about the problem you can try other solvers. Experimentation is generally required to determine the best solver for a particular model. As a general guide:

1. Models with either all fast or all slow changing variables are nonstiff problems:

   Use “Nonstiff Deterministic Solvers” on page 2-5

   - ode45 — best first guess
   - ode23 — may be more efficient than ode45 with crude tolerances and mild stiffness
   - ode113 — may be more efficient than ode45 with stringent tolerances
   - ode15i — for fully implicit differential equations

2. Models with both fast and slow changing variables are stiff problems:

   Use “Stiff Deterministic Solvers” on page 2-6

   - ode15s — try first if you suspect that a problem is stiff, or if ode45 failed or was very inefficient
   - ode23s — may be more efficient than ode15s at crude tolerances, and can solve some stiff problems that ode15s cannot
   - ode23t — use this solver if the problem is only moderately stiff and you need a solution without numerical damping
   - ode23tb — Like ode23s, this solver may be more efficient than ode15s at crude tolerances

3. Models with a small number of molecules:

   Use “Stochastic Solvers” on page 2-7

   - Stochastic — Most accurate, may be too slow if the initial number of molecules for a reactant species is large.
   - Explicit Tau — speeds up the simulation at the cost of some accuracy; can be orders of magnitude faster than Stochastic. Can be used for large problems (provided the problem is not numerically stiff.)
   - Implicit Tau — May be the fastest, at the cost of some accuracy. Can be used for large problems and also for numerically stiff problems. For non-stiff systems may not be a good choice because it adds computational overhead.

Nonstiff Deterministic Solvers

Models with either all fast or all slow changing variables may not be numerically stiff. Nonstiff Deterministic Solvers are appropriate to try.

ode45 (Dormand-Prince)

Based on an explicit Runge-Kutta (4,5) formula, the Dormand-Prince pair. It is a one-step solver - in computing $y(t_n)$. It needs only the solution at the immediately preceding time point $y(t_{n-1})$. In general, ode45 is the best function to apply as a "first try" for most problems.

ode23 (Bogacki-Shampine)

Based on an explicit Runge-Kutta (2,3) pair of Bogacki and Shampine. It may be more efficient than ode45 at crude tolerances and in the presence of mild stiffness. Like ode45, ode23 is a one-step solver

ode113 (Adams)

Variable order Adams-Bashforth-Moulton PECE solver. It may be more efficient than ode45 at stringent tolerances and when the ODE function is particularly expensive to evaluate. ode113 is a multistep solver - it normally needs the solutions at several preceding time points to compute the current solution.

Stiff Deterministic Solvers

Models with fast and slow changing variables are numerically stiff. Stiff Deterministic Solvers are the best choice.

ode15s (stiff/NDF)

Variable-order solver based on the numerical differentiation formulas (NDFs). Optionally it uses the backward differentiation formulas, BDFs, (also known as Gear's method). Like ode113, ode15s is a multistep solver. If you suspect that a problem is stiff or if ode45 failed or was very inefficient, try ode15s.

ode23s (stiff/Mod. Rosenbrock)

Based on a modified Rosenbrock formula of order 2. Because it is a one-step solver, it may be more efficient than ode15s at crude tolerances. It can solve some kinds of stiff problems for which ode15s is not effective.

ode23t (Mode. stiff/Trapezoidal)

An implementation of the trapezoidal rule using a "free" interpolant. Use this solver if the problem is only moderately stiff and you need a solution without numerical damping.

ode23tb (stiff/TR-BDF2)

An implementation of TR-BDF2, an implicit Runge-Kutta formula with a first stage that is a trapezoidal rule step and a second stage that is a backward differentiation formula of order 2. Like ode23s, this solver may be more efficient than ode15s at crude tolerances.

Stochastic Solvers

Models with a small number of molecules can realistically be simulated stochastically — that is, allowing the results to contain an element of probability, unlike a deterministic solution. The stochastic simulation algorithms provide a practical method for simulating reactions which are stochastic systems. Expect stochastic simulations to take more computation time than deterministic simulations.

Stochastic Simulation Algorithm (SSA)

Using the stochastic simulation algorithm for a system is equivalent to solving the Chemical Master Equation for the system. The Chemical Master Equation is otherwise impossible to solve for most practical problems. Thus, the stochastic simulation algorithm provides a practical method for simulating stochastic systems. The algorithm simulates one reaction at a time based on the propensity function for each reaction.

Advantages:

- This algorithm is exact

Disadvantages:

- Since it evaluates one reaction at a time, it may be too slow for large problems
- If the number of molecules of any of the reactants is huge, it may take a long time to complete the simulation

Explicit Tau-Leaping Algorithm

Since the stochastic simulation algorithm may be too slow for a lot of practical problems, this algorithm has been designed to speed up the simulation at the cost of some accuracy. The algorithm treats each reaction channel as being independent of the others. It automatically chooses a time interval such that the relative change in the propensity function for each reaction is less than the user specified error tolerance. After selecting the time interval, the algorithm computes the number of times each reaction channel fires during the time interval and makes the appropriate changes to the concentration of various chemical species involved.

Advantages:

- This algorithm can be orders of magnitude faster than the SSA
- It can be used for large problems (provided the problem is not numerically stiff.)

Disadvantages:

- Some accuracy is sacrificed for speed.
- Not good for stiff models.
- The error tolerance needs to be specified in such a manner that the resulting time steps are of the order of the fastest time scale.

Implicit Tau-Leaping Algorithm

Like the explicit tau-leaping algorithm, the implicit tau-leaping algorithm is also an approximate method of simulation designed to speed-up the simulation at the cost of some accuracy. It can handle numerically stiff problems better than the explicit tau-leaping algorithm. For deterministic systems, a problem is said to be numerically stiff if there are “fast” and “slow” time scales present in the system and the “fast modes” are stable. For such problems, the explicit tau-leaping method performs well only if it continues to take small time steps that are of the order of the fastest time scale. The implicit tau-leaping method can potentially take much larger steps and still be stable. The algorithm treats each reaction channel as being independent of others. It automatically chooses a time interval such that the relative change in the propensity function for each reaction is less than the user specified error tolerance. After selecting , the algorithm computes the number

of times each reaction channel fires during the time interval and makes the appropriate changes to the concentration of various chemical species involved.

Advantages

- This algorithm can be much faster than the SSA. It is also usually faster than the explicit-tau leaping algorithm
- It can be used for large problems and also for numerically stiff problems.
- The total number of steps taken is usually less than the explicit-tau leaping algorithm.

Disadvantages

- Some accuracy is sacrificed for speed.
- There is a higher computational burden for each step as compared to the explicit-tau leaping algorithm. This leads to a larger CPU time per step.
- This method often damps out the perturbations off the slow manifold leading to a reduced state variance about the mean.

References

[1] Gibson M.A, Bruck J. (2000), "Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels," Journal of Physical Chemistry, 105:1876-1899.

[2] Gillespie D (1977), "Exact Stochastic Simulation of Coupled Chemical Reactions," The Journal of Physical Chemistry, 81(25): 2340-2361.

[3] Gillespie D (2000), "The Chemical Langevin Equation," Journal of Chemical Physics, 113(1): 297-306.

[4] Gillespie D (2001), "Approximate Accelerated Stochastic Simulation of Chemically Reacting Systems," Journal of Chemical Physics,115(4):1716-1733.

[5] Gillespie D, Petzold L. (2004), "Improved Leap-Size Selection for Accelerated Stochastic Simulation," Journal of Chemical Physics, 119:8229-8234

[6] Rathinam M, Petzold L, Cao Y, Gillespie D (2003), "Stiffness in Stochastic Chemically Reacting Systems: The Implicit Tau-Leaping Method," Journal of Chemical Physics, 119(24):12784-12794.

Index