VITO VOLTERRA
(1860–1940)

Vito Volterra was born on May 3, 1860 in Ancona, Italy (then part of the Papal States) and died in 1940. The wars of the Risorgimento forced him to be a refugee several times as a child. Volterra received his doctoral degree in physics in 1882 at the University of Pisa, for work on hydrodynamics. At age 23 he was full professor of Mechanics at Pisa, and later in his career would be Professor of Mechanics and Chair of Mathematical Physics in Rome. Volterra's most important work was in mathematics—particularly the theory of integration and differential equations. In 1883, Volterra invented *functionals*—central to the Density Functional Theory developed by Walter Kohn (Chapter 7). Volterra's subsequent work on differential equations would be essential to the theory of optics, electromagnetism, elasticity, and x-ray diffraction. Volterra was an enthusiastic supporter of Italy's engagement in the First World War. Despite being 55 years old, he enlisted in the Italian Air Force, researching dirigibles (he proposed using helium as a lifting gas instead of flammable hydrogen) and submarines. After the war, Volterra collaborated with his son-in-law, Umberto D'Ancona, a marine biologist, to model the unusual fluctuations in species abundance when the commercial fisheries were interrupted by the war. Unknown to them, Alfred Lotka proposed similar differential equations for autocatalytic chemical reactions in 1910. Volterra was strongly opposed to the Italian Fascists, and was one of only 12 (out of 1250) professors who reused to take the Fascist loyalty oath in 1931. As a result, he was forced to leave the University of Rome and resign from all Italian Scientific Societies. In a 1930 postcard criticizing the Fascists, Volterra wrote: *Empires die, but Euclid's theorems keep their youth forever.*

CHAPTER 15

Kinetics

Chemical kinetics is the study of reaction rates—that is, the time evolution of a collection of interacting and interconverting compounds. Aside from pure chemistry, many biochemical, geological, physiological, environmental, engineering, and materials science problems involve understanding the time behavior of complicated reaction networks.

In a typical introductory chemistry class, chemical kinetic reaction rates are described using sets of differential equations for production and consumption of species. The variables in these differential equations represent the concentration of each species as functions of time. Section 15.1 introduces numerical methods for treating arbitrarily complicated sets of differential equations. However, the use of differential equations is an implicit mathematical assumption that concentration is a real number and not an integer, and thus can change continuously. This is an excellent approximation when there are many molecules per unit volume. But when the number of molecules per unit volume becomes small—for example, inside a cell or a nanosystem—this assumption corresponds to an unphysical division of discrete particles into fractional pieces. Interestingly, the discreteness of the reactants leads to very different dynamics. Section 15.2 introduces a *kinetic* Monte Carlo algorithm capable of simulating these types of systems.

Strictly speaking, rate constants and initial conditions should have consistent units of concentration and time. The units for rate constants can become unwieldy. Moreover, since we care only about the general behavior here, the particular units of time (seconds, hours, years) and concentration (μmol/L,mol/L) are unimportant. Therefore, for brevity, I will omit the units in the expressions for the rate constants and concentrations in the examples discussed in this chapter. However, you should beware of this potential problem when solving "real-life" problems. A common error, especially when using multiple reactions, is failure to use consistent units. Be sure that the units of concentration and time of the rate constants are the same, and are also the same as the units of concentration and time used for the initial conditions and time integration. Consider yourself warned.

15.1 DETERMINISTIC KINETICS

The typical introductory chemistry treatment of chemical kinetics using differential equations relies on two unstated assumptions. First, as discussed above, it assumes that the system

consists of a sufficiently large number of particles, such that the concentration behaves as a real number described deterministically by a differential equation. This assumption fails when the number of particles per volume is small. We'll discuss this case (and its computational solution) in Section 15.2. Second, it assumes that the system can be described as a well-stirred flask—i.e., that a single "concentration" describes the entire vessel, and the rate of the reaction is not limited by the diffusion of the reactants. This assumption fails when the reagents are inhomogeneously distributed and the time needed for the reactants to get close enough to each other to react is the rate-limiting step. We'll discuss this case in Section 15.3.

However, for now, let us assume that these two assumptions are valid. As you learned in introductory chemistry, one solves these types of kinetic problems by writing out a set of differential equations describing the reactions, and then deriving an analytical solution. For cases involving only a single reaction type, this can often be done exactly. For cases with a small number of reactions, it is often possible to introduce further assumptions, such as the existence of steady-state behavior, to obtain an approximate analytical solution. However, many interesting processes involve interacting reactions, whose corresponding coupled differential equations are too complicated to solve analytically. An alternative strategy—which is applicable to arbitrarily large and complex sets of reactions—is to *numerically* integrate the set of differential equations. The numerical approach described below is used for simulations in fields as diverse as atmospheric chemistry, chemical engineering, combustion modeling, and biochemistry.

15.1.1 First-order reactions

For a first-order decay reaction,

$$\mathrm{A} \xrightarrow{k_1} \mathrm{B}, \tag{15.1}$$

the concentration of species A as a function of time obeys the differential equation

$$\frac{\mathrm{d}\,[\mathrm{A}]}{\mathrm{d}t} = -k_1[\mathrm{A}], \tag{15.2}$$

having the solution

$$[\mathrm{A}(t)] = [\mathrm{A}(0)]e^{-k_1 t}. \tag{15.3}$$

The differential equation in Eq. (15.2) can either be solved symbolically or numerically. The symbolic solution—Eq. (15.3)—can be found using the `DSolve[]` Mathematica function. `DSolve[]` takes three arguments consisting of a list of the differential equations and boundary conditions, the function(s) to be solved, and its dependent variable(s). As an example, for Eq. (15.2), considering $k_1 = 0.1$ and $[\mathrm{A}(t = 0)] = 1$ (in the appropriate unit system of concentration and time units),

```
k1=0.1; (*define rate constant*)
soln0=DSolve[ {a'[t] == -k1*a[t], a[0]== 1}, a,t]
```

$$\{\{\texttt{a} \to \texttt{Function}[\{\texttt{t}\},\ 1.\texttt{e}^{-0.1\texttt{t}}]\}\}$$

To avoid the possibility of inadvertently using the same names as built-in Mathematica functions, we will use lowercase variable names to denote the concentrations of the species throughout this chapter. In this example, there is only one differential equation (Eq. (15.2)) and one initial condition (the initial concentration of `a` at time `0` is `1`). This set of equations specifies the constraints that must be satisfied by the solution. The list of these equations is provided as the first argument to `DSolve[]`. The `'` symbol (single quote or tick mark) denotes the first derivative of the function with respect to its (only) variable—alternatively, you could use the `D[]` function to take derivatives with respect to time. Since `DSolve[]` attempts to satisfy an equality constraint, the `==` operator is used instead of the `=` operator. The second argument is the function to be solved for (`a`), and the third argument is the dependent variable to be solved for (`t`).

The output describes the solution of `a`, as a `Function[]` of `t`; this is in exact agreement with the solution in Eq. (15.3). To get a specific numerical value, one `Evaluate[]`s the solution for a particular value of the dependent variables. For example, at time $t = 10$,

```
Evaluate[ a[10]/.soln0 ] (*analytical solution at t=10*)
```

```
{0.367879}
```

Analytical solutions can often be found for simple differential equations, but practical problems with many reacting species often do not have analytical solutions. However, numerical solutions—implemented using `NDSolve[]`—are applicable to arbitrarily complex sets of coupled differential equations (i.e., many different types of chemical reactions occurring simultaneously, and having each other as products and/or reactants). Whereas `DSolve[]` returns a mathematical function giving the concentrations at *any* time, `NDSolve[]` performs a numerical integration only for a limited interval. Thus, the main input difference between `NDSolve[]` and `DSolve[]` is that an explicit time interval over which to obtain a solution must be specified. The desired time interval (here from $t = 0$ to $t = 10$)—rather than just the name of the variable—is specified as the third argument:

```
soln1=NDSolve[ {a'[t] ==-k1*a[t], a[0]== 1}, a, t,0,10}][[1]]
```

```
{a → InterpolatingFunction[ Domain: {{0., 10.}}
                            Output: scalar ]}
```

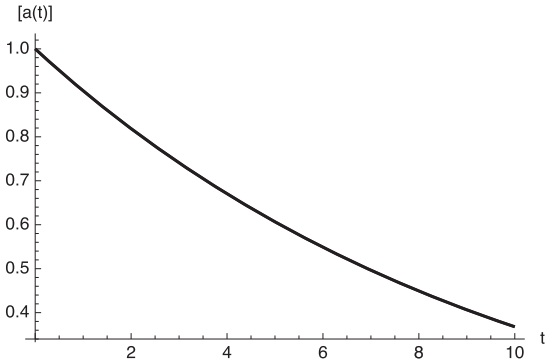
The `NDSolve[]` output is a more complicated output of nested lists. The additional `[[1]]` extracts the first part of the list, which will simplify later numerical calculations. Instead of returning a simple mathematical function, `NDSolve[]` returns an `InterpolatingFunction[]` representing the numerical solutions. The details of this function are not made explicit, but the output shows some basic characteristics, such as the valid domain of inputs (here $0 \rightarrow 10$) and the output type (here a scalar value). Clicking on the plus sign shows an expanded description of the polynomial type used, the order of the polynomial, and whether or not the function is periodic. Although this `InterpolatingFunction[]` does not have a simple form, it can be used like any other Mathematica function. For example, the `InterpolatingFunction[]` can be evaluated using the `Evaluate[]` funtion, as in the symbolic solution example

```
Evaluate[ a[10]/.soln1 ] (*numerical solution at t=10*)
```

```
0.367879
```

Note the perfect agreement between the analytical and numerical solutions at the (arbitrarily chosen) time of $t = 10$. The `InterpolatingFunction[]` can also be plotted as a function of time using the `Plot[]` function:

```
Plot[ Evaluate[a[t]/.soln1], {t,0,10},
  PlotStyle→{Black}, AxesLabel→{"t","[a(t)]"}]
```



15.1.2 Reversible first-order reactions

A reversible first-order reaction,

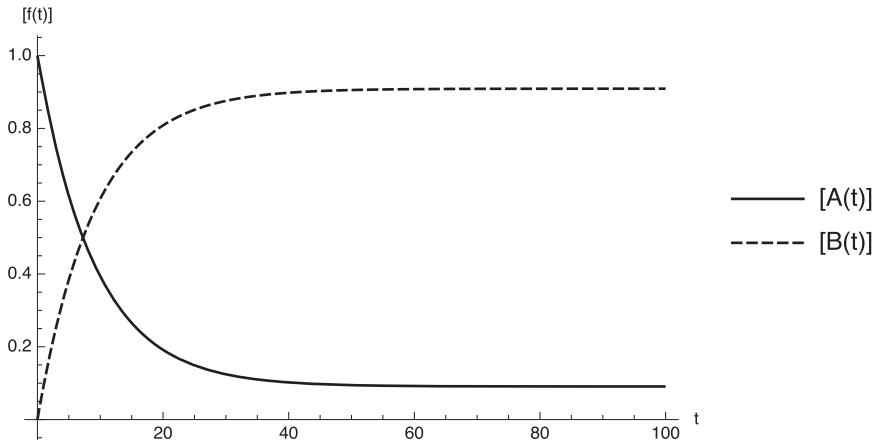
$$\mathrm{A} \underset{k_2}{\overset{k_1}{\rightleftharpoons}} \mathrm{B}, \tag{15.4}$$

can be written as the differential equations

$$\begin{aligned} \frac{\mathrm{d}\,[\mathrm{A}]}{\mathrm{d}t} &= -k_1[\mathrm{A}] + k_2[\mathrm{B}] \\ \frac{\mathrm{d}\,[\mathrm{B}]}{\mathrm{d}t} &= -k_2[\mathrm{B}] + k_1[\mathrm{A}]. \end{aligned} \tag{15.5}$$

Let's solve this problem numerically using `NDSolve[]` for the specific example of $k_1 = 0.1$, $k_2 = 0.01$, $[\mathrm{A}(t = 0)] = 1$, $[\mathrm{B}(t = 0)] = 0$:

```
k1=0.1; (*define rate constants*)
k2=0.01;
soln2=NDSolve[
  {a'[t]==-k1*a[t]+k2*b[t], b'[t]==+k1*a[t]-k2*b[t], a[0]==1, b[0]==0},
  {a,b},{t,0,100}][[1]];

Plot[ {Evaluate[a[t]/.soln2],Evaluate[b[t]/.soln2]}, {t,0,100},
  PlotLegends→{"[A(t)]","[B(t)]"},
  PlotStyle→{Black,Dashed},AxesLabel→{"t","[f(t)]"}]
```



As expected from detailed balance, at sufficiently long times the system reaches a dynamic equilibrium when the forward and reverse rates are equal. Taking the ratio of the rate constants for the forward and reverse reactions yields an analytical solution for the equilibrium ratio of the two species.

15.1.3 Parallel first-order reactions

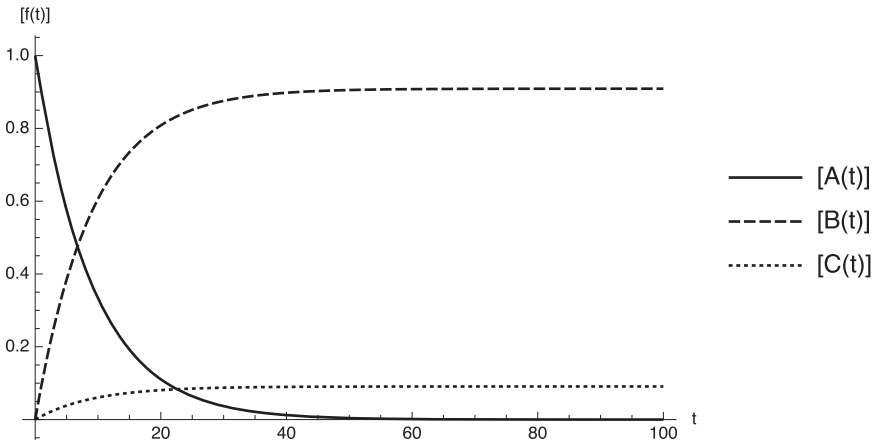
Another simple example is a pair of parallel first-order reactions:

$$\begin{aligned} \mathrm{A} &\xrightarrow{k_1} \mathrm{B} \\ \mathrm{A} &\xrightarrow{k_2} \mathrm{C}. \end{aligned} \tag{15.6}$$

The general strategy to solve this problem is exactly the same as the previous examples: (i) express the reactions as a set of differential equations; (ii) express the differential equations and their boundary Uonditions as a list of Mathematica functions; and (iii) use `NDSolve[]` to obtain a numerical solution for each of the functions.[1] Considering the specific example of $k_1 = 0.1$, $k_2 = 0.01$, $[\mathrm{A}(t=0)] = 1$, $[\mathrm{B}(t=0)] = 0$, $[\mathrm{C}(t=0)] = 0$, the numerical solution is

```
k1=0.1; (*define rate constants*)
k2=0.01;
soln3=NDSolve[
  {a'[t]== -k1*a[t]-k2*a[t],
   b'[t]== +k1*a[t],
   c'[t]== +k2*a[t],
   a[0]==1, b[0]==0, c[0]==0},
  {a,b,c},{t,0,100}][[1]];

Plot[ {Evaluate[a[t]/.soln3],Evaluate[b[t]/.soln3],Evaluate[c[t]/.soln3]},
  {t,0,100}, PlotLegends→{"[A(t)]","[B(t)]","[C(t)]"},
  PlotStyle→{Black,Dashed,Dotted},AxesLabel→{"t","[f(t)]"}]
```

1. This particular set of equations has an analytical solution, so one could also use `DSolve[]`.



15.1.4 Oscillatory reactions: The Lotka-Volterra reaction

Consider the following set of reactions:[2]

$$\begin{aligned} \mathrm{S} + \mathrm{X} &\xrightarrow{k_1} 2\mathrm{X} \\ \mathrm{X} + \mathrm{Y} &\xrightarrow{k_2} 2\mathrm{Y} \\ \mathrm{Y} &\xrightarrow{k_3} \mathrm{P}. \end{aligned} \tag{15.7}$$

Species S is a time-independent "source," and species P is a "product" whose ultimate fate doesn't concern us—therefore, we will only plot the time dependence of species X and Y. Both X and Y participate in autocatalytic reactions, where the presence of the species (along with another substrate) generates an additional unit of the species. The interaction of these autocatalytic reactions is interesting: When [Y] is low, [X] increases. The increasing [X] then enables increasing [Y]. But this higher [Y] depletes X more rapidly, slowing the production of new Y species. When the production rate of new Y is lower than the degradation rate to P, [Y] is depleted. Consequently, this slows down the elimination of X, allowing [X] to increase again, just as we had at the beginning. In other words, the interactions between X and Y result in periodic oscillations of [X] and [Y], provided the reaction is "fed" with S.

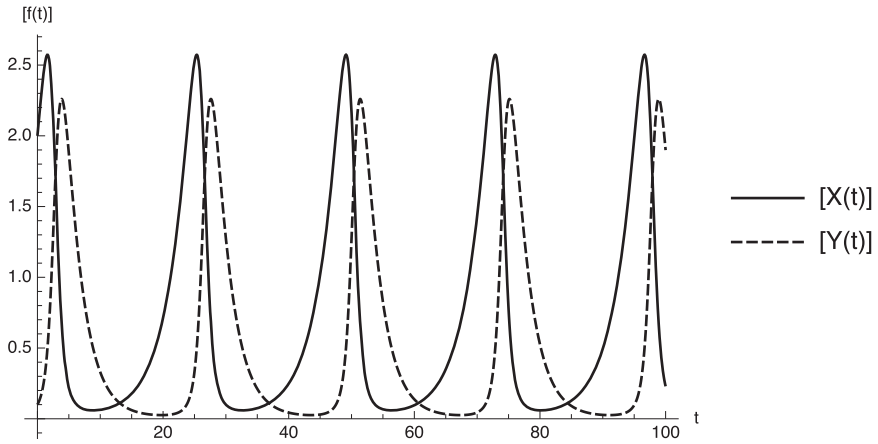
Let's compute a numerical solution to Eq. (15.7), using rate constants of $k_1 = 0.3$, $k_2 = 0.6$, $k_3 = 0.4$, $[\mathrm{S}] = 1.0$, $[\mathrm{X}(0)] = 2.0$, and $[\mathrm{Y}(0)] = 0.1$. The initial concentration of P is irrelevant, and is arbitrarily set to zero.

2. You can think of these equations as modeling an ecosystem with "predator" and "prey" species. X is a prey species that feeds on an undepletable food source S, and reproduces (making more X offspring) via the first reaction. Y is a predator species that feeds on X and reproduces (making more Y offspring) via the second reaction. The predators, Y, die of old age (via the third reaction). The biology of this model is a bit strange, because the species reproduce asexually after eating.

```
k1=0.3; k2=0.6; k3=0.4; s=1.0; (*define rate constants*)
soln4=NDSolve[
  {x'[t]==+k1*s*x[t]-k2*x[t]*y[t],
   y'[t]==+k2*x[t]*y[t]-k3*y[t],
   p'[t]==+k3*y[t],
   x[0]==2.0,
   y[0]==0.1,
   p[0]==0 },
  {x,y,p}, {t,0,1000}][[1]];
```

A `Plot[]` of the results shows that the species concentrations oscillate periodically, as surmised above.[3]

```
Plot[ {Evaluate[x[t]/.soln4], Evaluate[y[t]/.soln4]}, {t,0,100},
   PlotLegends→{"[X(t)]","[Y(t)]"},
   PlotStyle→{Black,Dashed},AxesLabel→{"t","[f(t)]"}]
```

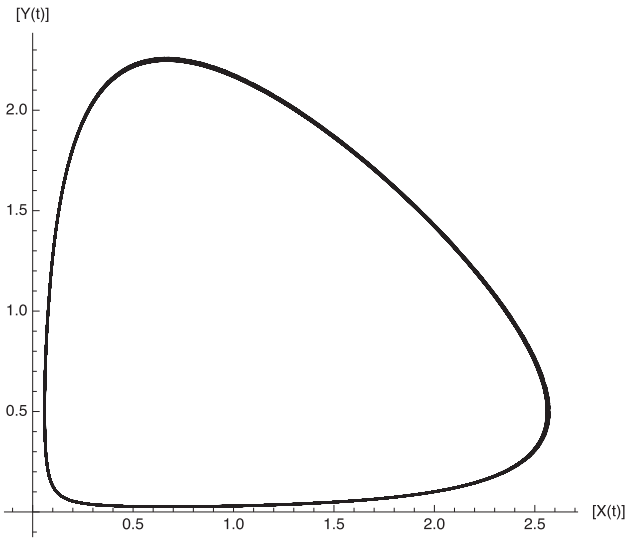


Both the qualitative analysis of Eq. (15.7) and this `Plot[]` indicate that [X(t)] and [Y(t)] are synchronized, with the rise in [X] preceding the rise in [Y].

An alternative way to visualize this synchronization is a *phase plane diagram*, in which the time evolution is projected into the [X],[Y]-plane. The `NDSolve[]` results for [X(t)] and [Y(t)] can be plotted together, for each time, using a `ParametricPlot[]`:

3. For the experts: Take a discrete Fourier transform of the concentration with respect to time, using the `Fourier[]` function, to obtain the oscillation frequency.

```
ParametricPlot[
  Evaluate[{x[t],y[t]}/.soln4], {t,0,1000},
  AxesLabel→{"[X(t)]","[Y(t)]"} ]
```



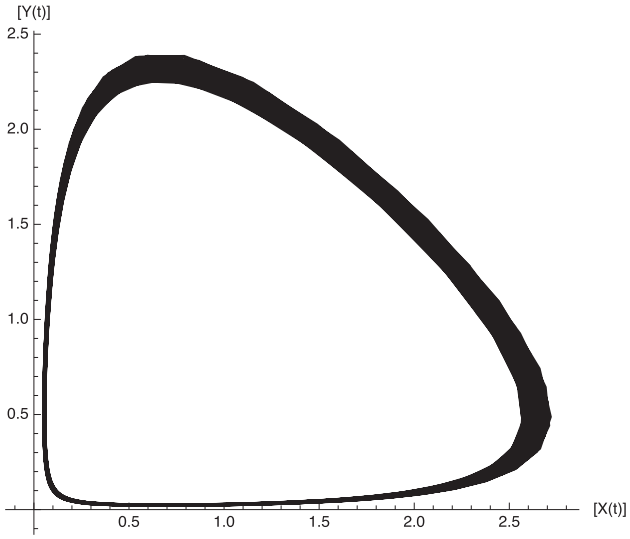
This phase plane diagram reveals two properties of the Lotka-Volterra reaction system. First, the closed loop indicates that the cycle repeats itself. Second, the single line of the trajectory indicates that there is no variation from cycle to cycle. In other words, the system returns to the same {[X(t)], [Y(t)]} state indefinitely.

One way to perturb this regular oscillation is to introduce noise into the evolution of species X by adding a small random term to the expression for d[X]/dt:

```
k1=0.3; k2=0.6; k3=0.4; s=1.0; (*define rate constants*)
soln5=NDSolve[
  {x'[t]==+k1*s*x[t]-k2*x[t]*y[t]+10^-4*RandomReal[{-1,+1}],
   y'[t]==+k2*x[t]*y[t]-k3*y[t],
   p'[t]==+k3*y[t],x[0]==2.0,
   y[0]==0.1,p[0]==0},
  {x,y,p}, {t,0,1000}][[1]];
```

After adding the noise term, the trajectories slightly differ at each cycle, resulting in a band of lines in the phase plane diagram, whose width indicates the cycle-to-cycle concentration deviation magnitude.

```
ParametricPlot[
  Evaluate[{x[t],y[t]}/.soln5], {t,0,1000},
  AxesLabel→{"[X(t)]","[Y(t)]"}]
```



Nonetheless, the cycle still forms a closed loop, indicating that the oscillation repeats over time and is therefore robust to random perturbations of this magnitude.

15.2 STOCHASTIC KINETICS

Modeling the concentrations as real numbers is an approximation—chemical systems are actually composed of discrete atomic and molecular components. Therefore, it is fundamentally more correct to treat the molecule numbers as integers, with each microscopic reaction adding and subtracting individual molecules according to the stoichiometry of the elementary reactions. As the reactant concentration becomes small, the difference between the discrete and continuous treatment becomes more pronounced.

In principle, chemical reactions are deterministic, provided the initial conditions are known and the system evolves either classically using Newton's laws (as in Chapter 14) or quantum mechanically using the time-dependent Schrödinger equation (as in Section 1.6). In practice, directly following the conversion of reactants into products in this way quickly becomes too complicated. Adopting a stochastic approach to kinetics makes the problem more tractable. This is analogous to the statistical treatment of thermodynamics, where we replaced the deterministic evolution (e.g., of particles in a gas) with expressions for their probabilistic characteristics. Like the Metropolis Monte Carlo introduced in Chapter 11, the *kinetic Monte Carlo* method provides a systemic way of performing state-to-state transitions. For kinetic problems, the "states" describe the quantities of all the different reactants and products. In Metropolis Monte Carlo, the state-to-state transitions occur according to the relative Boltzmann probabilities. To incorporate time into the kinetic Monte Carlo calculation, the state-to-state transition probabilities are determined by the kinetic equations. "Faster" reactions occur more frequently per unit time, and thus are more likely to occur during a

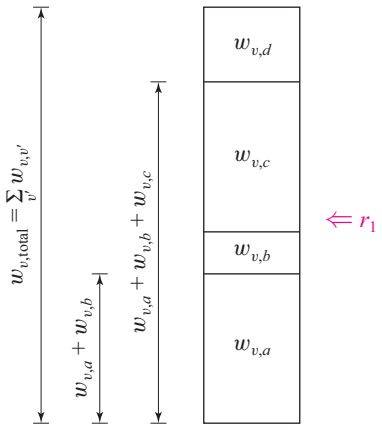


Figure 15.1. Selecting a new state of the system from the exit rates.

particular time interval. After describing the details of the kinetic Monte Carlo algorithm, we'll revisit the kinetic models from the previous section to demonstrate the method and illustrate differences between stochastic and deterministic kinetic behavior.

15.2.1 Kinetic Monte Carlo algorithm

Here we will consider the theoretical grounds and algorithmic description of the kinetic Monte Carlo procedure. If this seems too abstract, you may skim this discussion, implement the examples in Sections 15.2.2 and 15.2.3 to see how the simulation works in practice, and then come back to this section for a more thorough reading.

Suppose that a collection of particles at time t is in a particular state v with the probability $p_v(t)$. The state label v describes the numbers of each type of particle in the system (e.g., 100 A and 20 B). The numbers of each type of particle can change over time, so at a later time, $t + \mathrm{d}t$, the system exists in some state, v', with probability

$$p_{v'}(t + \mathrm{d}t) = \sum_v w_{v,v'} p_v(t)\, \mathrm{d}t, \tag{15.8}$$

where $w_{v,v'}$ is the rate at which the system goes from state v to v' within the time interval of $\mathrm{d}t$. Accordingly, since $w_{vv'}$ is a rate, it has units of time^{-1}. "Fast" reactions have a large $w_{v,v'}$, and "slow" reactions have a small $w_{v,v'}$. Note how this equation is similar to Eq. (11.4) from the discussion on detailed balance and the Metropolis Monte Carlo algorithm in Section 11.2.2. In both cases, v and v' denote different possible states of the system. There, the "state" of the system described the particular quantum numbers and other variables of the system; here, the "state" indicates the numbers of each of the chemical species. There, the transitions between states modeled the relative probabilities of two states; here, the transitions model the occurrence rates of chemical reactions. There, the simulation steps (Monte Carlo moves) had no relationship to time; here, the primary interest is time evolution of the numbers of particles.

To perform the time evolution correctly and obey mass conservation, each of the $w_{v,v'}$ must be consistent with the elementary rate equations when the numbers of particles go from v to v'. Each particular transition $v \to v'$ competes against all the other ways that the system can go to any possible outcome state v''. The total rate of exiting the state v to any of these possible outcomes, $v \to v''$, is the sum of all the possible exit process rates,

$$w_{v,\text{total}} = \sum_{v''} w_{v,v''}. \tag{15.9}$$

In turn, the *probability* that the particular process $v \to v'$ occurs is given by

$$P(v \to v') = \frac{w_{v,v'}}{w_{v,\text{total}}}, \tag{15.10}$$

where the denominator, $w_{v,\text{total}}$, normalizes the probability of $v \to v'$.

How does one select which reaction occurs? Think of all the possible ways of exiting the system as forming a one-dimensional dartboard of height $w_{v,\text{total}}$, which is divided into segments of heights $w_{v,v'}$ (shown in Figure 15.1). Reactions are selected by throwing a dart randomly at this dartboard to select which process $v \to v'$ to perform. Slow reactions correspond to a small $w_{v,v'}$, so they are less likely to be selected than fast reactions. This captures the common-sense reasoning that a fast reaction (with large $w_{v,v'}$) is more likely to occur than a slow reaction (with small $w_{v,v'}$) in a given time interval $\mathrm{d}t$. This "dart-throwing" algorithm can be implemented by choosing a uniformly distributed random number, r_1, in the interval $(0, w_{v,\text{total}})$, and using the value to select a reaction. The relative ordering of the possible reactions is irrelevant.

ED/AU: We changed sublinear "v, *total*" to "v, total" per the global change instruction in chap03 related to sublinear "total". Correct to do it in this construction throughout chap15?

How much time elapses per reaction? This can also be modeled as a stochastic process. Within a certain interval of time, either some reaction occurs or no reactions occur. The probability that "some event" occurs in a time interval of $\mathrm{d}t$ is $w_{v,\text{total}}\,\mathrm{d}t$. Since the total probability of "some event" and "no event" is normalized, the probability that "no event" occurs in that time interval $\mathrm{d}t$ is $1 - w_{v,\text{total}}\,\mathrm{d}t$. Suppose that $\bar{P}(t)$ is the probability that "no event" occurs between time 0 and time t. The probability that no event occurs during both the time interval $0 \to t$ and the time interval $t \to t + \mathrm{d}t$ is the product of these two probabilities:[4]

$$\bar{P}(t + \mathrm{d}t) = \bar{P}(t)\left[1 - w_{v,\text{total}}\,\mathrm{d}t\right]. \tag{15.11}$$

Since a reaction requires a finite amount of time to occur, if no time elapses then no reaction occurs, so it follows that $\bar{P}(t = 0) = 1$. Rearranging Eq. (15.11) into a differential equation for $\bar{P}(t)$ as $\mathrm{d}t$ approaches the limit of 0 yields

$$\frac{\mathrm{d}\bar{P}(t)}{\mathrm{d}t} = -w_{v,\text{total}}\bar{P}(t), \tag{15.12}$$

4. This is analogous to flipping two coins. If p_1 is the probability of getting heads on the first coin ($0 \to t$ interval) and p_2 is the probability of getting heads on the second coin ($t \to t + \mathrm{d}t$ interval), then the probability of both occurring is $p_1 p_2$.

which is mathematically the same first-order decay reaction differential equation solved in Eq. (15.3). Using the boundary condition $\bar{P}(t=0)=1$ from above, then

$$\bar{P}(t) = \exp\left[-w_{v,\text{total}}t\right]. \tag{15.13}$$

Remember that $\bar{P}(t)$ is the probability that nothing happens; the probability that something happens is thus

$$1-\bar{P}(t) = 1-\exp\left[-w_{v,\text{total}}t\right], \tag{15.14}$$

and the probability that it happens at time t is found by taking the derivative with respect to t,

$$\frac{\mathrm{d}\left(1-\bar{P}(t)\right)}{\mathrm{d}t} = w_{v,\text{total}}\exp\left[-w_{v,\text{total}}t\right]. \tag{15.15}$$

The right-hand side implies an exponentially decreasing probability distribution of reaction times t. The simplest way to sample this distribution is by working backward from the reaction probability. The probability that the reaction occurred by time t—described by the left-hand side of Eq. (15.15)—is sampled by choosing a uniformly distributed random number, $r_2 \in (0, 1)$. The corresponding random value of how long it took—the elapsed reaction time t—is found by substituting r_2 as the left-hand side of Eq. (15.15) and solving for t:

$$t = \frac{1}{w_{v,\text{total}}}\ln\left(\frac{1}{r_2}\right) = -\frac{1}{w_{v,\text{total}}}\ln\left(r_2\right). \tag{15.16}$$

The resulting t is used to update the total elapsed time of the simulation at each step. This process can be implemented as the function

```
drawTime[wtotal_]:=(-1/wtotal)*Log[RandomReal[]];
```

In summary, the kinetic Monte Carlo algorithm consists of the following steps:

1. Start at $t = 0$ and specify the state of the system, v, by the initial numbers of each type of species.
2. For the current state of the system, v, compute all the possible reaction rates, $w_{v,v'}$—corresponding to each of the elementary reaction mechanisms—and use these to compute the total exit rate, $w_{v,\text{total}} = \sum_{v'} w_{v,v'}$. Note that $w_{v,v'}$ and $w_{v,\text{total}}$ change over time, and must be recomputed after every reaction event.
3. Choose a uniformly distributed random number, r_1, between 0 and $w_{v,\text{total}}$, and then use it to select the new state of the system, v' (i.e., which reaction occurs), following the "1D dartboard" illustrated in Figure 15.1.
4. Update the number of each species according to the stoichiometry of the $v \to v'$ reaction selected in the previous step.
5. Advance the total simulated time, t, by $-\ln(r_2)/w_{v,\text{total}}$, by choosing a uniformly distributed random number r_2 between 0 and 1. (This can be done by calling the `drawTime[]` function defined above.)
6. End the simulation if a stopping criterion has been met (e.g., no more reactants, maximum total time has elapsed); otherwise return to step 2.

15.2.2 First-order reactions

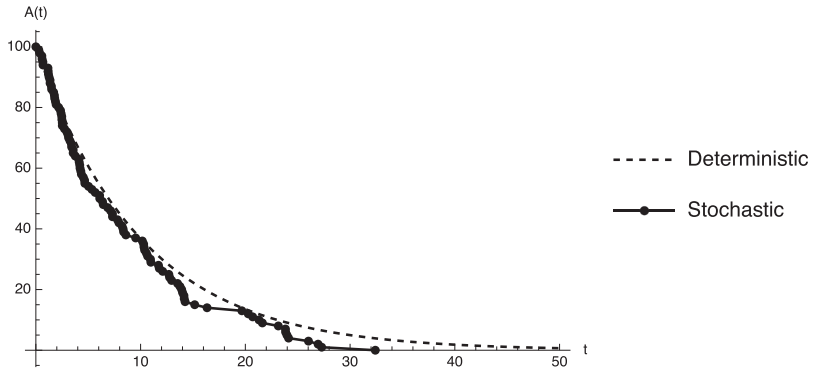
Let's implement a stochastic version of the first-order decay reaction previously discussed in Section 15.1.1. The mechanism is the same as Eq. (15.1); the only difference here is that the units are numbers of particles and not concentration. For concreteness, consider the example of $k_1 = 0.1$ particles/time and the initial number of A particles at 100:

```
k1=0.1;                          (*define rate constants*)
nA=100;                          (*number of starting particles*)
currentTime=0;                   (*initial time*)
trajectory={{currentTime,nA}}; (*state of the system with time*)

While[(nA>0),  (*run simulation while reactant A remains*)
  rate1=k1*nA; (*compute the reaction rates; here only one is possible*)
  rateTotal=rate1;

  nA-=1;       (*pick a reaction to occur; here only one is possible*)

  currentTime+=drawTime[rateTotal];      (*update time*)

  AppendTo[trajectory,{currentTime,nA}]; (*update trajectory*)
];
```

Let's step through each line of this example. The rate constant, `k1`, number of A particles, `nA`, and `currentTime` are defined and initialized. The `trajectory` list stores the number of particles as a function of time, in the form of time-number pairs. The simulation runs `While[]` there are still species A left to react (i.e., `nA > 0`). Since Eq. (15.1) is the only option, there is no need to randomly choose which reaction occurs. Likewise, the total rate, `rateTotal`, is simply `rate1`, since this is the only possible reaction. The time is advanced by calling the `drawTime[]` function defined above. Appending the `currentTime` and number of particles, `nA`, to the `trajectory` list maintains a record of the time course of reaction.

Let's compare the deterministic solution, Eq. (15.3), to this stochastic trajectory (black curve):

```
Show[
  Plot[100*Exp[-k1*t], {t,0,50},
    PlotLegends→{"deterministic"},
    PlotStyle→{Dashed}, AxesLabel→{"t","A(t)"}],
  ListPlot[trajectory,
    PlotLegends→{"stochastic"}, Joined→True] ]
```



Your stochastic trajectory will differ; this is an important feature in the kinetics of small numbers of particles. During a particular time interval, dt, a particle either reacts or does not. A collection of initially "identical" systems[5] would all behave slightly differently.

The deterministic approach used in Section 15.1.1 does not capture these statistical variations in the system. Moreover, the deterministic approach predicts noninteger numbers of species A at certain points in time, which is unphysical. In contrast, the stochastic result "jumps" between integer numbers of particles as the reaction occurs. The variations produced by the kinetic Monte Carlo algorithm reflect the *actual* behavior of a real physical system, and are *not* errors. Statistical fluctuations are inherent to these small systems, and are not averaged over a large ensemble (as is implicit in the deterministic solution). Do you expect these variations to become larger or smaller as the system size is increased? Perform a numerical experiment by modifying the code to test your answer.

15.2.3 Reversible first-order reactions

Next, consider a stochastic version of the reversible first-order reaction studied in Section 15.1.2, for the specific case of $k_1 = 0.1$, $k_2 = 0.01$, $N_A(t = 0) = 100$, and $N_B(t = 0) = 0$. For this reaction, it is unlikely that N_A will ever be zero; even if it were, the reverse reaction converting B to A would produce more. Therefore, the simulation is run for a predetermined duration of 2000 time units, instead of terminating when the particle count goes to zero. Aside from using time as the stopping criterion, the main difference from the previous example is the selection of which possible reaction to perform:

```
k1=0.1; k2=0.01;              (*define rate constants*)
nA=100; nB=0; currentTime=0; (*define initial conditions*)
trajectoryA={{currentTime,nA}}; trajectoryB={{currentTime,nB}};
While[(currentTime<2000),
  rate1=k1*nA;                (*compute rates of possible reactions*)
  rate2=k2*nB;
  rateTotal=rate1+rate2;
  If[(RandomReal[{0,rateTotal}]≤rate1), (*select reaction*)
    nA--;nB++; , (*reaction 1: forward reaction*)
    nA++;nB--;   (*reaction 2: reverse reaction*)
  ];
  currentTime+=drawTime[rateTotal]; (*update time*)
  AppendTo[trajectoryA,{currentTime,nA}];
  AppendTo[trajectoryB,{currentTime,nB}];
];
```

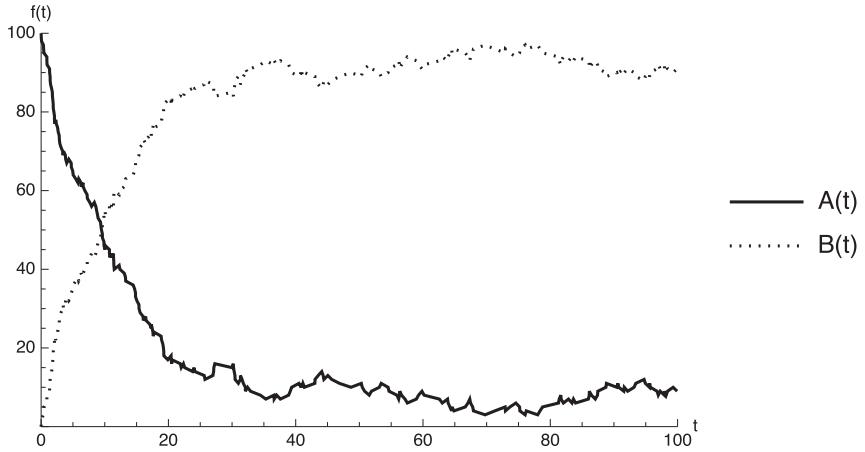
The rate at which each reaction in Eq. (15.5) occurs (`rate1` and `rate2`) is determined by the number of particles and the rate constants. The total exit rate is the sum of these two reaction rates. A uniform random number between zero and `rateTotal` is chosen using

5. Identical in the sense that they have the same number of each type of particle. What is beyond our control are the variations of the particular microscopic state of the system, which will give rise to the differences in time evolution modeled by the stochastic trajectory.

`RandomReal[]` and used to select the reaction outcome. `If[]` this random number is less than or equal to `rate1`, then reaction one is selected; otherwise reaction two is selected. The `If[]` statement increments and decrements the number of each particle type accordingly. As before, the `currentTime` is updated, and the trajectory of the number of particles as a function of time is constructed; each particle type has its own trajectory list.

Like the previous deterministic solution, the amount of A initially decreases and the amount of B increases. However, unlike the deterministic case, where the concentrations reach constant values, here the particle numbers fluctuate with time.

```
ListPlot[ {trajectoryA,trajectoryB},
  PlotRange→{{0,100},{0,100}},
  PlotLegends→{"A(t)","B(t)"},
  Joined→True, PlotStyle→{Black,Dotted},
  PlotMarkers→None, AxesLabel→{"time","f(t)"}]
```



Note that the total number of particles is constant; the system is in a state of dynamic equilibrium, and the peaks and troughs show interconversion between A and B. Like the deterministic solution, the system reaches equilibrium after a certain amount of time elapses, but this is only seen in the *average value* of the number of particles. The ratio of the average number of each type of particle (or the average of the particle-number ratio) should be the same as the deterministic case. (Check for yourself, by plotting this ratio as a function of time.) In general, the relative fluctuation becomes less important as the system size increases. In the deterministic version of this reaction, the fluctuations of a few particles isn't noticeable (e.g., for micromolar concentrations in a milliliter sample, there are 10^{14} or so particles). On the other hand, in this system with only 100 particles, the conversion of one particle is a large relative change in the system.

15.2.4 Parallel first-order reactions

Next, let's implement a stochastic version of the parallel first-order reaction from Section 15.1.3. To facilitate simulating many possible trajectories—and thus quantifying the variations that occur in small-particle-number systems—we'll implement it as a function,

whose arguments specify the rate constants (k1 and k2), initial conditions (startA, startB, and startC), and simulation duration (maxTime). The function is organized along the lines discussed previously:

```
stochasticParallelFirstOrder[k1_,k2_,startA_,startB_,startC_,maxTime_]:=
  Module[
    {currentTime=0,nA=startA,nB=startB,nC=startC,
     trajA={{0,startA}},trajB={{0,startB}},trajC={{0,startC}},
     rateList={0,0},random,whichRxn},

    While[(currentTime≤maxTime)&&(nA>0),
      rateList[[1]]=k1*nA; (*construct the list of rates*)
      rateList[[2]]=rateList[[1]]+k2*nA;

      random=RandomReal[{0,Last[rateList]}];
      whichRxn=1;
      Do[ (*determine which reaction happened...*)
        If[rateList[[i-1]]<random≤rateList[[i]], whichRxn=i; ];
        ,{i,2,Length[rateList]}];

      If[whichRxn==1, nA--; nB++;]; (*...then do it!*)
      If[whichRxn==2, nA--; nC++;];

      currentTime+=drawTime[Last[rateList]]; (*update time*)
      AppendTo[trajA,{currentTime,nA}];      (*update trajectories*)
      AppendTo[trajB,{currentTime,nB}];
      AppendTo[trajC,{currentTime,nC}];
    ];
    {trajA,trajB,trajC} (*return the trajectory lists*)
  ];
```

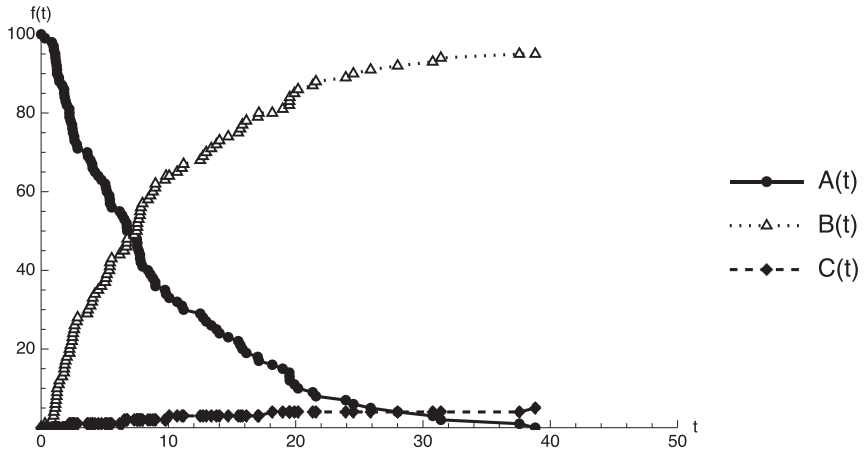
The simulation runs until the maximum time has elapsed or until there is no more species A left to react (there is no point in continuing otherwise, so we might as well stop; this also simplifies measuring the time to totally deplete A). Since there are only two possible reactions, one could use a single If[] statement to select the reaction to perform (like in the previous example). However, organizing the rates as a list is a strategy that is more easily generalized to larger sets of elementary reactions. The rateList is just a representation of the 1D dartboard shown in Figure 15.1. Each reaction is labeled with an arbitrarily chosen (but internally consistent) index (e.g., 1, 2, 3). There is nothing special about the particular labels that are assigned; they are merely used to organize the reactions. Each entry in rateList is the total of rates up to and including the current process, constructed at the current timestep. After computing this list of all possible exit rates, a uniformly distributed RandomReal[] number between zero and the sum of all the possible rates (the Last[] element in rateList) selects which reaction occurs. This is done by finding which "bin" (as shown in Figure 15.1) the random number falls into. The resulting value of whichRxn determines which reaction to perform (following the same indexing scheme used to construct rateList). A series of If[] statements changes the particle numbers consistent with the stoichiometry of the selected

reaction.[6] As before, the `currentTime` is updated using the `drawTime[]` function, and the trajectory lists for each of the particles is appended to include the current state of the system. The `While[]` loop terminates once `maxTime` has elapsed or there is no more of species A to react, and the function returns trajectories for each species.

Let's compute a sample trajectory for the parallel first-order reaction using $k_1 = 0.1$, $k_2 = 0.01$, $N_A(t = 0) = 100$, $N_B(t = 0) = 0$, and $N_C(t = 0) = 0$ using the `stochasticParallelFirstOrder[]` function:

AU: Do you want to edit the sentence to avoid loose lines?

```
k1=0.1; k2=0.01; (*define rate constants*)
ListPlot[
  stochasticParallelFirstOrder[k1,k2,100,0,0,100],
  PlotRange→{{0,50},{0,100}},
  PlotLegends→{"A(t)","B(t)","C(t)"},
  Joined→True, PlotStyle→{Black,Dotted,Dashed},
  AxesLabel→{"time","f(t)"} ]
```



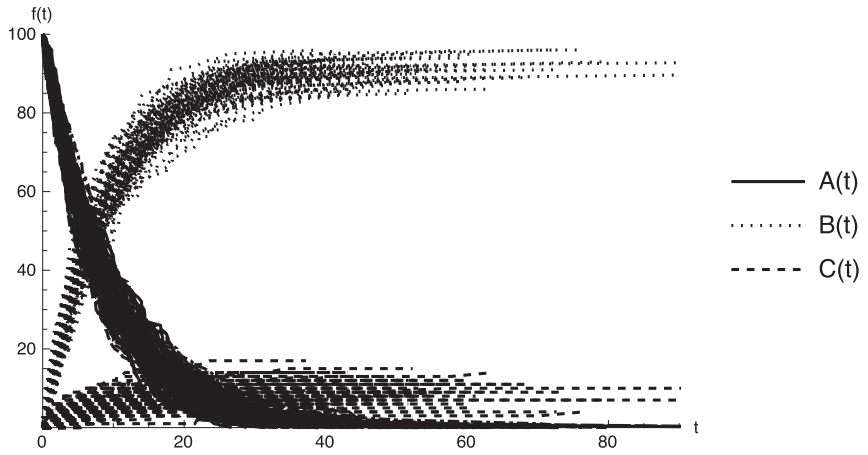
(Like the previous examples, expect to see a slightly different result when you run this calculation.) The behavior is qualitatively similar to the deterministic version shown in Section 15.1.3; species A is degraded into mostly species B and a lesser quantity of species C. Note that the simulation is stopped once there is no longer any species A, which stops the lines from extending further.

Let's examine the possible trajectory variations within a collection of these one-hundred-particle systems. Conceptually, this is equivalent to setting up multiple copies of the "experiment" and seeing what outcomes occur in each case. To track the outcomes of a hundred different experiments, we'll store each trajectory output from `stochasticParallelFirstOrder[]` in a list of `multipleTrajectories`. The corresponding `colorList`

6. For the experts: This can be done more concisely using the `Switch[]` function.

encodes which type of trajectory is being displayed. You are encouraged to change the `Dotted` and `Dashed` color descriptions into `Red` and `Blue` when you run this:

```
k1=0.1; k2=0.01;
multipleTrajectories={};colorList={};
Do[ (*repeat 100 trials*)
  {trajA,trajB,trajC}=stochasticParallelFirstOrder[k1,k2,100,0,0,100];
  AppendTo[multipleTrajectories,trajA];
  AppendTo[multipleTrajectories,trajB];
  AppendTo[multipleTrajectories,trajC];
  AppendTo[colorList,Black];
  AppendTo[colorList,Dotted];
  AppendTo[colorList,Dashed];
  ,{100}];

ListPlot[ multipleTrajectories,
  PlotRange→{{0,90},{0,100}},
  PlotLegends→{"A(t)","B(t)","C(t)"},
  Joined→True, PlotStyle→colorList, PlotMarkers→None,
  AxesLabel→{"t","f(t)"} ]
```



Observe how some example reactions go to completion much sooner than others and the final amounts of species B and C vary. The variations of these trajectories can be quantified by descriptive statistical measures, such as mean, minimum, maximum, standard deviation, skew, etc. (See Appendix B.4.) For example, how much time is required for the reaction to run to completion (i.e., complete disappearance of the reactant A)? To answer this, we'll run a sample of 1000 independent trajectories and collect the last time into a list of `finalTimes`, making use of the fact that the simulation stops when there is no more of species A:[7]

7. Note that the observed `Max[finalTime]` output value of `120.59` below is less than the limit of `maxTime = 200` provided as the last argument of `stochasticParallelFirstOrder[]`. If the `Max[finalTime]` were equal to this limit, it would suggest that the simulation had been stopped prematurely, and consequently that one should increase the `maxTime` argument value.

```
finalTimes={};
Do[ (* loop over independent trajectories *)
  {trajA,trajB,trajC}=stochasticParallelFirstOrder[k1,k2,100,0,0,200];
  AppendTo[finalTimes,trajA[[-1,1]]];
  ,{1000}]
```

The "typical" amount of time, described by the `Mean[]` and `Median[]`, is

```
{Mean[finalTimes], Median[finalTimes]}
```

```
{47.3255, 45.1399}
```

and the minimum (`Min[]`) and maximum (`Max[]`) amount of time are

```
{Min[finalTimes], Max[finalTimes]}
```

```
{24.1401, 120.59}
```

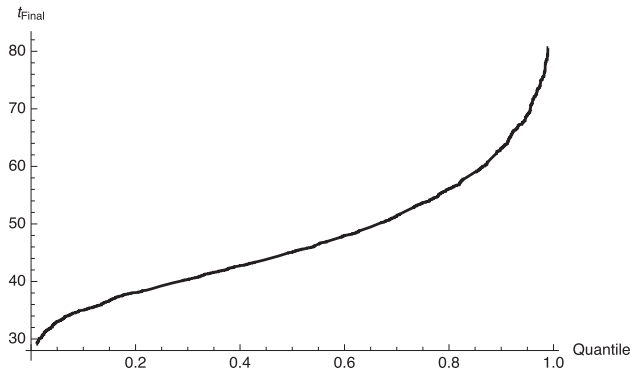
Surprisingly, the reaction can be over in as little as 24 time units or can take as long as 120 time units; these are significantly shorter and longer, respectively, than the average time. *Quantiles* provide a more nuanced description of the completion time distribution. You are already familiar with the median, i.e., the value at which half of the samples are larger and half are smaller. The quantile generalizes this to consider any fraction, i.e., the median is simply the particular case of the 1/2 quantile. Similarly, you have probably received standardized test results reported in percentile—i.e., the quantile expressed in percents. A score in the 90th percentile—i.e., the 0.9th quantile—corresponds to having a higher score than 90% of the other "samples." In a similar way, we can determine the time needed for the first 10% and first 90% of reactions to run to completion by computing the 0.1th and 0.9th `Quantile[]`s, respectively:

```
{Quantile[finalTimes,0.1], Quantile[finalTimes,0.9]}
```

```
{34.7935, 63.1713}
```

These results indicate that 10% (i.e., the 0.1th quantile) of the reaction samples finish in less than 35 time units, and 90% (i.e., the 0.9th quantile) finish within about 63 time units. This is useful information—although more than 120 time units are needed for all the reactions to reach completion, 90% are done in about half that time. The `Quantile[]` can also be displayed as a `Plot[]`:

```
Plot[ Quantile[finalTimes,x], {x,0.01,0.99},
  AxesLabel→{"Quantile","t_Final"} ]
```



How do you expect these variations to change as $N_A(t = 0)$ is increased?

A similar statistical analysis can be performed for the final N_C/N_B ratio by computing a sample of trajectories and then collecting the results into a list for subsequent analysis:

```
finalCBratios={};
Do[
  {trajA,trajB,trajC}=stochasticParallelFirstOrder[k1,k2,100,0,0,200];
  AppendTo[finalCBratios,(trajC[[-1,2]]/trajB[[-1,2]])];
  ,{1000}];
```

The `Mean[]` and `Median[]` of the particle-number ratio,

```
{Mean[finalCBratios], Median[finalCBratios]}//N
```

```
{0.10156, 0.0989011}
```

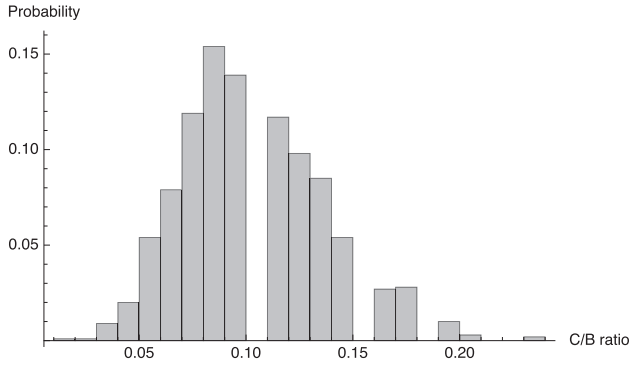
are essentially the same as the deterministic solution from Section 15.1.3,

```
Evaluate[ (c[100]/b[100])/.soln3 ]
```

```
0.1
```

but unlike the deterministic case, the stochastic results will have a wide variation in the ratio. This can be seen by plotting a `Histogram[]` of the `finalCBratios`:

```
Histogram[ finalCBratios, Automatic, "Probability",
  AxesLabel→{"C/B ratio","probability"} ]
```



The gaps in this `Histogram[]` are because some ratio values are not possible for this small number of particles. How do you expect these variations to change as $N_A(t=0)$ is increased?

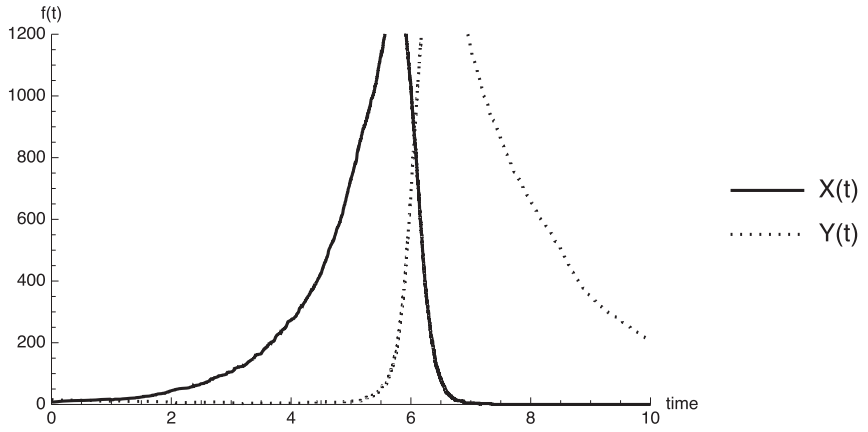
15.2.5 Lotka-Volterra Reaction

The Lotka-Volterra reaction introduced in Section 15.1.4 can be implemented by modifying the `stochasticParallelFirstOrder[]` function from the last section to incorporate reactions Eq. (15.7). Aside from changing the variable names (which is merely cosmetic), the only significant changes are adding the necessary reactions to the `rateList` and adding the necessary stoichiometric consequences for the randomly selected reaction.

```
stochasticLotkaVolterra[k1_,k2_,k3_,startS_,startX_,startY_,maxTime_]:=
  Module[
    {currentTime=0, nX=startX, nY=startY,
     trajX={{0,startX}}, trajY={{0,startY}},
     rateList={0,0,0}, random, whichRxn},

    While[(currentTime≤maxTime)&&((nX>0)||(nY>0)),
      rateList[[1]]=k1*nX*startS; (*construct the list of rates*)
      rateList[[2]]=rateList[[1]]+k2*nX*nY;
      rateList[[3]]=rateList[[2]]+k3*nY;

      random=RandomReal[{0,Last[rateList]}];
      whichRxn=1;
      Do[ (*determine which reaction happened...*)
        If[rateList[[i-1]]<random≤rateList[[i]], whichRxn=i;];
        ,{i,2,Length[rateList]}];

      If[whichRxn==1, nX++;]; (*...then do it!*)
      If[whichRxn==2, nX--; nY++;];
      If[whichRxn==3, nY--;];

      currentTime+=drawTime[Last[rateList]]; (*update time*)

      AppendTo[trajX,{currentTime,nX}]; (*update trajectories*)
      AppendTo[trajY,{currentTime,nY}];
    ];
  {trajX,trajY} (*return trajectory lists*)
];
```

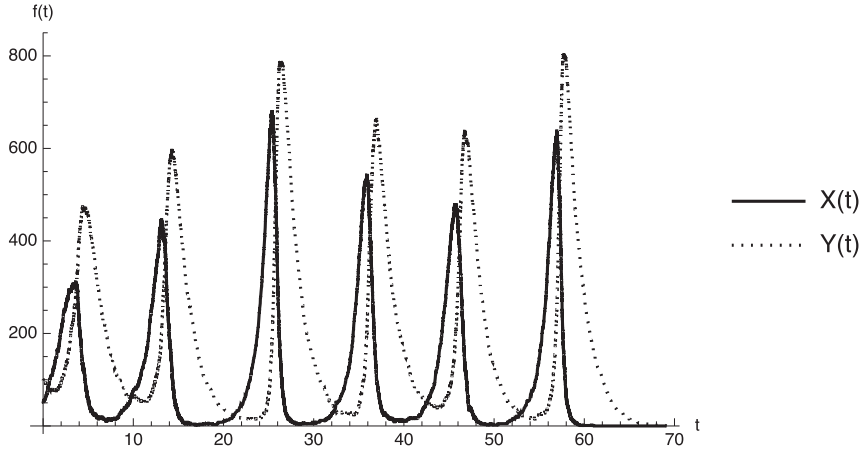
To test the function, consider rate constants $k_1 = 1$, $k_2 = 0.005$, $k_3 = 0.6$, initial conditions of $N_X(t=0) = 8$, $N_Y(t=0) = 16$, and a time-independent source number of $N_S(t) = 1$:

```
k1=1; k2=0.005; k3=0.6; (*define rate constants*)
ListPlot[ stochasticLotkaVolterra[k1,k2,k3,1,8,16,10],
  PlotRange→{{0,10},{0,1200}},
  PlotLegends→{"X(t)","Y(t)"},
  Joined→True, PlotStyle→{Black,Dotted},
  PlotMarkers→None, AxesLabel→{"time","f(t)"} ]
```



The stochastic Lotka-Volterra reaction is typically not stable over long periods when the total particle counts are small, because it is possible to completely use up all of species X, thus causing the cycle to terminate.[8] For example, the initial conditions of $N_X(t=0)=8$ and $N_Y(t=0)=16$ shown above typically only sustain one cycle. Different conditions can give rise to more robust fluctuations. For example, increasing the initial values to $N_X(t=0)=50$ and $N_Y(t=0)=100$ allows the oscillations to run longer:

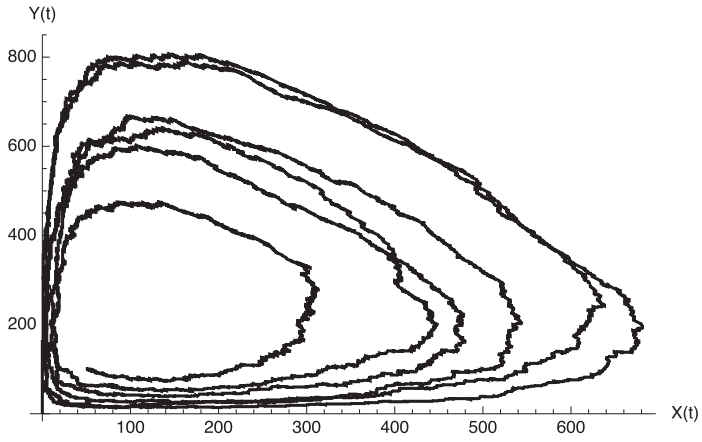
```
{trajX,trajY}=stochasticLotkaVolterra[k1,k2,k3,1,50,100,100000];

ListPlot[ {trajX,trajY},
  PlotLegends→{"X(t)","Y(t)"},
  Joined→True, PlotStyle→{Black,Dotted},
  PlotMarkers→None, AxesLabel→{"time","f(t)"}]
```



8. Referring back to the predator-prey analogy discussed in footnote 2, this corresponds to a low prey population, which is susceptible to going extinct. Stochastic kinetics can be used to determine the minimum population sizes needed to sustain a functioning ecosystem. Maintaining sufficiently large population sizes not only prevents genetic inbreeding, but is necessary to preserve ecosystem *kinetics* over long timescales.

This still only sustains three complete cycles before the population goes to zero. (Your results may vary due to stochastic fluctuations.) Another difference is that the maximum amounts of each species fluctuate from cycle to cycle, in contrast to the regular oscillations seen in the deterministic simulation of Section 15.1.4. These cycle variations are manifested as a "wandering" trajectory in the phase plane plot:

```
xyParamList={};
Do[
  AppendTo[xyParamList,{trajX[[i,2]],trajY[[i,2]]}]
  ,{i,Length[trajX]}];

ListPlot[ xyParamList,
  Joined→True, PlotMarkers→None,
  AxesLabel→{"X(t)","Y(t)"} ]
```



The population trajectory oscillates out of control, reaching large values of X and then large values of Y, before going to zero for both species. This is in contrast to the regular periodic behavior of the deterministic simulation in Section 15.1.4. The deviations between stochastic results and the deterministic predictions for low particle numbers has important implications for biological oscillatory reactions (e.g., circadian clocks, heartbeats). Reducing this type of noise requires increasing the rate and quantity of signaling events that occur, which can be energetically costly to implement in living organisms.[9] Biological systems have evolved to make sophisticated trade-offs between thermodynamic efficiency and noisy behavior.

9. A general proof for negative feedback systems has been described by I. Lesta, G. Vinnicombe, and J. Paulsson, "Fundamental Limits on the Suppression of Molecular Fluctuations," *Nature* **467**, 174 (2010), http://dx.doi.org/10.1038/nature09333.

15.3 LOOKING FORWARD

Both deterministic and stochastic kinetic simulations have their place in modeling chemical systems. Deterministic simulations are suitable for problems with sufficiently large concentrations of the reactant species, such that the discreteness of those species can be ignored. This is often the case in well-mixed chemical reactors, such as arise in chemical engineering or in atmospheric and geochemical processes.

Stochastic simulations are essential for simulating reactions with small concentrations, where it is crucial to take into account the discrete nature of the species involved. This includes most problems in molecular biology—for example, gene regulation "circuits" rely on stochastic effects.[10] A comparison of classical and stochastic kinetic modeling of biochemical reaction systems has been presented by Goutsias.[11] The stochastic algorithm used in this chapter is relatively simple; more sophisticated (and faster) methods are discussed in a review article by Gillespie.[12]

15.3.1 The role of diffusion

All of the examples in this chapter treat concentration only as a function of time—not as a function of spatial position. This is equivalent to the assumption of a uniform concentration through the entire reaction vessel. While this assumption is valid in a well-stirred solution, it does not hold if the solution is poorly mixed (e.g., the cytoplasm inside a cell or the atmosphere contained in the Los Angeles basin). The lack of active mixing leads to spatial reactant concentration variations, which only slowly equalize by diffusion. For example, even if there are many copies of an enzyme in one location, the effective reaction rate might be small if its substrates are localized somewhere else, since both the enzyme and the substrate must diffuse over some distance before reacting.

Diffusion effects can be incorporated into both the deterministic and stochastic simulations by tracking both the amounts of each species *and where they are located*. Instead of having a single number represent the amount of species A in the entire vessel, the amount of species A at each particular position is organized as a list. The list of rates will be generalized to include both chemical reactions (rates for each type of reaction using the reactant concentrations in each position) and diffusion processes (rates for each type of reactant diffusing into adjacent positions). The diffusion rate will depend on the medium (gas or liquid), the presence of active mixing conditions (e.g., wind in the case of atmospheric systems), and the molecular interactions of the solute in its solution. The actual numerical values for the diffusion rate can be either experimental or obtained from molecular dynamics simulations like those in Chapter 14.

---

10. H. H. McAdams and A. Arkin, "Stochastic Mechanisms in Gene Expression," *Proc. Natl. Acad. Sci. USA* **94**, 814–819 (1997).
11. J. Goutsias, "Classical versus Stochastic Kinetics Modeling of Biochemical Reaction Systems," *Biophys. J.* **92**, 2350–2365 (2007), http://dx.doi.org/10.1529/biophysj.106.093781.
12. D. T. Gillespie, "Stochastic Simulation of Chemical Kinetics," *Annu. Rev. Phys. Chem.* **58**, 35–55 (2007), http://dx.doi.org/10.1146/annurev.physchem.58.032806.104637.

PROBLEMS

15-1. Many reaction mechanisms involve an intermediate species. For example, the reaction $A + 2B \rightarrow Y + Z$ can occur via an intermediate, X, following the mechanism

$$A + B \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} X$$

$$X + B \xrightarrow{k_2} Y + Z$$

(a) Write the differential equations for [A], [B], [X] as functions of time.

(b) Solve the set of differential equations numerically using `NDSolve[]` and initial concentrations of 1 M for [A] and [B] (and nothing for [X], and rate constants $k_1 = 0.1\ M^{-1}s^{-1}$, $k_{-1} = 0.1\ s^{-1}$, and $k_2 = 10\ M^{-1}s^{-1}$. Plot your results. Because [X](t) is of much smaller magnitude than [A] and [B], either put it on a separate plot or multiply its concentration by a factor of 50 so that it is visible at the same scale.

(c) The solution to (b) indicates that a steady-state approximation for [X] is appropriate. Invoking the steady-state approximation, derive exact expressions for [A](t) and [B](t), and compute the percent error of these approximate expressions (with respect to the exact numerical solutions computed in part (b)), as a function of time.

15-2. Ozone creation and destruction in the stratosphere can be described by the following mechanism:

$$O_2(g) + h\nu \xrightarrow{j_1} 2O(g)$$

$$O(g) + O_2 + M(g) \xrightarrow{k_2} O_3(g) + M(g)$$

$$O_3(g) + h\nu \xrightarrow{j_3} O_2(g) + O(g)$$

$$O(g) + O_3(g) \xrightarrow{k_4} 2O_2(g),$$

where the j_i indicate rate constants for photochemical reactions incorporating the appropriate photon ($h\nu$) flux rates; i.e., the photon concentration does *not* have to be included in the differential equation expressions.

(a) Write the differential equations for [O], [O_2], [O_3] as functions of time.

(b) Solve this set of differential equations numerically using `NDSolve[]`. At an altitude of 30 km, the rate constants are $j_1 = 2.51 \times 10^{-12}\ s^{-1}$, $k_2 = 1.99 \times 10^{-33}\ cm^6 \cdot molecule^{-2} \cdot s^{-1}$, $j_3 = 3.16 \times 10^{-4}\ s^{-1}$, and $k_4 = 1.26 \times 10^{-15}\ cm^3$. Using initial concentrations of $[O_2] = 3.16 \times 10^{17}$ molecules $\cdot\ cm^{-3}$ and $[M] = 3.98 \times 10^{17}$ molecules $\cdot\ cm^{-3}$, calculate the equilibrium values of [O_3] and [O], by integrating the differential equations over a sufficiently long period of time. Is this period of time on the order of seconds, minutes, hours, days, weeks, months, etc.?

15-3. Write a stochastic simulation for the reaction intermediate problem (Problem 15-1), and run a calculation using initial conditions of $N_A = N_B = 100$, $N_X = 0$, and rate constants $k_1 = 0.1\ particle^{-1}time^{-1}$, $k_{-1} = 0.1\ time^{-1}$, and $k_2 = 10\ particle^{-1}time^{-1}$.

15-4. For the stochastic reversible first-order reaction example of Section 15.2.3, characterize the descriptive statistics (e.g., `Min[]`, `Max[]`, `Mean[]`, `Median[]`, `StandardDeviation[]`) of the N_B/N_A ratio once the system has reached equilibrium. How do these values compare to the dynamic equilibrium solution from deterministic kinetics shown in Section 15.1.2? Consider the cases of (a) $N_A(t=0) = 100$; (b) $N_A(t=0) = 1000$; and (c) $N_A(t=0) = 10000$.

15-5. For the stochastic Lotka-Volterra reaction described in Section 15.2.5, using the rate constants $k_1 = 1, k_2 = 0.005$, and $k_3 = 0.6$ and initial conditions $N_S(t) = 1$ (constant), $N_X(t = 0) = 50$, and $N_Y(t = 0) = 100$, characterize the descriptive statistics (e.g., `Min[]`, `Max[]`, `Mean[]`, `Median[]`, `Histogram[]`) for a sample of 1000 independent trajectories, for the following properties: (a) time until $N_X(t) = N_Y(t) = 0$; (b) maximum values of $N_X(t)$ and $N_Y(t)$; and (c) delay time between peak in $N_X(t)$ and the subsequent peak in $N_Y(t)$.

15-6. Dimerization reactions are a generalization of the reversible first-order reaction considered in Section 15.1.2:

$$\text{Dimerization}: \quad 2\text{A} \xrightarrow{k_1} \text{A}_2$$
$$\text{Dissociation}: \quad \text{A}_2 \xrightarrow{k_{-1}} 2\text{A},$$

where typically $k_1 >> k_{-1}$.

(a) Solve this as a problem in deterministic kinetics. First, write out the differential equations corresponding to this set of reactions. Then solve the equations numerically using `NDSolve[]`, with initial concentrations $[\text{A}](t=0) = 300$ mM, and $[\text{A}_2](t = 0) = 0$ mM, and rate constants $k_1 = 100\ \text{mM}^{-1}\,\text{s}^{-1}$ and $k_{-1} = 0.1\ \text{mM}^{-1}\text{s}^{-1}$. Plot your result. What is the equilibrium ratio $[\text{A}]/[\text{A}_2]$?

(b) Solve this as a problem in stochastic kinetics. Solve the equations numerically as described in Section 15.2, using initial quantities of A and A_2 of 300 and 0 particles, respectively, and $k_1 = 100\ \text{s}^{-1}$ and $k_{-1} = 0.1\ \text{s}^{-1}$. Plot one trajectory. What is the equilibrium ratio $[\text{A}]/[\text{A}_2]$? Describe the equilibrium ratio of a sample of 1000 reaction outcomes using statistical measures such as the `Mean[]`, `Median[]`, and `StandardDeviation[]`, and plot a `Histogram[]` of the observed outcomes.

15-7. The Michaelis-Menten mechanism describes how enzymes, E, act as biological catalysts that convert a substrate, S, into a product, P, by forming an enzyme-substrate intermediate complex, ES. The process is modeled by the set of reactions

$$\text{Binding}: \quad \text{E} + \text{S} \xrightarrow{k_1} \text{ES}$$
$$\text{Dissociation}: \quad \text{ES} \xrightarrow{k_{-1}} \text{E} + \text{S}$$
$$\text{Conversion}: \quad \text{ES} \xrightarrow{k_2} \text{E} + \text{P}$$

While it is a common exercise to solve the deterministic version of this problem analytically using the steady-state approximation and considering only the initial state of the system, in this problem you should consider the exact numerical solution without approximations.

(a) Solve this as a problem in deterministic kinetics. First, write the differential equations corresponding to this set of reactions. Then solve the equations numerically using `NDSolve[]`, with initial concentrations $[S](t=0) = 300$, $[E](t=0) = 20$, and $[SE](t=0) = [P](t=0) = 0$, and rate constants $k_1 = 1.5 \times 10^{-3}$, $k_{-1} = 10^{-4}$, and $k_2 = 0.1$ (in appropriate units). Plot your result. How long does it take for complete conversion of the substrate into the product (i.e., at what time does $[P](t) < 1$)?

(b) Solve this as a problem in stochastic kinetics. Solve the equations numerically as described in Section 15.2, using initial quantities S = 300, E = 20, SE = 0, and P = 0, and rate constants $k_1 = 1.5 \times 10^{-3}$, $k_{-1} = 10^{-4}$, and $k_2 = 0.1$ (in appropriate units). Plot one trajectory. Then characterize the time required for complete conversion of the substrate into product (i.e., the time required for P = 300) using statistical measures (e.g., `Min[]`, `Max[]`, `Mean[]`, `Median[]`, `Quartile[]`, `Histogram[]`) on a sample of 1000 reaction trajectories.

15-8. Prokaryotic cells use an "auto-regulation" gene expression mechanism, wherein a protein regulates its own synthesis. At high concentrations of protein, P, protein dimers, P_2, form, and these dimers bind the gene (G) associated with the protein monomer. Binding to the gene prevents transcription of messenger RNA (mRNA), R, and in turn prevents new protein, P, synthesis. Degradation processes inside the cell cause the mRNA and proteins to be destroyed at constant rates. A minimal kinetic model of this process consists of the following elementary processes:

$$\begin{aligned}
\text{Gene transcription}: &\quad \mathrm{G} \xrightarrow{k_1} \mathrm{G} + \mathrm{R} \\
\text{Protein synthesis}: &\quad \mathrm{R} \xrightarrow{k_2} \mathrm{R} + \mathrm{P} \\
\text{Protein dimerization}: &\quad 2\mathrm{P} \underset{k_{-3}}{\overset{k_3}{\rightleftharpoons}} \mathrm{P}_2 \\
\text{Gene repression}: &\quad \mathrm{G} + \mathrm{P}_2 \underset{k_{-4}}{\overset{k_4}{\rightleftharpoons}} \mathrm{P}_2\mathrm{G} \\
\text{mRNA degradation}: &\quad \mathrm{R} \xrightarrow{k_5} \emptyset \\
\text{Protein degradation}: &\quad \mathrm{P} \xrightarrow{k_6} \emptyset
\end{aligned}$$

For the purposes of this problem, consider the initial number of genes, G, to be 10 and the initial number of all other species to be zero. For the rate constants, use the values $k_1 = 0.01$, $k_2 = 10$, $k_3 = 1$, $k_{-3} = 1$, $k_4 = 1$, $k_{-4} = 10$, $k_5 = 0.1$, and $k_6 = 0.01$ (with appropriate units). Because the number of copies of the gene and all other species is so small, only a stochastic simulation is appropriate for modeling this system.

(a) Perform a simulation for 10^4 units of time. Make separate plots for the number of mRNA, protein, and protein dimers present as a function of time during the course of a single trajectory.

(b) Using the results from a single trajectory, plot a `Histogram[]` of the probability of finding a particular number of protein molecules present during the course of the simulation. Characterize the `Mean[]` and `StandardDeviation[]` of that distribution.

(c) Perform a simulation where the number of gene copies is G = 5, and characterize the `Mean[]` and `StandardDeviation[]` of the number of protein molecules present during the simulation. Explain why your results are reasonable.

(d) Generalize part (c) and perform simulations where the number of genes varies from G = 1, 2, 3, . . . 20. Use an `ErrorListPlot[]` to show the `Mean[]` and `StandardDeviation[]` of the number of proteins present for the values of G. Explain how the number of copies of the gene regulates the protein concentration. Is this behavior linear or nonlinear?

15-9. The Schlögl reaction,

$$\mathrm{A} + 2\mathrm{X} \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} 3\mathrm{X}$$

$$\mathrm{B} \underset{k_{-2}}{\overset{k_2}{\rightleftharpoons}} \mathrm{X},$$

is an example of a bistable process, in which the final concentration of a species, X, can evolve to either of two possible final states, depending on the initial conditions and stochastic fluctuations.[13] For the purposes of this problem, consider the initial numbers of X, A, and B to be 190, 2×10^5, and 4×10^5, respectively, and the rate constants to be $k_1 = 7.72 \times 10^{-6}$, $k_{-1} = 6 \times 10^{-3}$, $k_2 = 2.66 \times 10^{-2}$, and $k_{-2} = 167.5$ (with appropriate units).

(a) Solve this as a problem in deterministic kinetics. First, write the differential equations corresponding to this set of reactions. Then solve the equations numerically using `NDSolve[]` as described in Section 15.1. Plot the number of X molecules as a function of time over 2500 units of time.

(b) Solve this as a problem in stochastic kinetics, as described in Section 15.2, again over a period of 2500 units of time. Perform ten trajectories, and plot the number of X molecules as a function of time for each of these trajectories. Do you see a pattern emerging?

(c) Perform 1000 stochastic trajectories, and plot a `Histogram[]` of the final number of X molecules. Using these results, compute the fraction of trajectories in which the final number of X molecules is greater than the initial number of X, and the fraction in which the final number of X molecules is less than the initial number.

---

13. J. Mira, C. González Fernández, J. Martínez Urreaga, "Two Examples of Deterministic versus Stochastic Modeling of Chemical Reactions," *J. Chem. Educ.* **80**, 1488–1493, (2003), http://dx.doi.org/10.1021/ed080p1488.

The following problems are suitable for short, independent-study projects.

15-10. The planetary carbon cycle consists of "reactions" exchanging carbon dioxide between the atmosphere and the ocean. Using deterministic kinetics, simulate the effect of anthropogenic carbon production using the nonlinear "three-box" model described by Guido Fano in "A Primer on the Carbon Cycle," *Am. J. Phys.* **78**, 367–376 (2010), http://dx.doi.org/10.1119/1.3298429.

15-11. Glycolytic oscillations are an example of a biryhythmic oscillation. Using deterministic kinetics, simulate the minimal Decroly and Goldbeter model and visualize your results using a phase plane diagram, following the description by Ferreira et al., in "Uncovering Oscillations, Complexity, and Chaos in Chemical Kinetics Using Mathematica," *J. Chem. Educ.* **76**, 861–866 (1999), http://dx.doi.org/10.1021/ed076p861.

15-12. The VKBL model is a (relatively) simple two-gene circuit model, consisting of 16 reactions. Implement a stochastic simulation of this model following the description by Robert C. Hilborn in "Modeling Discrete-Variable Stochastic Dynamics: Ecological Populations, Gene Networks, and a Nanotube Ion Channel," *Am. J. Phys.* **82**, 466–475 (2014), http://dx.doi.org/10.1119/1.4870076.

15-13. The adsorption of a binary gas mixture on a surface is a competition between (thermodynamic) binding energy and (kinetic) binding rate. Combine a lattice gas Metropolis Monte Carlo simulation (Section 13.4) with a kinetic Monte Carlo simulation to model this competition, following the description in Burde and Calbi, "Early Removal of Weak-Binding Adsorbates by Kinetic Separation," *J. Phys. Chem. Lett.* **1**, 808–812 (2010), http://dx.doi.org/10.1021/jz900468t.