CHAPTER 3

Variation: Analytical Solutions of Ordinary Differential Equations

LEARNING OBJECTIVES

- Draw compartment diagrams that integrate system inputs and outputs
- Write a set of ordinary differential equations (ODEs) based on compartment diagrams
- Solve simple ODEs analytically
- Compare various systems based on key equation parameters, such as steady-state solution, step response, and response time

This chapter and the next two chapters focus on the workhorse of computational systems biology: sets of ODEs based on **conservation of mass** and **mass action kinetics**. Most models that are published in systems biology papers consist of a set of ODEs.

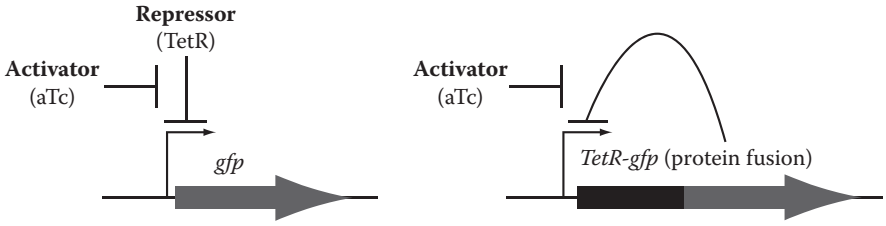
Steven Strogatz, an outstanding mathematician and teacher at Cornell University, says that every modeler should have three tools for solving sets of ODEs: analytical, graphical, and numerical solving techniques. We are going to investigate all of these tools in the context of our simple circuit. First, we'll solve the ODEs analytically to figure out what they are telling us about what feedback does and why it might be advantageous for a cell to use it.

SYNTHETIC BIOLOGICAL CIRCUITS

We can also begin to look at "real" circuits. The quotation marks around "real" are there because the circuits exist, but not naturally. Instead, they were assembled using the techniques of molecular biology and are called synthetic circuits. Figure 3.1 shows two synthetic gene transcription circuits, one lacking feedback and the other with feedback. The transcription units in both cases involve an activator to control gene expression and a reporter to indicate when transcription is active. The activator, anhydrotetracycline (aTc), binds a negative regulatory protein called TetR and prevents it from binding the promoter region of the gene. The "double negative" of aTc-inhibiting TetR repression leads to a net activation of gene expression when the transcription unit is in the presence of aTc.

The reporter protein, encoded by the gene, is a fluorescent protein from jellyfish called GFP (for green fluorescent protein). When the gene is transcribed and translated, these proteins fluoresce green, and this fluorescence can be detected and recorded.

The transcription unit in Figure 3.1a does not undergo negative autoregulation; it is simply *gfp* with a binding site for TetR in the promoter region. TetR is produced **constitutively** by another gene in this strain of *E. coli*, so the activity of the gene depends on whether sufficient aTc is present to prevent TetR from binding the *gfp* promoter. If aTc is present, gene expression occurs, and we can detect fluorescence; otherwise, there is neither expression nor fluorescence.



(a) Simple transcription circuit (b) Negative autoregulatory circuit

FIGURE 3.1 Synthetic transcription circuits without (a) or with (b) negative autoregulation. Expression of green fluorescent protein (GFP) is negatively regulated by binding of the repressor TetR, which is either encoded at another locus (a) or fused to the GFP coding sequence for autoregulation (b). TetR is itself inhibited by binding of the activator anhydrotetracycline (aTc), a chemical that can be added to the medium by the investigator. (Reprinted from Rosenfeld, N., Elowitz, M.B., and Alon, U. *Journal of Molecular Biology* 2002, **323**(5): 785–793, with permission from Elsevier.)

In contrast to the simple transcription unit of Figure 3.1a, the circuit drawn in Figure 3.1b includes autoregulatory feedback. In this case, the gene encodes a **fusion protein**: The *gfp* gene follows directly behind the *tetR* gene, so they are transcribed and translated as a single protein. Here, *E. coli* does not produce TetR constitutively (the native *tetR* is absent from this strain), so TetR is only present when gene expression of the synthetic circuit has occurred. The absence of aTc prevents expression of the fusion gene just as before. The presence of aTc, however, induces expression of the circuit and greatly increases the concentration of TetR::GFP (the double colon indicates the fusion protein). TetR binds tightly to aTc, so when enough TetR molecules are present, all aTc molecules are bound and effectively inactivated, allowing TetR to repress further gene expression.

These two transcription units were constructed to experimentally investigate the differences between a system with feedback and one without it. We are going to address the same question here using a set of ODEs that describe the system.

FROM COMPARTMENT MODELS TO ODES

I begin by giving you a brief introduction to ODEs using a rain gutter analogy. When I was young, I once kept measuring cups outside in the rain so that I could determine how much rainfall Palo Alto was receiving in an hour on rainy days. Let's pretend that I wanted to use that rainfall data and other information to determine what was happening to the water in the rain gutter on my house.

Figure 3.2a depicts a house in the rain, complete with rain gutter and drainpipe. As the rain pours into the gutter, water can either flow out of the drainpipe or build up in the gutter. We can indicate this conceptually by using a **compartment model**, as illustrated with a compartment diagram in Figure 3.2b. Here, the rate of rainfall *Rain* and the rate of drainage *Drain* are both considered as functions of time; *GW* indicates the amount of water in the rain gutter.

Let's assume that I can obtain a decent estimate of the rain and drain rates with my measuring cups and a stopwatch, and I want to see what happens to the amount of gutter water over time. I assume that water is not entering the gutter from anywhere but the rain and not leaving from anywhere except the drain. If I know the level of gutter water at a certain time t and want to predict the level after a short time period Δt, I can write the following equation:

$$GW(t + \Delta t) = GW(t) + Rain(t) \cdot \Delta t - Drain(t) \cdot \Delta t \qquad (3.1)$$

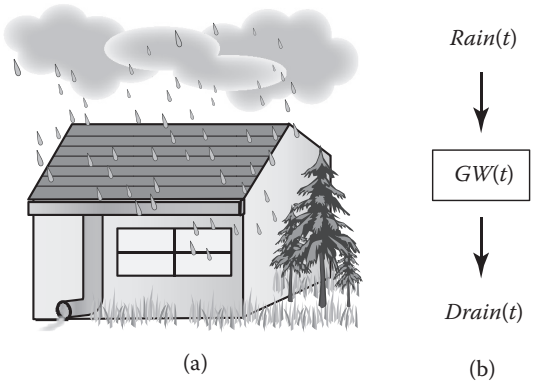


FIGURE 3.2 Conceptualizing ODEs with a rain gutter analogy. (a) During a rainstorm, water enters the gutter at rate *Rain*, flows out of the drainpipe at rate *Drain*, or builds up in the gutter to an amount *GW*. (b) The compartment diagram for the analogy indicates that water only enters the gutter from the rain and only exits the gutter from the drain; the diagram also captures the dependence on time of these processes.

In other words, the amount of gutter water at the later time is simply equal to how much water was in the gutter before, plus whatever fell in, minus whatever poured out. We calculate the amount of water that fell in by multiplying our gutter rainfall rate, which would be some amount over time, by the time interval Δt. Similarly, we multiply the drain rate by the time interval to determine the amount of water that poured out.

If we rearrange Equation 3.1 by subtracting $GW(t)$ from both sides and dividing by Δt, we obtain:

$$\frac{GW(t+\Delta t)-GW(t)}{\Delta t} = Rain(t)-Drain(t) \tag{3.2}$$

If we then take the limit of both sides of this equation as $\Delta t \to 0$, we obtain an ODE:

$$\frac{dGW}{dt} = Rain(t)-Drain(t) \tag{3.3}$$

where dGW/dt is the derivative of the amount of gutter water with respect to time. If I have a good enough understanding of how *Rain* and *Drain* depend on time or on *GW*, I can make some interesting predictions. For example, I can predict the amount of water in the gutter at

future times. I can also predict the effect of perturbations (for example, clogging the drain) on the time it takes for the gutter to overflow.

And there we have it—from a grade-school science fair project to an ODE! Systems of ODEs have been used for all kinds of applications. For example, ODEs not much more complicated than our rain gutter example have been used to model the spread of human immunodeficiency virus (HIV) from cell to cell and from person to person, leading to recommendations that had an impact on health policy. They have been used to model waste removal in patients undergoing dialysis and to describe the conversion of sugars to biofuel in microorganisms. Right now we're going to use them to describe our transcription units from Figure 3.1 in an effort to understand the importance of autoregulation.

In all of these cases, the general form of the ODEs looks like a generalized form of Equation 3.3 with more inputs or outputs:

$$\frac{dx}{dt} = \sum Rates_{production} - \sum Rates_{loss} \tag{3.4}$$

In other words, the change in amount or concentration of some entity x over time is equal to the sum of the rates of production of x minus the rates that lead to loss of x.

Take a moment to consider what we have just done. We started with a cartoon-like picture of a system (Figure 3.2a), moved first to a compartment model and diagram (Figure 3.2b), and then continued to a mathematical equation (Equation 3.3) that we can use to analyze, interpret, and even predict the behavior of the system.

If you can learn how to encode a biological network schematic into a set of equations like this, you are well on your way to making substantial, independent contributions to biological research! If your educational background is in biology instead of engineering, you may wonder whether you will be able to catch up with the engineers. Good news: In many ways, biologists are better suited for this part of model construction because it requires strong knowledge and intuition about the system. (In fact, I was once told this by the head of systems biology at a large pharmaceutical company!) If you have that knowledge or intuition, drawing the block diagrams and writing (and even solving, as you will see) the ODEs become fairly straightforward.

Let's practice by writing the equations for our feedback system with a few changes from Chapter 2. First, let's assume that the activator is always present—as you already learned, solutions without activator are

somewhat trivial. Second, we will explicitly include loss of the mRNA and protein in this chapter. Remember that loss can occur by degradation of the molecule or by dilution as cells grow and divide.

PRACTICE PROBLEM 3.1

Given the system in Figure 3.3, draw a compartment model for the mRNA and protein concentrations. Write the equations that describe how these concentrations change over time. Decompose the overall loss terms into decay and dilution components.

SOLUTION

The two compartment diagrams appear in Figure 3.4. Notice that there are two compartments now, and that each compartment has two possible ways for either mRNA or protein to be lost.

Using these diagrams, it is relatively straightforward to write the two ODEs for this system:

$$\frac{d[mRNA]}{dt} = Rate_{transcription} - Rate_{decay} - Rate_{dilution} \tag{3.5}$$

$$\frac{d[Protein]}{dt} = Rate_{translation} - Rate_{decay} - Rate_{dilution} \tag{3.6}$$

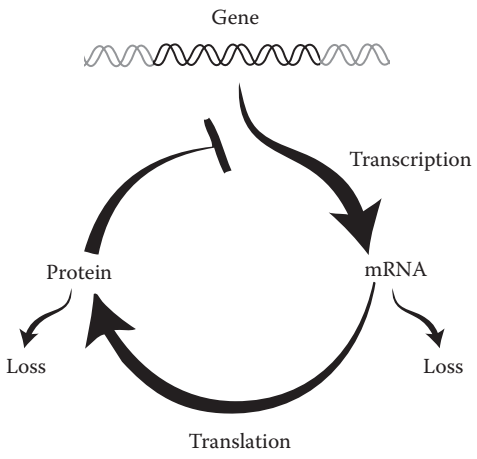


FIGURE 3.3 Slightly altering our feedback system for Practice Problem 3.1. This updated schematic includes the loss of mRNA and protein molecules as well as their production by transcription and translation, respectively. As in the original circuit, the protein product of this updated circuit represses its own transcription.

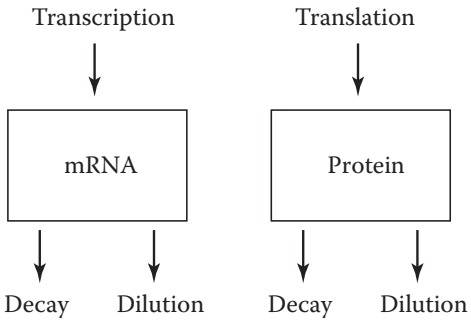


FIGURE 3.4 Compartment diagrams illustrating the mechanisms for creation and loss of mRNA (left) and protein (right) in our updated autoregulatory circuit. All of these processes are associated with specific rates.

SPECIFYING AND SIMPLIFYING ODES WITH ASSUMPTIONS

Now, we've accomplished the first step: writing the equations. However, these rate terms are not specific. How do we add detail to these rate expressions? There are several assumptions we can make about a given rate term. The simplest assumption would be that a given rate is zero or small with respect to other terms in the equation. Removing or simplifying terms as appropriate can make equations much easier to solve.

For example, you learned in Chapter 2 that protein loss by decay often takes much longer (occurs at a much slower rate) than loss by dilution. As a result, the decay rate will be substantially lower than the dilution rate, and we could write that $Rate_{decay} + Rate_{dilution} \approx Rate_{dilution}$, or $Rate_{decay} \approx 0$. In contrast, mRNA loss by decay is typically faster than the dilution rate, so we make the opposite assumption: $Rate_{decay} + Rate_{dilution} \approx Rate_{decay}$, or $Rate_{dilution} \approx 0$.

A second, related assumption is that a rate for a given system does not change significantly under the conditions that interest us. For example, certain genes are known as **housekeeping genes** because their protein products are so critical to cellular function that they are essentially always transcribed at a nearly constant rate. These genes are often used as controls in experiments monitoring gene or protein expression. For housekeeping genes, we would replace $Rate_{transcription}$ with a constant, such as k_{trs}, with units of concentration over time. This constant is a parameter of the model, which needs to be assigned a value to solve numerically. Ideally, we would measure k_{trs} for our circuit; if that were not possible, we would need to estimate it.

Another common assumption is that the process follows mass action kinetics: that the reaction rate is proportional to the product of the reactant concentrations. For example, mRNA decay depends on a reaction between the mRNA molecule and a class of enzymes called ribonucleases (RNases). Assuming that the rate follows mass action kinetics, the rate of mRNA decay at a given moment in time is proportional to the product of the mRNA and RNase concentrations at that time. We then write:

$$Rate_{decay} \propto [mRNA][RNase] \tag{3.7}$$

Often, this kind of equation is simplified by assuming that certain reactants such as RNase are present at constant levels. If we simplify Equation 3.7 in this way and define a kinetic constant to describe the proportionality between the mRNA concentration and the transcription rate, we obtain:

$$Rate_{decay} = k_{mdec}[mRNA] \tag{3.8}$$

In other words, the more mRNA that exists, the more that will be degraded in a given interval of time. Moreover, the units of k_{mdec}, 1/time, are different from the translation constant k_{trs}, which has units of concentration over time. Looking at Equation 3.8, you may be asking: "Wait a minute, where did the RNase concentration go?" The answer is that the RNase concentration is bundled as part of k_{mdec} (see the explanation for Equation 3.7).

The three assumptions described above—rate of zero, constant rate, and mass action kinetic rate—are by far the most common starting points for building an ODE-based model of a biological system. You will see that there are other ways to describe rates as well, but for now, simplifying the equation for protein concentration using these three assumptions is pretty straightforward. We use the mass action kinetics assumption for all three terms on the right-hand side of the equation:

$$\frac{d[Protein]}{dt} = k_{trl}[mRNA] - k_{pdec}[Protein] - k_{pdil}[Protein] \tag{3.9}$$

Protein decay and dilution are simply proportional to the protein concentration, and because translation creates protein from mRNA, we assume that the translation rate k_{trl} is proportional to the mRNA concentration. Just as we assumed that the RNase concentration did not change much over time to obtain Equation 3.8 from Equation 3.7, we are

assuming that the concentrations of ribosomes and protease are relatively stable and can therefore be bundled into our kinetic constants. Since both the decay and the dilution terms depend on the protein concentration, we can also combine these terms into one general "loss" term:

$$\frac{d[Protein]}{dt} = k_{trl}[mRNA] - k_{ploss}[Protein] \tag{3.10}$$

Our mRNA equation is somewhat harder to write as a result of our system's autorepression. We need to come up with an expression for the rate of mRNA production that reflects the idea that less mRNA will be produced if there is a lot of protein available. We do not know exactly what this function looks like yet, but we want to assert that protein concentration is the key player. For now, let's just say that the rate of transcription is a function of the protein concentration, $fxn_{trs}([Protein])$. We can then rewrite Equation 3.5 as:

$$\frac{d[mRNA]}{dt} = fxn_{trs}([Protein]) - k_{mloss}[mRNA] \tag{3.11}$$

Except for that as-yet-unknown function, Equations 3.10 and 3.11 are pretty well specified.

THE STEADY-STATE ASSUMPTION

Can we reduce this set of two equations to only one equation that retains the key information? One way to approach this simplification is to consider timescales, just as we did with our Boolean modeling. In Chapter 2, you learned that the production of mRNA and protein occurs at similar rates, but that mRNA decays in general more quickly than protein. This difference in the rate of loss means that if there is a perturbation to our system, the mRNA concentration will probably recover to a stable level more quickly than the protein concentration. As a result, we can consider what happens to the protein concentration when mRNA is considered to be at its **steady-state** concentration, which we will call $[mRNA]_{ss}$. Thus:

$$\frac{d[mRNA]_{ss}}{dt} = 0 = fxn_{trs}([Protein]) - k_{mloss}[mRNA]_{ss} \tag{3.12}$$

rearranges to:

$$[mRNA]_{ss} = fxn_{trs}([Protein])/k_{mloss} \tag{3.13}$$

We can then reduce our two-equation system into a single equation, substituting this steady-state value into our protein equation:

$$\frac{d[Protein]}{dt} = k_{trl}[mRNA]_{ss} - k_{ploss}[Protein] \tag{3.14}$$

$$\frac{d[Protein]}{dt} = \frac{k_{trl}}{k_{mloss}} fxn_{trs}([Protein]) - k_{ploss}[Protein] \tag{3.15}$$

To simplify Equation 3.15, let us define another function, fxn_{trl} ([*Protein*]):

$$fxn_{trl}([Protein]) = \frac{k_{trl}}{k_{mloss}} fxn_{trs}([Protein]) \tag{3.16}$$

Notice that fxn_{trl} is also only a function of the protein concentration. Substituting our new function into Equation 3.15, we obtain:

$$\frac{d[Protein]}{dt} = fxn_{trl}([Protein]) - k_{ploss}[Protein] \tag{3.17}$$

and now we have an equation that is written only in terms of a single variable: [*Protein*].

SOLVING THE SYSTEM WITHOUT FEEDBACK: REMOVAL OF ACTIVATOR

At this point, let's revisit the actual nature of our translation function fxn_{trl} (Equation 3.16). First, consider the case without autoregulatory feedback (Figure 3.1a). Remember, when activator is added to the system, the gene is expressed at a constant high rate; when the activator is removed, gene expression ceases. We can represent this scenario mathematically as:

$$fxn_{trl}([Protein]) = \begin{cases} 0 \text{ (no activator)} \\ k_{trl,\,max} \text{ (activator)} \end{cases} \tag{3.18}$$

where $k_{trl,\,max} = (k_{trl} \cdot k_{trs,\,max})/k_{mloss}$ and $k_{trs,\,max}$ is the absolute maximum amount of transcription that can occur from the gene. There are two cases of primary interest: We either remove activator from an active system or we add activator to an inactive system. Let's consider each case in turn.

First we have the case in which the system has been expressing protein for a while and has reached a steady state; then, the activator is removed. Assume for now that removal of the inducer leads to immediate cessation of protein production. Thus, $fxn_{trl} = 0$, and:

$$\frac{d[Protein]}{dt} = -k_{ploss}[Protein] \tag{3.19}$$

This equation can be readily solved analytically (Sidebar 3.1):

$$[Protein](t) = [Protein]_{t=0}\, e^{-k_{ploss}t} \tag{3.20}$$

SIDEBAR 3.1 ANALYTICALLY SOLVING A SIMPLE ODE

Given an ODE describing a variable u such that:

$$\frac{du}{dt} = au$$

First, rewrite the ODE so that all of the terms involving u are on one side and terms involving t are on the other:

$$\frac{du}{u} = a\,dt$$

Next, perform the integration of both sides. An integral table may be helpful, and these are easy to find online:

$$\ln\left(\frac{u}{u_0}\right) = a\left(t - t_0\right)$$

Assuming that $t_0 = 0$, solving for u leads to:

$$u(t) = u_0 e^{at}$$

You can double-check your answer by differentiating:

$$\frac{du}{dt} = \frac{d\left(u_0 e^{at}\right)}{dt} = a\left(u_0 e^{at}\right) = au$$

Notice that the solution involves an **exponential** component, in which e is raised to the power of at. Exponential terms will play a large role in our discussion of ODE models. The process of verifying that Equation 3.20 is the solution of Equation 3.19 is analogous.

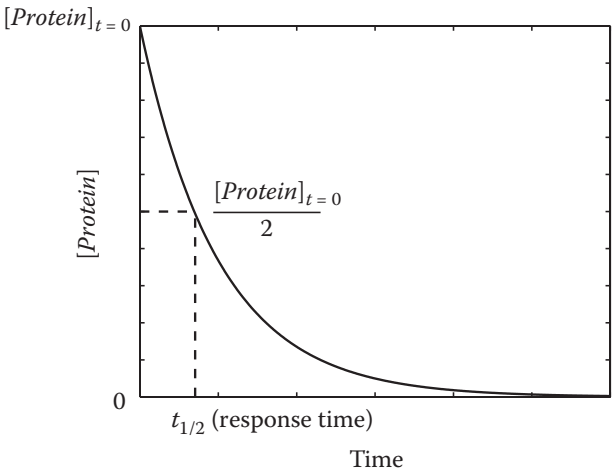


FIGURE 3.5 Removing the activator immediately inhibits transcription of our circuit, causing the protein concentration to decay over time. By assuming a solution with an exponential form, we can monitor the protein concentration at any given point in an experiment. The response time of the system occurs when the initial protein concentration has been halved.

From Equation 3.20, it is relatively easy to graph the protein concentration over time. As t increases, the value of the exponential term becomes smaller and smaller, approaching zero (Figure 3.5). This scenario is called an exponential decay, and you will see many of these kinds of terms as we continue.

KEY PROPERTIES OF THE SYSTEM DYNAMICS

I used MATLAB to construct Figure 3.5, but you could just as easily sketch it with three quick calculations and your intuition. These three calculations represent important aspects of the system dynamics: (1) where the system starts, (2) what happens at long times, and (3) how the system transitions from the initial to the final conditions.

First, you already know that the initial protein concentration is $[Protein]_{t=0}$, so you can plot a point at (0, $[Protein]_{t=0}$). Second, when the system reaches a steady state, $d[Protein]/dt = 0$, so in this case $[Protein]_{ss}$ must equal zero as well. Therefore, you can plot that at long times $[Protein] = [Protein]_{ss} = 0$.

Finally, you can use the equation to determine the time $t_{1/2}$ at which the initial protein concentration is decreased by half, so that

$[Protein]_{(t=t_{1/2})} = 1/2[Protein]_{(t=0)}$. This time is sometimes called the **response time** and is determined by substitution:

$$[Protein](t = t_{1/2}) = [Protein]_{t=0} e^{-k_{ploss} t_{1/2}} = \frac{[Protein]_{t=0}}{2} \tag{3.21}$$

Simplifying, we find that:

$$t_{1/2} = \frac{\ln(2)}{k_{ploss}} \tag{3.22}$$

Using these three points, and knowing that there is an exponential decay in Equation 3.20, you could draw Figure 3.5 by hand. Of course, you could also use MATLAB to draw the exact plot:

```
% set constants equal to one for illustrative purposes
Protein_0 = 1;
k_ploss = 1;
time = 0:0.1:6;
Protein_t = Protein_0 * exp(-k_ploss * time);
plot(time,Protein_t);
```

SOLVING THE SYSTEM WITHOUT FEEDBACK: ADDITION OF ACTIVATOR

Now, let's consider the case in which there is no gene expression and the activator is added such that gene translation suddenly occurs at its maximal rate $k_{trl,\,max}$. In this case, $[Protein]_{t=0} = 0$, and the steady-state solution is:

$$\frac{d[Protein]}{dt} = 0 = k_{trl,\,max} - k_{ploss}[Protein]_{ss} \tag{3.23}$$

$$[Protein]_{ss} = \frac{k_{trl,\,max}}{k_{ploss}} \tag{3.24}$$

The full analytical solution is not derived here, but it is common enough to find in an integral table; the solution is:

$$[Protein](t) = [Protein]_{ss}\left(1 - e^{-k_{ploss} t}\right) \tag{3.25}$$

Once again, you see an exponential component as the protein concentration rises to its steady-state value (Figure 3.6). Notice also that the steady-state

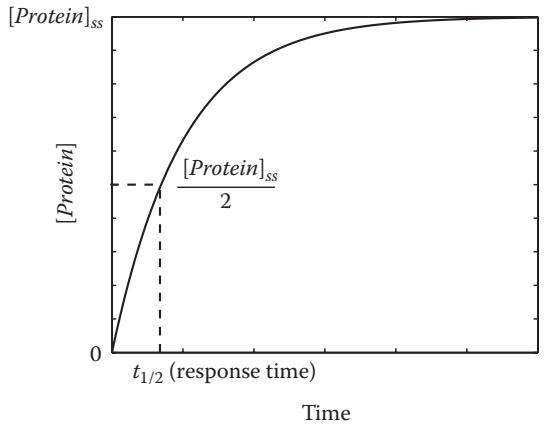


FIGURE 3.6 Dynamics of our system when addition of activator immediately drives translation to its maximal rate. Notice that the response time of this system is the same as the response time of the system in Figure 3.5.

solution of the system when activator is removed is equal to the initial condition of the system when activator is suddenly added and vice versa.

With Equation 3.25, you can also determine the response time of the activation of gene expression:

$$[Protein]_{(t=t_{1/2})} = [Protein]_{ss}\left(1 - e^{-k_{ploss}t_{1/2}}\right) = \frac{[Protein]_{ss}}{2} \tag{3.26}$$

Interestingly, the response time is the same as calculated in Equation 3.22:

$$t_{1/2} = \frac{\ln(2)}{k_{ploss}} \tag{3.27}$$

What does this mean? Recall that k_{ploss} is a combination of two terms, a dilution term and a decay term, and the dilution term is dominant. The dilution term is related to the observation that every cell division reduces the protein concentration in each daughter cell by half. In other words, the time it takes to divide the protein concentration in half is the time it takes for one cell to divide. Therefore, $t_{1/2}$ is simply the doubling time of the bacterium, and because doubling time is related to growth rate by:

$$\text{doubling time} = \frac{\ln(2)}{\text{growth rate}} \tag{3.28}$$

k_{ploss} is roughly equal to the growth rate.

COMPARISON OF MODELING TO EXPERIMENTAL MEASUREMENTS

Now, let's look at some experimental measurements. Figure 3.7 shows the results of an experiment using the genetic construct in Figure 3.1a, where the system is suddenly induced from a state of no gene expression. You can see that the theory and experiment agree nicely.

Let's review what we have learned so far in this chapter. We focused on a construct without any feedback (Figure 3.1a) for which the input functions and rates are relatively simple. We discussed some critical assumptions, such as mass action kinetics, and relating timescales to each other. We examined the response time as well as the steady states. We also worked on building intuition: If you know a steady state and you know how quickly something responds, you can sketch a simple model, equations, and often even a plot on the back of an envelope.

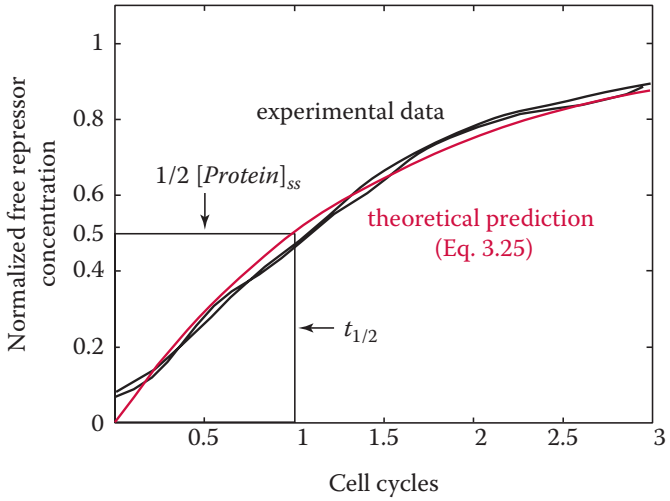


FIGURE 3.7 Experimental measurement of protein (repressor) concentration over time following sudden induction of a system without feedback (Figure 3.1a). *E. coli* cells were grown in batch culture and periodically monitored for GFP fluorescence, which served as a proxy for protein concentration. The experimental measurements (black lines) are in reasonable agreement with the concentrations predicted (red line) by the set of ODEs. Note that the normalized protein concentration is halved after one cell cycle, as predicted by Equations 3.27 and 3.28. (Modified from Rosenfeld, N., Elowitz, M. B., and Alon, U. *Journal of Molecular Biology* 2002, **323**(5): 785–793, with permission from Elsevier.)

ADDITION OF AUTOREGULATORY FEEDBACK

Let's see what happens when we add autoregulatory feedback, using the system in Figure 3.1b as our example. We apply some of the same steps learned previously to determine why autoregulatory feedback loops are so common in *E. coli*'s transcriptional regulatory network.

The key challenge in this case will be to determine fxn_{trs} and its dependence on the protein concentration. To begin thinking about this, look at the depiction in Figure 3.8 of the interaction between the protein and the operator region. Free protein binds to free DNA to form a complex, which can also dissociate to release protein and free DNA. The association and dissociation processes both have kinetic constants associated with them.

PRACTICE PROBLEM 3.2

Given the system in Figure 3.8, write an ODE that represents the concentration of bound DNA over time. State any assumptions that you make. How is the bound DNA concentration related to the free DNA concentration under steady-state conditions?

SOLUTION

Bound DNA is produced by association of free protein and DNA and lost by dissociation of the protein-DNA complex. Your initial equation will therefore look something like:

$$\frac{d[DNA_{bound}]}{dt} = Rate_{association} - Rate_{dissociation} \tag{3.29}$$

To further specify the rates, assume that association and dissociation follow mass action kinetics:

$$\frac{d[DNA_{bound}]}{dt} = k_a[Protein][DNA_{free}] - k_d[DNA_{bound}] \tag{3.30}$$

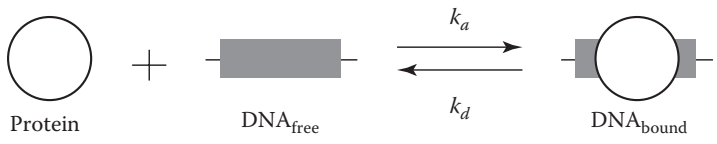


FIGURE 3.8 The dynamics of binding of a transcription factor (protein) to free DNA include association (k_a) and dissociation (k_d) kinetics. The transcription factor binding site is represented as a gray box on the DNA (straight line).

At steady state:

$$\frac{d[DNA_{bound}]_{ss}}{dt} = 0 = k_a[Protein][DNA_{free}]_{ss} - k_d[DNA_{bound}]_{ss} \tag{3.31}$$

Rearranging, we find that:

$$[DNA_{bound}]_{ss} = \frac{k_a}{k_d}[Protein][DNA_{free}]_{ss} \tag{3.32}$$

If we define an **equilibrium** dissociation constant $K = k_d/k_a$, we can simplify the expression to:

$$[DNA_{bound}]_{ss} = \frac{[Protein]}{K}[DNA_{free}]_{ss} \tag{3.33}$$

Now, we have a relationship between the concentrations of free and bound DNA that depends on the protein concentration and the binding affinity between the protein and DNA. A strong affinity means that DNA and protein are likely to associate and not let go of each other; K is therefore small because $k_a >> k_d$. Similarly, a weak affinity leads to a high K.

We are after something a little different, however: We would like to know how much of the total DNA is active and can be transcribed into mRNA. Our transcription factor (TetR) is a repressor, so the free DNA is the active DNA. What is the ratio of free DNA to total DNA? Here, another conservation equation can be useful. We assume that the total amount of DNA is constant and that the DNA can only exist as free or bound. Then, we can write:

$$[DNA_{total}] = [DNA_{bound}] + [DNA_{free}] \tag{3.34}$$

Equation 3.34 is valid at any point in time, including at steady state. We can therefore substitute Equation 3.33 into Equation 3.34 to obtain

$$[DNA_{total}] = \frac{[Protein]}{K}[DNA_{free}]_{ss} + [DNA_{free}]_{ss} \tag{3.35}$$

Simplifying yields:

$$\frac{[DNA_{free}]_{ss}}{[DNA_{total}]} = \frac{1}{1 + \frac{[Protein]}{K}} \tag{3.36}$$

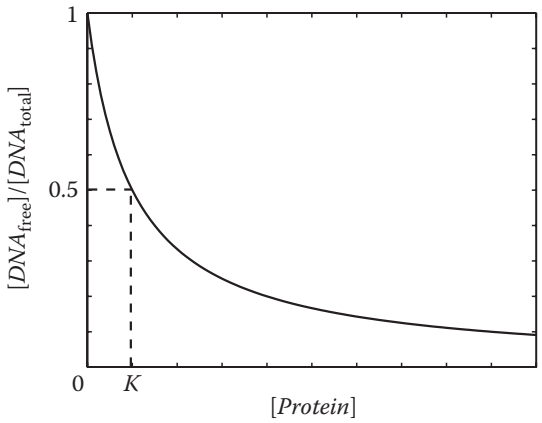


FIGURE 3.9 The fraction of free DNA is determined by the amount of protein available to bind the DNA. Note the definition of K, the equilibrium dissociation constant, when half the DNA in the system is bound by protein.

Take a minute to understand what this equation means, using Figure 3.9 as a guide. If there is no protein available to bind DNA ([*Protein*] = 0), then the concentration of free DNA is equal to the total amount of DNA because all DNA is free. If the protein concentration is very high such that [*Protein*] >> K, then the ratio of free DNA to total DNA is approximately zero, and virtually all DNA is bound. If the protein concentration is equal to K, then half of the DNA is free and the other half is bound. In other words, the fraction of active DNA is completely, and in this case negatively, dependent on the protein concentration for this system.

With the fraction of active DNA determined as a function of the protein concentration, we can identify a strategy for defining fxn_{trl}. Specifically, we can assume that the ratio of protein production to the maximum production rate is equal to the fraction of active DNA to the total amount of DNA:

$$fxn_{trl} = k_{trl,max} \frac{\left[DNA_{free}\right]_{ss}}{\left[DNA_{total}\right]} \tag{3.37}$$

Therefore:

$$fxn_{trl} = k_{trl,max} \frac{1}{1+\frac{\left[Protein\right]}{K}} \tag{3.38}$$

and our ODE now looks like this:

$$\frac{d[Protein]}{dt} = \frac{k_{trl,max}}{1+[Protein]/K} - k_{ploss}[Protein]. \tag{3.39}$$

Notice that the mass action assumption-based approach to modeling protein-DNA binding led to a more complicated term for protein production in Equation 3.39.

COMPARISON OF THE REGULATED AND UNREGULATED SYSTEMS

Now that the ODE is defined, let's compare this system (Figure 3.1b) to the one without feedback (Figure 3.1a). To do this, we will find the steady-state solution, the response curve for a sudden activation of gene expression (the **step response**), and the response time $t_{1/2}$.

The steady-state solution is determined from Equation 3.39 setting $d[Protein]/dt = 0$ and then solving for $[Protein]_{ss}$. In this case, we end up with a quadratic equation:

$$\frac{k_{ploss}}{K}[Protein]_{ss}^2 + k_{ploss}[Protein]_{ss} - k_{trl,max} = 0 \tag{3.40}$$

which we simplify by multiplying both sides of the equation by K/k_{ploss} and then obtain the solution:

$$[Protein]_{ss} = \frac{-K + \sqrt{K^2 + 4\frac{K\,k_{trl,max}}{k_{ploss}}}}{2} \tag{3.41}$$

We can simplify Equation 3.41 with one more assumption: very strong autorepression. As mentioned, this scenario means that $k_a >> k_d$; K is very small and can be neglected with respect to other terms. Applying this assumption, both inside and outside the radical, and dividing both the numerator and denominator by 2 yields:

$$[Protein]_{ss} = \sqrt{\frac{K\,k_{trl,max}}{k_{ploss}}} \tag{3.42}$$

Recall from Equation 3.24 that the system without feedback had a steady-state protein concentration equal to $k_{trl,\,max}/k_{ploss}$. Given that the value of K is expected to be very small, you would expect that $[Protein]_{ss}$ is significantly smaller with feedback than without. Indeed, Rosenfeld and colleagues (who carried out the experimentation with these circuits) calculated the steady-state protein concentration using these equations and determined approximate parameter values for bacterial repressors from the literature, estimating a steady-state protein concentration of ~4,000 proteins per cell in the case of no autoregulation and ~200 proteins per cell with autoregulation—a 20-fold difference!

To find the response time, we have to integrate Equation 3.39. You can refer to an integral table if you would like proof, but the solution is:

$$\begin{aligned} &t([Protein])-t([Protein]_0) \\ &=-\frac{1}{2k_{ploss}}\left[\ln\left(\left([Protein]-[Protein]_{ss}\right)\left([Protein]+K+[Protein]_{ss}\right)\right)\right. \\ &\left.+\frac{K}{K+2[Protein]_{ss}}\ln\left(\frac{[Protein]-[Protein]_{ss}}{[Protein]+K+[Protein]_{ss}}\right)\right]\Bigg|_{[Protein]_0}^{[Protein]} \end{aligned} \tag{3.43}$$

We can make a few assumptions to simplify this expression. For example, let's say that the starting time and protein concentration $[Protein]_0$ are both equal to 0. As before, we also assume strong autorepression and therefore that K is very small compared to the other terms. These assumptions enable us to reduce Equation 3.43 to:

$$t\left([Protein]\right)=-\frac{1}{2k_{ploss}}\ln\left(\frac{[Protein]_{ss}^2-[Protein]^2}{[Protein]_{ss}^2}\right) \tag{3.44}$$

Solving for the ratio of protein concentration to steady-state protein concentration:

$$\frac{[Protein]}{[Protein]_{ss}}=\sqrt{1-e^{-2k_{ploss}t}} \tag{3.45}$$

PRACTICE PROBLEM 3.3

Determine the response time $t_{1/2}$ for the autoregulated system in Figure 3.1b.

SOLUTION

We determine the response time with Equation 3.45 by setting the protein concentration equal to one-half of its steady-state value. Substitution yields:

$$\frac{1}{2} = \sqrt{1 - e^{-2k_{ploss}t_{1/2}}} \tag{3.46}$$

and by rearranging and solving, we obtain:

$$t_{1/2} = \frac{\ln(4/3)}{2k_{ploss}} \tag{3.47}$$

Now that we have investigated the unregulated and autoregulated systems, we can compare some of the key features. Figure 3.10 compares protein concentration with respect to the steady-state protein concentration over time (the perturbation response or step response), the steady-state protein concentration, and the response time for the unregulated and strongly autoregulated models. The results in each case appear similar, but the perturbation response leads to a somewhat shorter response time—about a five-fold difference if you work out the math. The steady-state protein level is also lower, as discussed previously.

	Without negative autoregulation	With negative autoregulation
Step response	$1 - e^{-k_{ploss}t}$	$\sqrt{1 - e^{-2k_{ploss}t}}$
Steady-state protein level	$\frac{k_{trl,\,max}}{k_{ploss}}$	$\sqrt{K\frac{k_{trl,\,max}}{k_{ploss}}}$
Response time	$\frac{\ln(2)}{k_{ploss}}$	$\frac{\ln(4/3)}{2k_{ploss}}$

FIGURE 3.10 Comparison of the key features of gene circuits without and with strong negative autoregulation. The circuits from Figure 3.1 have similar features, but the feedback system has a lower steady-state protein concentration as well as a shorter response time to perturbation.

The incorporation of autoregulation thus leads to a lower steady-state protein level and a faster response to perturbation such that the steady state is reached more quickly. In other words, feedback makes the circuit more responsive and efficient. In contrast, the circuit without feedback will either be producing more protein than is necessary or will be taking a relatively long time to respond to perturbations.

Are these predicted differences actually borne out in the laboratory? The answer turns out to be yes. We have already discussed how the steady-state protein levels are significantly lower in autoregulated circuits than in unregulated circuits. The same team of experimentalists measured the response times of both circuits in Figure 3.1; their results appear in Figure 3.11, which is similar to Figure 3.7. First, note that the *y* axis is scaled for comparative purposes—we expect the two circuits to have different steady-state values, but they have been normalized. The experimental data are shown near the theoretical predictions, and indeed, the response time is significantly shorter in the autoregulated circuit.

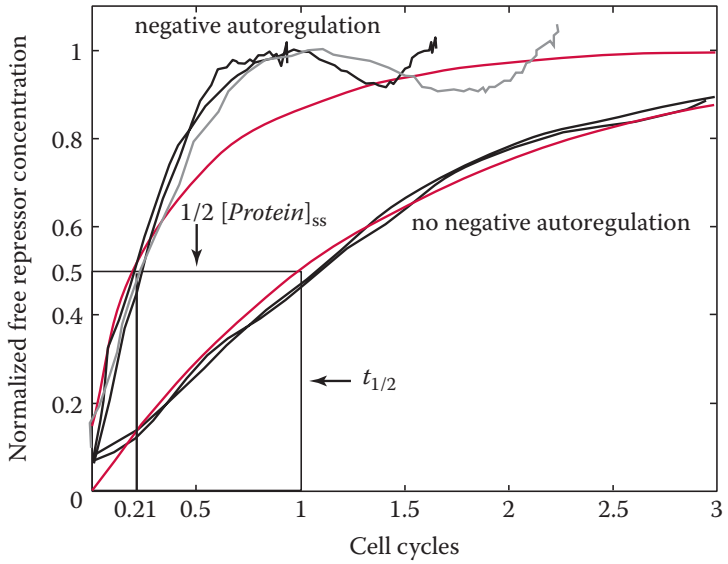


FIGURE 3.11 Experimental measurement of protein (repressor) concentration over time following sudden induction of systems without or with strong negative autoregulation. ODEs were used to predict the normalized protein concentrations for the system without feedback and the system with feedback (red lines). Experimental measurements are depicted with black and gray lines. (Modified from Rosenfeld, N., Elowitz, M. B., and Alon, U. *Journal of Molecular Biology* 2002, **323**(5): 785–793, with permission from Elsevier.)

Now that we have looked at two very different methods of approaching the same problem—autoregulatory negative feedback in transcriptional regulation—I think you will agree that the methods are quite different! I hope you have seen that these tools contribute several elements that we could not have deduced with our Boolean modeling. For example, we could not have calculated and compared response times and steady states. ODEs are a powerful tool in your systems biology tool kit!

By covering the analytical solution of ODEs in this chapter, I have also been trying to help you build more biological and computational intuition. Biologically, we unraveled another layer of the feedback scenario to gain further insight about the role that feedback plays in transcription and, as you will see further in this book, in other systems as well.

Computationally, I hope you saw how powerfully certain assumptions enabled us to create and solve these equations. The key assumptions were conservation of mass, mass action kinetics, strong affinity binding, and steady-state conditions, and they are going to show up again and again throughout this text. The next chapter focuses even more on intuition and back-of-the-envelope calculation using graphical approaches to solve ODEs.

CHAPTER SUMMARY

The most common approach for describing models of biological systems is to write a set of ODEs. A convenient way to conceptualize ODEs is to draw a compartment model diagram, which shows the variables as blocks; the processes that change the variable values are given as arrows pointing toward or away from the blocks. Once the variables and processes have been identified, the processes must be specified in more detail. Three common assumptions that can aid in specification are:

1. The rate at which the process occurs is very small compared to other processes and therefore can be neglected.

2. The rate at which the process occurs is constant and does not change based on any other variable or rate.

3. The rate at which the process occurs follows mass action kinetics, meaning that the rate is proportional to the product of the concentrations of the reactants.

Only relatively small and uncomplicated sets of ODEs can be solved analytically, but the analytical solution can give you important insights into how a system works. For this reason, it is often worth the effort to simplify your model, as we demonstrated by reducing our two-equation model to a single equation, in order to build intuition. This simplification depended on a further common assumption, based on the consideration of timescales: We assumed that one of our variables reached a steady state faster than the other variable. This assumption enabled us to solve for that steady-state value and substitute into the other equation.

In keeping with our focus on negative autoregulation in gene expression, we considered a pair of complementary synthetic gene circuits with one difference: One was negatively autoregulated, and the other was not. The analytical approach enabled us to derive the properties of the two circuits, including the response time, the step response, and the steady-state expression level. We found that the step responses were related, but different; in the autoregulated circuit, the response time was significantly shorter and the steady-state protein level was lower than in the nonautoregulated circuit. This observation suggests that autoregulation enables cells to respond quickly to changes in their environment by tuning the expression of fewer proteins. Finally, we showed that our theoretical analysis matched well with reported experimental measurements.

In summary, analytical solutions of ODEs add another layer of dynamical information and complexity to our modeling tool kit. We cannot see all of the possible solutions at once, as we did with Boolean models. Nevertheless, the gain in our ability to make quantitative predictions by using ODEs often outweighs this drawback. Moreover, techniques exist to simultaneously visualize multiple ODE systems, as you will see in the next chapter.

RECOMMENDED READING

Alon, U. *An Introduction to Systems Biology: Design Principles of Biological Circuits*. Boca Raton, FL: Chapman & Hall/CRC Press, 2007.

Bolouri, H. *Computational Modeling of Gene Regulatory Networks—A Primer*. London: Imperial College Press, 2008.

Rosenfeld, N., Elowitz, M. B., and Alon, U. Negative autoregulation speeds the response times of transcription networks. *Journal of Molecular Biology* 2002, **323**(5): 785–793.

Strang, G. *Linear Algebra and Its Applications*. 4th edition. Stamford, CT: Brooks Cole, 2005.

PROBLEMS

PROBLEM 3.1
Cell Growth

Cells growing in a nutrient-rich environment (such as *E. coli* growing in a flask full of rich broth) are observed to grow exponentially; their growth can be represented as:

$$\frac{dX}{dt} = \mu X$$

where X is the cellular biomass concentration, and μ is the growth rate.

a. Write the analytical solution of the growth equation, assuming an initial cellular biomass concentration of X_0.

b. Determine the relationship between the growth rate and the doubling time: the time at which $X(t) = 2X_0$. Relate this relationship to the concept of response time that you read about in this chapter.

c. Now, imagine that you are growing cells in a flask and measuring the concentration of cells over time. From your measurements, you determine a growth rate that you call μ_{flask}. However, when you watch these cells growing under a microscope, you notice that some of the cells lyse and die at a rate proportional to the biomass concentration, with rate constant k_{death}. How would you change the ODE in your cell growth model to account for the death of cells over time?

d. Write the analytical solution of the ODE in (c).

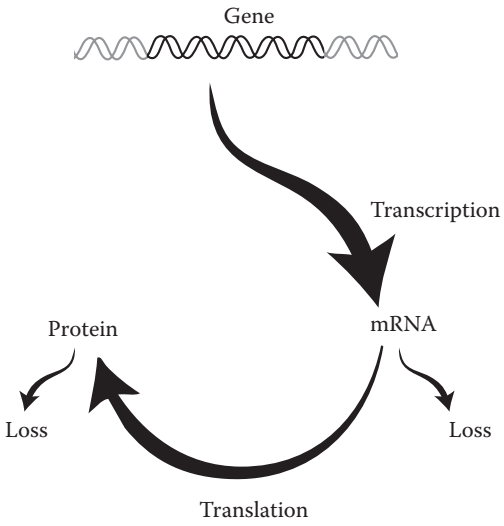
e. What condition is required for this system to reach steady state?

f. Going back to your microscope, you decide to measure the growth rate of individual cells by recording the time between cell division events in a single cell. Based on your model from (c), how would you expect your new single-cell measurements to compare to the growth rate that you measured in (a)?

PROBLEM 3.2
Negative Autoregulation

In this problem, we focus on the same circuits discussed in this chapter, but with more detail.

First, we consider the system without autoregulation.



a. Derive a single ODE for the change in protein concentration over time, given rate constants for the rates of mRNA and protein loss (k_{mloss} and k_{ploss}, respectively), translation (k_{trl}), and maximum transcription ($k_{trs,\,max}$). List the assumptions that were required to derive this equation.

b. Using the equation from (a), calculate k_{ploss} and the steady-state protein concentration in both micromoles and the number of proteins per cell. You may assume that $[Protein]_{(t=0)}$ = 0 μM, and that transcription is suddenly induced by the addition of an activator. You will also need the following constants:

   Division time of a cell = 35 min

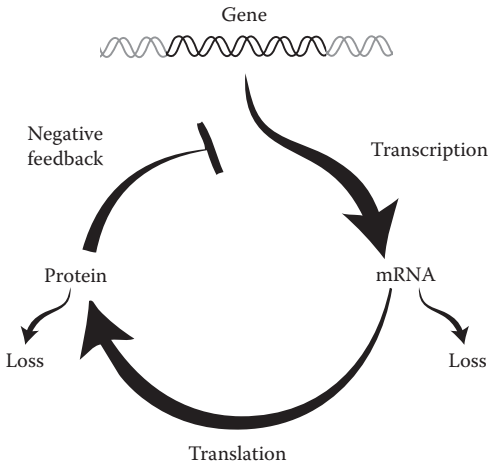
   $k_{trl,\,max}$ (maximum protein production rate) = 0.0015 μM/s

   1 μM protein concentration ≈ 1,000 protein molecules/cell

c. Write the analytical solution of the equation you derived in (a) and generate a graph of the change in protein concentration over time based on this equation and the constants given.

d. Now, use the MATLAB ODE solver `ode45` to plot the numerical solution to the ODE (type `help ode45` in the command line in MATLAB). How does this plot compare to your solution to (c)?

We consider now the case of negative autoregulation.



Here, the protein can bind to the operator, and transcription only occurs in the absence of protein binding. As before, K is the ratio of the rate of protein dissociation from the operator to the rate of protein association with the operator, and the rate of protein production is

$$\frac{k_{trl,max}}{\left(1+\frac{[Protein]}{K}\right)}$$

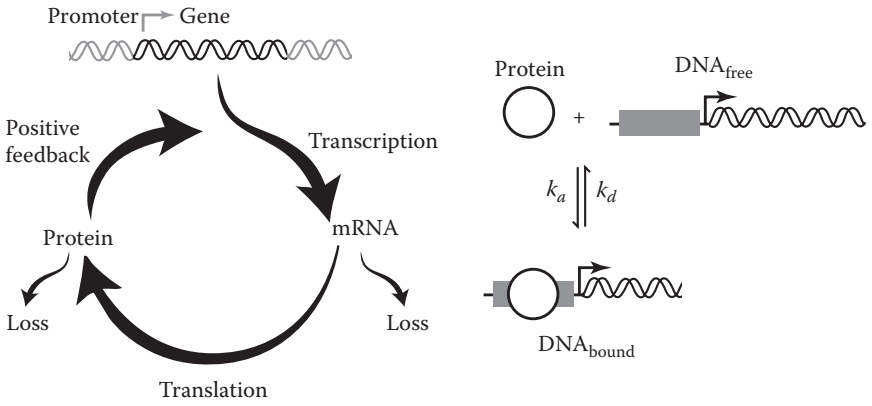
e. Calculate the steady-state protein concentration (in micromoles and proteins/cell) and the response time of the system. Be careful with units. The value for K is 15 nM.

f. Plot the changes in protein concentration over time using `ode45`. Again, the initial protein concentration is zero. Put this plot on the same axes as your output for (c).

g. Compare the values and plots from (e) and (f) to those for the case without regulation in (b) and (c). Comment briefly on the possible advantages of this difference in behavior in biological terms.

PROBLEM 3.3
Positive Autoregulation

In Problem 2.1, we considered an autoregulatory loop with positive, instead of negative, feedback. We now consider this circuit in more detail using ODEs.



On the right side of the figure for this problem is a diagram of the binding between the protein and free DNA (DNA_{free}) to form a complex, DNA_{bound} (you have already seen this diagram in Figure 3.8). In this case, only the bound DNA can be transcribed.

a. Write an ODE for the change in DNA_{free} over time in terms of DNA_{free}, DNA_{bound}, and *Protein*, together with the association and dissociation rate constants k_a and k_d.

b. What assumption did you make to write the equation in (a)? Give an example of a case for which this assumption does not hold.

c. Use your answer from (a) to derive an expression for the ratio of $[DNA_{bound}]$ to $[DNA_{total}]$. Your answer should be in terms of *Protein* and K, where $K = k_d/k_a$.

d. Use your answer to (c) together with Equation 3.17 to write a single ODE that describes the change in protein concentration over time.

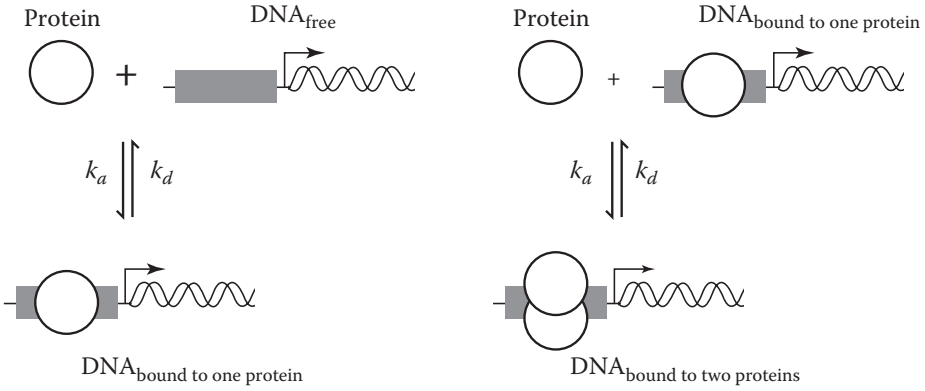
e. Write an equation for the steady-state protein concentration in terms of K, k_{ploss}, and $k_{trs,\,max}$. How does this new equation compare to the equations for the system with no regulation?

f. Using MATLAB, plot a graph of the dynamics for (1) the equation you used in (d), with $k_{trl,\,max} = 0.0008$ μM/s, $k_{ploss} = 0.0001925$/s, and $K = 0.010$ μM; (2) the same as in (1), but with $K = 1$ μM; and (3) the system without positive regulation. Plot them all on the same graph for 0–5 h. How do the three compare? Note that for (3) you can use the initial conditions $t = 0$ h and $[Protein] = 0$ μM, but for (1) and (2), you will prefer initial conditions such as $t = 0$ h and $[Protein] = 0.1$ μM. Why?

PROBLEM 3.4
Negative Autoregulation with Cooperativity

A common variation on our negative autoregulatory circuit is that, often, more than one binding site for the same protein occurs in the promoter region of the DNA. Therefore, more than one molecule of the same species can bind to the DNA and more strongly repress expression. This phenomenon is known as **cooperativity**.

We now analyze a negative autoregulatory circuit with cooperativity. In this case, two proteins can bind to the DNA, and both must be bound to the DNA to repress transcription.



The reactions describing the interactions between DNA and protein are:

$$protein + DNA_{free} \underset{k_d}{\overset{k_a}{\rightleftharpoons}} DNA_{bound\ to\ one\ protein}$$

$$protein + DNA_{bound\ to\ one\ protein} \underset{k_d}{\overset{k_a}{\rightleftharpoons}} DNA_{bound\ to\ two\ proteins}$$

a. From these reactions, derive an equation for $[DNA_{free}]/[DNA_{total}]$ in terms of protein concentration and K.

b. The equation that you will see most often in the literature as the answer to (a) looks like this:

$$\frac{[DNA_{free}]}{[DNA_{total}]} = \frac{1}{1+\left(\frac{[Protein]}{K}\right)^H}$$

where H is the Hill coefficient, which expresses the amount of cooperativity in the system. To obtain a solution in this form, you will need to cancel one term from your solution. Where does this term come from? Explain why it is reasonable to cancel this term in biological terms.

c. What is the value of H for our system? What happens when H is equal to 1? To what do you think higher values of H correspond?

d. For H = {1, 2, 3, 4, 5}, plot the ratio $[DNA_{free}]/[DNA_{total}]$ versus [*Protein*], where [*Protein*] varies from 0 to 0.05 μM. Let K = 10 nM. How do the curves change as H increases? What does this change mean in biological terms? What implications could this change have for modeling a cooperative reaction?

e. Substitute the equation given in (b) into Equation 3.17 and plot the changes in [*Protein*] over time for this case of negative autoregulation and cooperativity. Initially, the system contains no protein, and there is a sudden increase in $k_{trl,\,max}$ to 0.0008 μM. The value for k_{ploss} is 0.0001925/s. Use `ode45`. Show lines for H = {1, 2, 3, 4, 5}. How does the steady-state protein concentration change as H increases? Explain this trend in terms of the biology of the system.

CHAPTER 4

Variation: Graphical Analysis

LEARNING OBJECTIVES

- Generate plots of X versus dX/dt for single ODE systems
- Calculate fixed points and determine their stability using vector fields
- Draw time-course plots from arbitrary initial conditions using graphical analysis
- Understand how changing parameters can lead to bifurcation

When I ask my students about the most surprising moments they experienced in my systems biology course, the power of Boolean analysis is usually on their list. Many of them are particularly caught off guard by the number of conditions that you can examine simultaneously using this approach. By comparison, our ODE-based analytical approach is relatively limited; we only looked at a couple of initial conditions and had to make pretty specific assumptions to get there.

What makes graphical analysis so exciting is that, once again, we can examine many initial conditions at once to see how the system will behave. We can also deduce the ranges of initial conditions for which the response will be similar, and we can even characterize how parameter changes lead to dramatic, qualitatively different, altered responses.

Also, graphical analysis is *beautiful*. There is not a single figure I will show you here that I would not love to put on my wall at home! This aesthetic quality also helps your intuition; by visualizing many responses at once, your eyes and mind will help you to identify patterns that you can remember.

REVISITING THE PROTEIN SYNTHESIS ODES

Let's start with our ODE descriptions of our circuits in which the protein concentration varies without (Figure 3.1a) and with (Figure 3.1b) feedback, taken from Equations 3.23 and 3.39, respectively:

$$\frac{d[Protein]}{dt} = k_{trl,\,max} - k_{ploss}[Protein] \tag{4.1}$$

$$\frac{d[Protein]}{dt} = \frac{k_{trl,\,max}}{1+[Protein]/K} - k_{ploss}[Protein] \tag{4.2}$$

Remember that $k_{trl,\,max}$ is a constitutive (constant) production rate, and k_{ploss} is a rate constant that describes the relationship between protein concentration and protein loss over time, based on an assumption of mass action kinetics. The more complicated production term in Equation 4.2 also assumes mass action kinetics, but here it reflects the binding of repressor protein to DNA to form an inactive complex and the dissociation of that complex.

We solved these equations in Chapter 3. The solution to Equation 4.1 is:

$$[Protein](t) = [Protein]_{ss}\left(1 - e^{-k_{ploss}t}\right) \tag{4.3}$$

where

$$[Protein]_{ss} = \frac{k_{trl,\,max}}{k_{ploss}} \tag{4.4}$$

We then plotted Equation 4.3 (Figure 3.6) to visualize the dynamics of the system responding to a sudden increase in expression, starting at [*Protein*] = 0; this plot is shown again in Figure 4.1, but notice the new annotations in black and gray. First, the quantity $k_{trl,\,max} \cdot t$ is plotted in gray. At small values of t, when the response of the circuit has just started, the gray dashed line is a good approximation of the response.

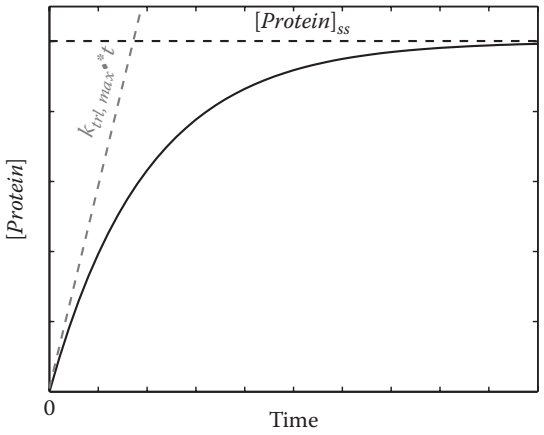


FIGURE 4.1 The response of our system (without feedback) to a sudden increase in protein expression (See also Figure 3.6). The initial increase in protein concentration closely approximates the translation rate (dashed light gray), but eventually, the system plateaus to a steady state (dashed black).

Reexamination of Equation 4.1 reveals that when not much time has passed, the protein concentration is low and the loss term is close to zero, leaving an equation in which $d[Protein]/dt \sim k_{trl,\,max}$. At larger values of t, the response is well approximated by the black dashed line, which is simply the steady-state protein concentration. At early times, the rate of change is therefore approximately $k_{trl,\,max}$, and at late times, the rate of change is zero.

PLOTTING *X* VERSUS *DX/DT*

Let's look at the rates of change in another way: still graphically, but this time plotting the variable value versus the variable's change over time. For our case, this strategy means plotting the protein concentration against the protein concentration over time, or [*Protein*] versus $d[Protein]/dt$. The plot in Figure 4.2 is a straight line, with the y intercept at $k_{trl,\,max}$ and a slope of $-k_{ploss}$. You could also determine these values by examining Equation 4.1, but you can gain insight about the system by visualizing it as a graph and comparing it to Figure 4.1.

In particular, notice that $k_{trl,\,max}$ is the instantaneous rate of change for the protein concentration over time when the protein concentration is equal to zero—our initial conditions in Figure 4.1. You can also determine the steady-state protein concentration simply by finding the spot on the graph where $d[Protein]/dt = 0$.

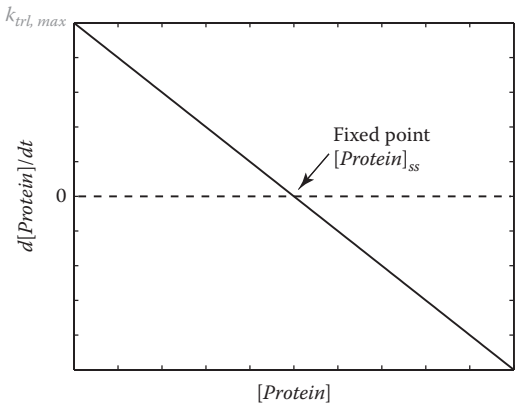


FIGURE 4.2 Comparing the change in protein concentration over time to the protein concentration reveals a linear relationship. As in Figure 4.1, the initial change in protein concentration over time mirrors the translation rate. The steady-state protein level occurs at $d[Protein]/dt = 0$; this point is called a "fixed" point.

FIXED POINTS AND VECTOR FIELDS

Points where the derivative of the variable is equal to zero are called **fixed points** of the system. One notable aspect of Figure 4.2 is that there is one and only one fixed point.

Fixed points can be either stable or unstable, with a critical impact on the system's behavior. Think of stability the way we discussed in previous chapters, beginning with Boolean analysis: If the system reaches a stable state, it will remain in that state. Likewise, a system in an unstable state moves away from that state.

In our case, you might already guess that the fixed point is stable because the trajectory in Figure 4.1 continues to approach it over time, but let's take this opportunity to show how you could infer this if you did not already know—by drawing a **vector field**. The vector field is an indicator of how much and in which direction the derivative is changing at a given variable value. We use our vector field to answer two questions. First, for a given protein concentration, will the rate of change be positive or negative? Second, how fast will the concentration change in that positive or negative direction?

The vector field for our system is shown in Figure 4.3 as a set of red arrows on the $d[Protein]/dt = 0$ line. We draw it by looking at the sign and magnitude of the $d[Protein]/dt$ value for a given protein concentration; for example, at $[Protein] = 0$, $d[Protein]/dt = k_{trl,\,max}$, a positive value that will result in a higher protein concentration over time. We therefore draw an arrow in the positive direction beginning at $[Protein] = 0$.

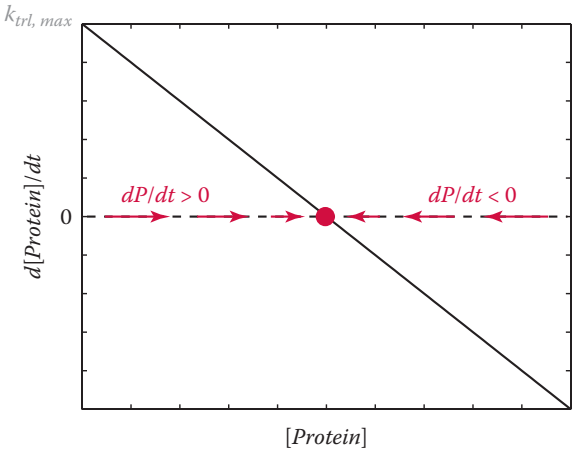


FIGURE 4.3 The vector field for the system in Figure 4.2. The arrows indicate the magnitude and direction of the change of $d[Protein]/dt$ as we approach the stable fixed point (filled circle). This vector field allows us to predict the system's response to any set of initial conditions.

If we move slightly to the right on the x axis to higher protein concentrations, we see that $d[Protein]/dt$ is still positive, but not as large as it was for $[Protein] = 0$. Accordingly, we draw another arrow, still pointing in the positive direction, but not as long (sometimes people make the arrow the same length as the value of $d[Protein]/dt$, but for now let's just look at the arrow lengths relative to each other). As we move slightly farther to the right, we find a relatively small but positive value of $d[Protein]/dt$ and draw a short, right-pointing arrow to reflect it. Finally, there is no change in protein concentration at the fixed point, so let's draw a circle around that point. The stability of fixed points is often denoted by marking stable fixed points with filled circles and unstable fixed points with empty circles, so we fill in our fixed point as well.

This next concept is important: The vector field that we've drawn is so far an equivalent way of representing the trajectory of the system represented in Figure 4.1. Beginning at $[Protein] = 0$, the rate of change is maximal, after which it steadily and linearly decreases until the protein concentration reaches a steady value of $d[Protein]/dt = 0$. This decrease in rate produces the concave-down shape of the trajectory in Figure 4.1.

Now, let's complete the vector field and examine the conditions under which the protein concentration is greater than the steady-state level, $[Protein] > [Protein]_{ss}$. Starting at the right side of the x axis, we see high values for the protein concentration, $d[Protein]/dt < 0$, so we draw arrows

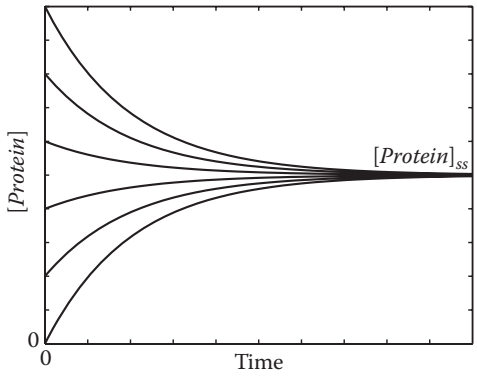


FIGURE 4.4 Sample trajectories of the system in Figure 4.2 given different initial starting conditions. All trajectories converge on the steady-state protein level, which appeared as a fixed point in Figure 4.3.

pointing to the left, and again the arrows become shorter as we approach the steady-state value until we reach the steady state.

FROM VECTOR FIELDS TO TIME-COURSE PLOTS

From this vector field, we can now predict the response of the system to any set of initial conditions. Specifically, if the system begins with [*Protein*] < $[Protein]_{ss}$, the protein concentration will increase at a steadily decreasing rate over time until it reaches the steady-state level, producing a concave-down trajectory. If the system starts at [*Protein*] > $[Protein]_{ss}$, the protein concentration will decrease at a steadily decreasing rate until it reaches that same steady-state level, producing a concave-up trajectory. Since the trajectories on either side approach the fixed point, we call it a stable state.

Some sample trajectories appear in Figure 4.4 above. I used MATLAB to draw this figure, but I want you to see that it is not necessary; likewise, you do not need the analytical solution. You could draw this figure by hand and amaze your friends with what you know about the vector field!

NONLINEARITY

In the first example, we looked at a case in which there was a linear relationship between [*Protein*] and *d*[*Protein*]/*dt*. Now, let's look at a slightly altered version of Equation 4.1, incorporating **nonlinearity** with a second-power term:

$$\frac{d[Protein]}{dt} = k_{trl,max} - k_{ploss}[Protein]^2 \tag{4.5}$$

Such an equation may be appropriate if, for example, protein dimerization were required for the complex to be degraded (these systems do exist). You can see from Equation 4.5 that if $[Protein] = 0$, then $d[Protein]/dt = k_{trl,\,max}$, just as previously indicated. The steady-state solution, however, is slightly different:

$$[Protein]_{ss} = \pm\sqrt{\frac{k_{trl,\,max}}{k_{ploss}}} \tag{4.6}$$

In reality, the protein concentration must be positive, but for learning purposes, let's consider all possible solutions of the equation. The plot of Equation 4.5 (Figure 4.5) crosses the y axis at $k_{trl,\,max}$ just as before, but now there are two fixed points, one at the positive solution of $[Protein]_{ss}$ and the other at the negative solution.

PRACTICE PROBLEM 4.1

Draw the vector field and determine the stability of each fixed point in Figure 4.5. Then, plot the protein concentration over time under the following initial conditions: (1) $[Protein]_0 = 0$; (2) $[Protein]_0 < -[Protein]_{ss}$; (3) $-[Protein]_{ss} < [Protein]_0 < 0$; (4) $0 < [Protein]_0 < +[Protein]_{ss}$; (5) $[Protein]_0 > +[Protein]_{ss}$.

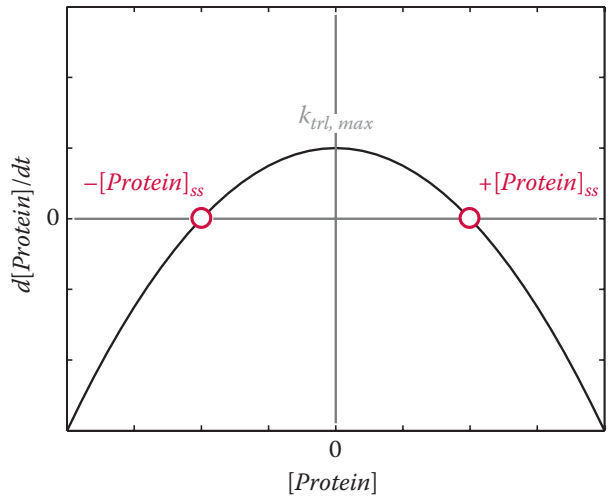


FIGURE 4.5 Plotting Equation 4.5 to visualize a nonlinear relationship between protein concentration and the change in protein concentration over time. Although two fixed points appear in the figure, protein concentration must be positive in living cells.

SOLUTION

The vector field appears in Figure 4.6. At the left side of the plot, $d[Protein]/dt$ is very negative, so the protein concentration continues to drop. The rate of change increases but remains negative as the leftmost fixed point is approached.

Just to the right of that fixed point, the rate of change is slow but positive and increases until it reaches a maximum value at $[Protein] = 0$. As $[Protein]$ becomes more positive, the rate decreases until it becomes zero at the rightmost fixed point. At values that are more positive, the rate again becomes negative, pointing toward that fixed point.

To determine the stability of each fixed point, notice that the vectors on either side of the rightmost fixed point are pointing toward that fixed point, just as we saw in Figure 4.3. This point is therefore a stable state of the system. In contrast, the vectors on either side of the leftmost fixed point are directed away from that point, resulting in an unstable state.

You can easily sketch the responses to any of the initial conditions using the vector field in Figure 4.6; some examples appear in Figure 4.7. Notice that time courses beginning near $-[Protein]_{ss}$ all move away from that fixed point; time courses beginning near $+[Protein]_{ss}$ move toward that value.

If we focus on the time courses that start at protein concentrations between 0 and $-[Protein]_{ss}$, we see that these trajectories begin by moving away from the unstable fixed point at an increasing rate as they

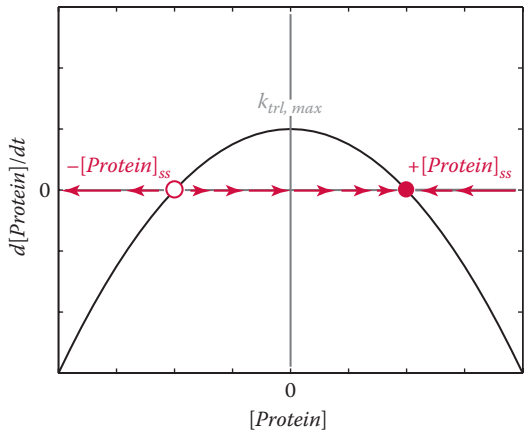


FIGURE 4.6 The vector field for the system in Practice Problem 4.1. The left fixed point is unstable, as indicated by the arrows facing away from the point.

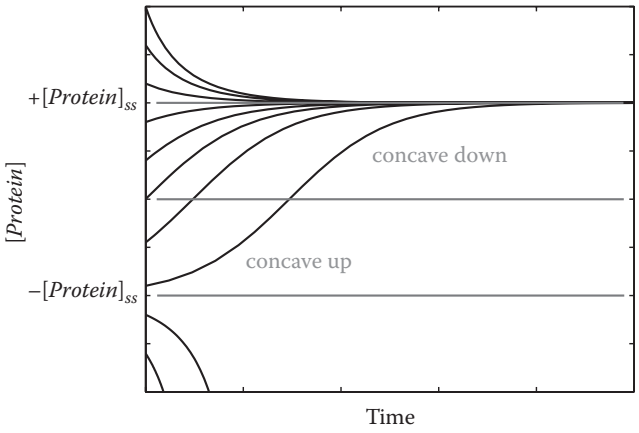


FIGURE 4.7 Responses of the system in Practice Problem 4.1, given different starting conditions. The inflection point in Figure 4.6 is indicated at [*Protein*] = 0, and the positive and negative fixed points are indicated at [*Protein*] = +/−[*Protein*]$_{ss}$.

approach [*Protein*] = 0. The shape of this curve, a slower change in [*Protein*] that increases over time, can be described as "concave up." After the time courses cross the [*Protein*] = 0 line, however, they begin moving toward the stable fixed point with a decreasing rate, resulting in a concave-down shape. The point at which the dynamics switch from concave up to concave down is called an **inflection point**.

BIFURCATION ANALYSIS

Changing the values of key parameters in an ODE can have a dramatic effect on the dynamics of your system, and graphical methods can powerfully illustrate these effects. Practice Problem 4.2 gives a pertinent example.

PRACTICE PROBLEM 4.2

Repeat the steps performed in Practice Problem 4.1 for $k_{trl,\,max} < 0$ and $k_{trl,\,max} = 0$. Draw vector fields, identify and characterize the fixed points, and sketch time courses for representative initial conditions. Don't use MATLAB until you've tried it by hand! Do either of these values of $k_{trl,\,max}$ have any biological relevance?

SOLUTION

When $k_{trl,\,max}$ is negative (Figure 4.8, top), there are no fixed points, but [*Protein*] = 0 remains an inflection point where d[*Protein*]/dt switches from decreasing in magnitude to increasing in magnitude.

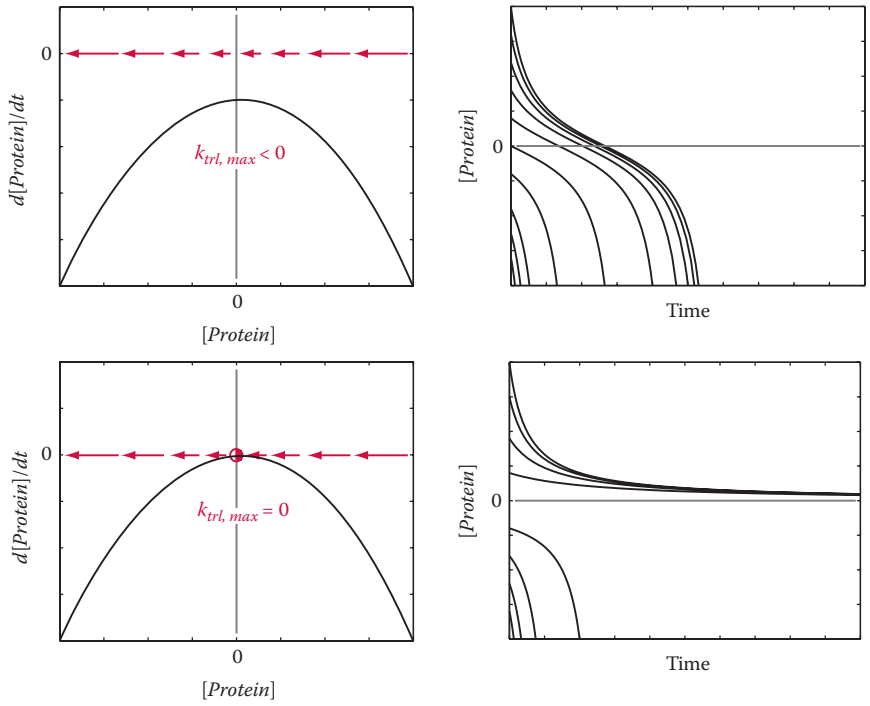


FIGURE 4.8 The value of $k_{trl,\,max}$ influences the number of fixed points (Practice Problem 4.2). Top, when $k_{trl,\,max}$ is negative, there are no fixed points. Bottom, when $k_{trl,\,max} = 0$, a half-stable point occurs at [*Protein*] = 0.

When $k_{trl,\,max} = 0$ (corresponding to complete repression of gene expression; Figure 4.8, bottom), there is only one fixed point; this point is unusual because the vector field on the left side of the fixed point is directed away, while the vector field on the right is oriented toward the fixed point. This point is therefore a half-stable fixed point because any time course with initial conditions of $[Protein]_0 > 0$ will approach a steady state of [*Protein*] = 0; initial conditions of $[Protein]_0 < 0$ lead to rapidly decreasing protein concentrations.

A most interesting conclusion can therefore be drawn from our analysis: The number of fixed points, a critical determinant of system dynamics, depends on the value of $k_{trl,\,max}$! Figure 4.9a summarizes the vector fields that we produced in Figures 4.6–4.8. When $k_{trl,\,max}$ is large and positive, the fixed points, one unstable and one stable, are located at a certain distance that becomes increasingly smaller as $k_{trl,\,max}$ approaches zero. At $k_{trl,\,max} = 0$, the two fixed points fuse into one half-stable fixed point, which disappears at negative values of $k_{trl,\,max}$.

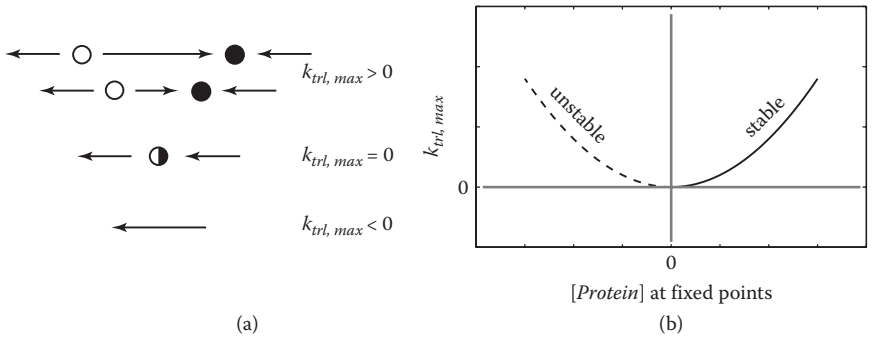


FIGURE 4.9 Summary of the observations in Practice Problem 4.2. (a) Schematic of the changes in the vector field for different values of $k_{trl,\,max}$. Stable fixed points appear as filled circles, unstable fixed points are empty circles, and the half-stable fixed point is half shaded. (b) Using a bifurcation diagram to visualize the effect of $k_{trl,\,max}$ on protein concentration at the fixed points.

Figure 4.9b (plotted from Equation 4.6) depicts the relationship between $k_{trl,\,max}$ and the fixed points in more detail. A change in the number of fixed points, caused by a change in parameter values, is called a **bifurcation**, and the plot in Figure 4.9b is called a **bifurcation diagram**. There are many types of bifurcations; this case illustrates a **saddle-node bifurcation**.

With regard to relevance, it is biologically impossible to have a $k_{trl,\,max}$ less than zero. But sometimes, for example in the case of the inducible promoter that we considered in Chapter 3, transcription can be halted, and therefore $k_{trl,\,max} = 0$.

To summarize what we have seen so far in this chapter, graphical analysis enables us to quickly visualize the results of changing initial conditions or parameters, much like our previous Boolean analysis, but in more detail.

ADDING FEEDBACK

Now that we have addressed the system without feedback described in Equation 4.1, let's move to the system with feedback shown in Equation 4.2. The protein production term in this equation is substantially more complicated than in Equation 4.1, making the entire equation somewhat more difficult to plot without the aid of a computational tool.

Fortunately, there is an easier way to generate our vector field: Instead of plotting the complete equation for $d[Protein]/dt$, we can plot each term (the production rate v_{prod} and the loss rate v_{loss}) on the right side of the equation separately and compare those two plots.

Comparing the magnitude of the protein production (Figure 4.10a, gray) and loss (Figure 4.10a, black) rates yields the vector field (Figure 4.10b). At low protein concentrations, protein production is substantially greater than protein loss, so we draw an arrow toward higher protein concentrations (pointing to the right). Moving rightward, the difference between the production and loss rates shrinks; thus, the arrows pointing right are shorter. High protein concentrations lead to a loss rate that is higher than the production rate, so the vectors at high concentrations point to the left. Finally, there is a single point at which the production rate is equal to the loss rate; therefore, $d[Protein]/dt = 0$. This point is fixed, and because the vectors in the field all point toward this point, we know that it is a stable fixed point.

Let's think about whether changing some of our parameters would lead to a bifurcation. Figure 4.11 illustrates the outcome of changing k_{ploss}, $k_{trl,\,max}$, and K individually. In each case, the protein production and loss rates only meet at a single point, and as a result, changing these parameters will not lead to a bifurcation. In order for these rates to meet at multiple points, one or the other terms would need significantly more curvature, or nonlinearity. For example, cooperativity introduces nonlinearity into a system, which therefore may lead to a bifurcation.

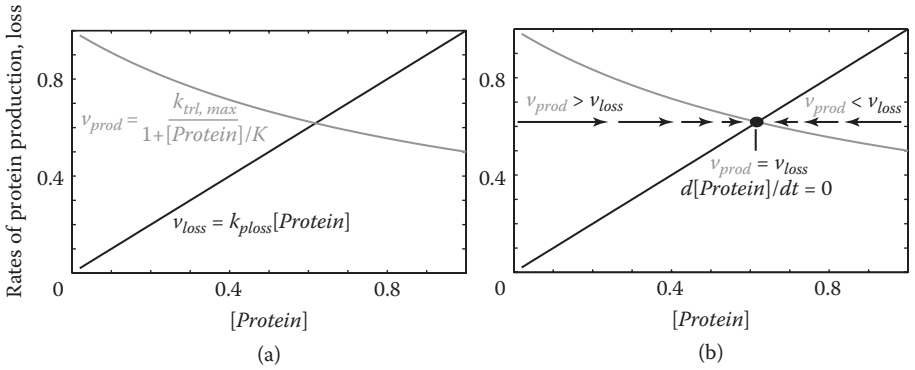


FIGURE 4.10 Graphing the dynamics of the system (with feedback) in Equation 4.2. (a) Protein production (gray) and loss (black) in the system. (b) The vector field reveals a single stable fixed point.

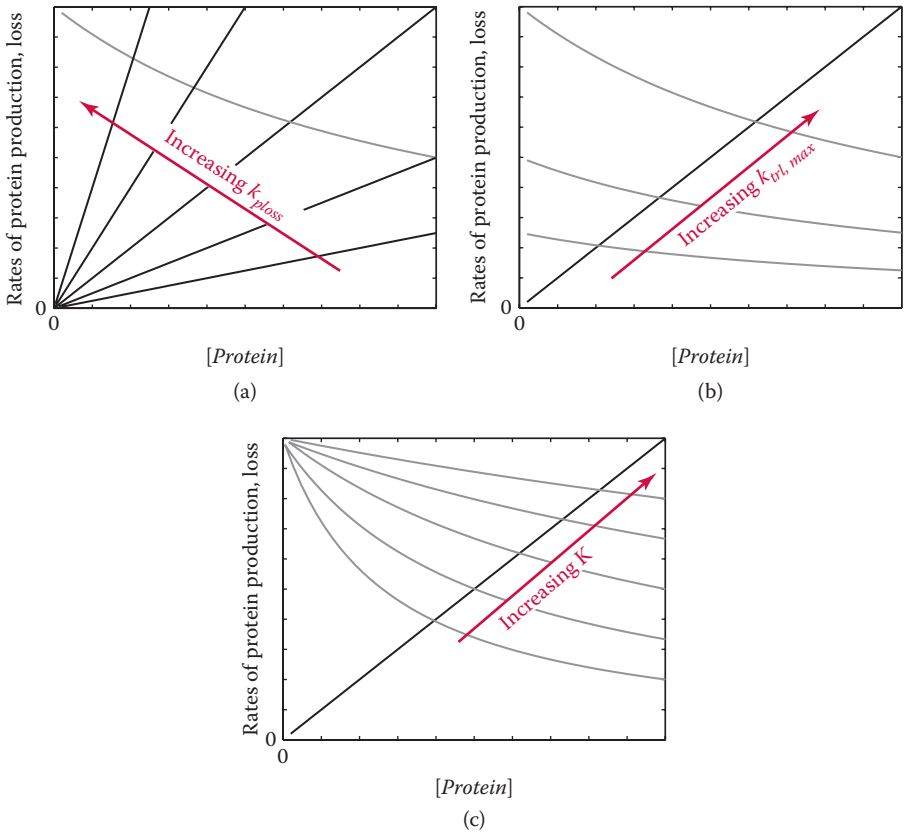


FIGURE 4.11 The effects of changing k_{ploss} (a), $k_{trl,\,max}$ (b), or K (c) in Equation 4.2 on the position of the fixed point. Protein production is shown in gray, and protein loss appears in black. The point at which the gray and black lines intersect is the fixed point of the system. Notice that none of these changes leads to a bifurcation; there is only one fixed point in each of the intersections shown.

TWO-EQUATION SYSTEMS

To this point, our focus has been on graphical analysis of single equations. However, our original system depended on two equations, one for protein, and another for mRNA:

$$\frac{d[Protein]}{dt} = k_{trl}[mRNA] - k_{ploss}[Protein] \tag{4.7}$$

$$\frac{d[mRNA]}{dt} = k_{trs,\,max}\left(\frac{1}{1+[Protein]/K}\right) - k_{mloss}[mRNA] \tag{4.8}$$

In Chapter 3, we assumed that the mRNA concentration reaches a steady state much more quickly than the protein concentration, which enabled us to solve Equation 4.8 for $d[mRNA]_{ss}/dt = 0$ and then to substitute the resulting value of $[mRNA]_{ss}$ into Equation 4.7 to obtain Equation 4.2. In this case, we want to solve both equations together without making any additional assumptions; graphical analysis will be helpful.

We have two variables to worry about instead of one, but otherwise, our approach will be similar to our approach for single equations. We start by considering all the steady states together and drawing **nullclines**, which are the steady-state solutions of each equation in the system. As usual, we set $d[Protein]/dt$ and $d[mRNA]/dt$ equal to zero and solve for $mRNA$ in terms of $Protein$:

$$[mRNA]_{ss} = \frac{k_{ploss}}{k_{trl}}[Protein]_{ss} \tag{4.9}$$

$$[mRNA]_{ss} = \frac{k_{trs}}{k_{mloss}}\left(\frac{1}{1+[Protein]_{ss}/K}\right) \tag{4.10}$$

Figure 4.12 contains plots of both of these lines for $K = k_{ploss}/k_{trl} = k_{trs}/k_{mloss} = 1$. This plot looks a lot like Figure 4.10, and for good reason, since they depict the same equations. This should not surprise you because our second system is a more complicated version of our first system.

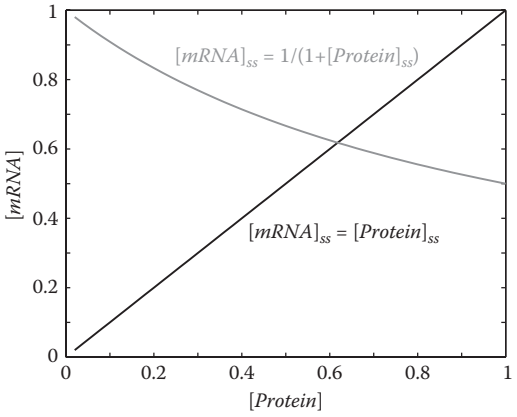


FIGURE 4.12 Including the change in mRNA production in our analysis of the system with feedback. In this two-dimensional case, the black and gray lines are nullclines. Also, notice the similarity to Figure 4.10a.

Notice that there is only one point of intersection between the nullclines, and we identify those coordinates (0.62, 0.62) by solving Equations 4.9 and 4.10. This location reflects the point at which the protein and mRNA levels, and therefore the system as a whole, are at a steady state.

Again, we draw the vector field to determine how the system approaches that steady state. Let's start with four individual vectors at each corner of the plot in Figure 4.12; the first vector will be the origin (0, 0). If we substitute these values for *Protein* and *mRNA* into Equations 4.7 and 4.8, we obtain:

$$\frac{d[Protein]}{dt} = [mRNA] - [Protein] = 0 - 0 = 0 \tag{4.11}$$

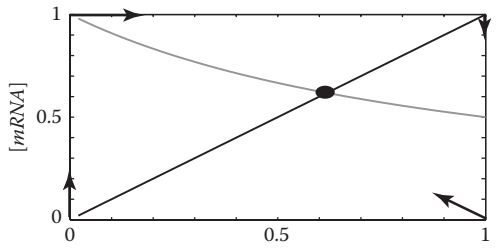
$$\frac{d[mRNA]}{dt} = \left(\frac{1}{1+[Protein]}\right) - [mRNA] = 1 - 0 = 1 \tag{4.12}$$

As a result, we draw a vector that points upward in the *y* direction. Using the same approach, we identify the three other corners of the plot in Figure 4.13a and draw the four vectors in Figure 4.13b.

You see that it is fairly easy to calculate what *d*[*Protein*]/*dt* and *d*[*mRNA*]/*dt* should be, given any initial conditions. Even with this small

	$\frac{d[Protein]}{dt}$	$\frac{d[mRNA]}{dt}$
$P = 0, M = 0$	0	1
$P = 0, M = 1$	1	0
$P = 1, M = 1$	0	−0.5
$P = 1, M = 0$	−1	0.5

(a)



(b)

FIGURE 4.13 Computing (a) and visualizing (b) the vector field for our system with feedback. Note that the outside vectors do not point directly at the fixed point. The vector lengths shown here are scaled down 1:4 to facilitate visualization. P, [*Protein*]; M, [*mRNA*].

beginning, you may notice a few things, for example that these outside vectors do not point directly at the fixed point, and that there are times when only the protein concentration or only the mRNA concentration changes as the immediate response to a set of initial conditions.

A computational package such as MATLAB makes these calculations even easier, and you will learn how to perform some of these calculations in the problems at the end of the chapter. A much more complete vector field, generated using MATLAB, appears in Figure 4.14a.

Notice that the vectors seem to spiral toward the fixed point, allowing us to conclude that the fixed point is stable. For further evidence, we can plot specific trajectories on the vector field using the same four initial conditions

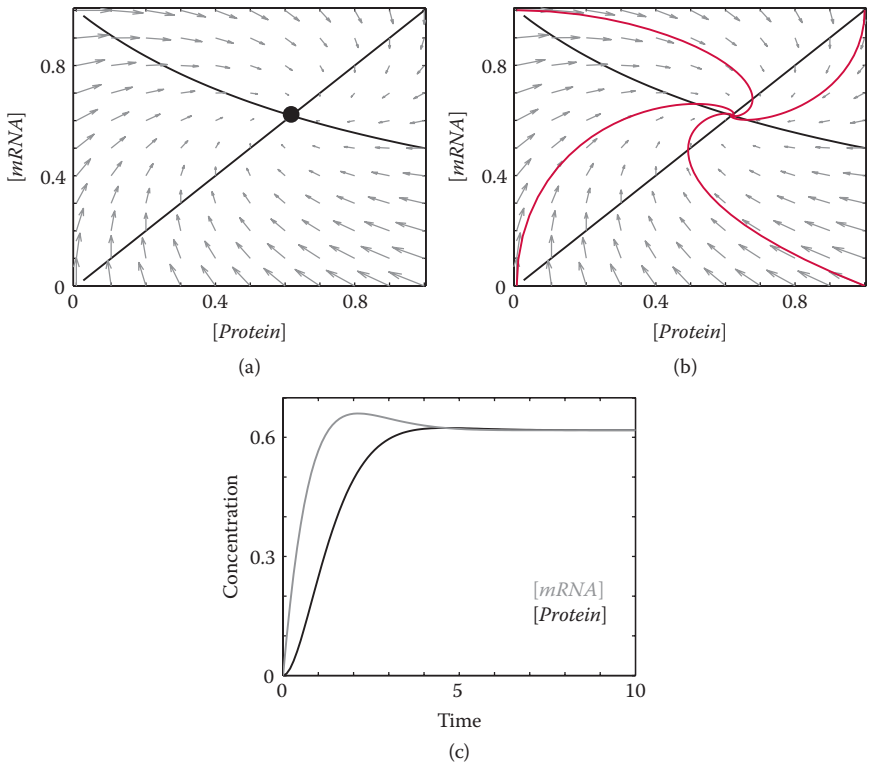


FIGURE 4.14 Visualizing the changes in mRNA and protein concentration in our system with feedback. (a) The complete vector field. (b) Plotting trajectories (red) for the four initial conditions in Figure 4.13 demonstrates that the system moves toward the stable fixed point. (c) The dynamic responses of the mRNA (gray) and protein (black) concentrations over time beginning at conditions (0, 0). Notice that [*mRNA*] overshoots the stable state. The MATLAB functions `meshgrid` and `quiver` were used to create the vector fields in (a) and (b).

that we considered in Figure 4.13b, for example. These trajectories (also calculated by solving the ODEs numerically in MATLAB) indeed all move toward the stable fixed point in Figure 4.14b.

However, they do not move directly; all four trajectories are curved in Figure 4.14b. In some cases, the trajectory even passes the fixed-point value of either [*mRNA*] or [*Protein*] before reaching a steady state at both values.

These interesting aspects of the trajectories carry over to the time courses. Figure 4.14c depicts the dynamic responses of [*mRNA*] and [*Protein*] given initial conditions with both concentrations equal to zero. The corresponding red trajectory in Figure 4.14b begins at the origin and moves up and to the right before reaching the stable fixed point. The mRNA concentration rises quickly, overshoots the steady-state value, and then returns to it; the protein concentration undergoes a more simple rise to the steady-state value.

Other trajectories lead to different time courses, but you can see from the vector field that overshoot commonly occurs by one variable or the other. These time courses were drawn using MATLAB, but the real key to drawing them is the vector field.

I hope you can see that graphical analysis has several nice features. First, we exploit the specificity and detail that accompany ODEs, but like the Boolean approach, we are able to examine many responses simultaneously to obtain an idea of how the system works in general. We can also relax some of the assumptions required to make analytical solutions of ODEs tractable; note, for example, that the results in Figure 4.14c are inconsistent with our previous assumption (made largely for reasons of expediency) that the mRNA concentration reaches a steady state *before* the protein concentration. Finally, we can see how parameter changes affect system dynamics.

The challenge with these approaches is that it is somewhat difficult to visualize information in multidimensional space. As a result, the graphical approach is powerful for two- or three-dimensional systems, but not for higher systems. If our system is small enough, however, or if we can make a realistic reduction in the complexity of our system, it is hard to beat graphical approaches for the intuition they can give us.

CHAPTER SUMMARY

Graphical analysis can lead us to deeper intuition about ODEs by enabling us to identify stable and unstable states, to plot dynamics, and to consider the effects of varying initial conditions and parameter values, rapidly and without any need for a computer. We began with a single ODE system,

plotting the values of a given variable (some variable X) against the change in that variable over time (dX/dt). The fixed points of the system are defined as the x intercepts of the plot of X versus dX/dt. The stability of the fixed points was determined by calculating a one-dimensional vector field in which the direction of the vector was in the positive direction when $dX/dt > 0$ and in the negative direction when $dX/dt < 0$. The magnitude of the vector is simply the absolute value of dX/dt (although these vectors were scaled down in the figures to make them easier to visualize).

The vector field indicates in which direction and at what speed a given value of X will change. Stable fixed points are those in which the vectors on both sides point toward them, and unstable fixed points are surrounded by vectors pointing away from them; there are also half-stable points that are stable on one side and unstable on the other. We can use the dX/dt plots, together with the fixed points and vector field, to quickly sketch the time-course plots for X for any initial conditions.

We also discussed how the fixed points and vector field change for different ODEs; in particular, we saw that nonlinearity was required for a single ODE to have multiple fixed points. Changing the values of parameters in an ODE can also lead to changes in the fixed points and vector field; the study of these changes, in particular when parameter value changes lead to loss or gain of a fixed point, is called bifurcation analysis.

Graphical analysis can be adapted for more complicated single-ODE systems by plotting the positive terms in the ODE separately from the negative terms and using the difference between these two plots to create the vector field. This analysis can be expanded to two-ODE systems. The nullclines of the system, one line for each ODE in which dX/dt is set to zero and the result is plotted, are plotted together with a two-dimensional vector field. The fixed points in this case are where the nullclines intersect.

The main weakness of graphical analysis is that it can only be applied to small systems. However, if you have or can obtain a relatively small (one- or two-dimensional) model of your circuit, it can be a powerful way to build understanding.

RECOMMENDED READING

Ellner, S. and Guckenheimer, J. *Dynamic Models in Biology*. Princeton, NJ: Princeton University Press, 2006.

Gardner, T. S., Cantor, C. R., and Collins, J. J. Construction of a genetic toggle switch in *Escherichia coli*. *Nature* 2000, **403**(6767): 339–342.

Huang, D., Holtz, W. J., and Maharbiz, M. M. A genetic bistable switch utilizing nonlinear protein degradation. *Journal of Biological Engineering* 2012, **6**:9–22.

Strogatz, S. H. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering (Studies in Nonlinearity)*. Boulder, CO: Westview Press, 2001.

PROBLEMS

PROBLEM 4.1
A Simple Oscillatory System

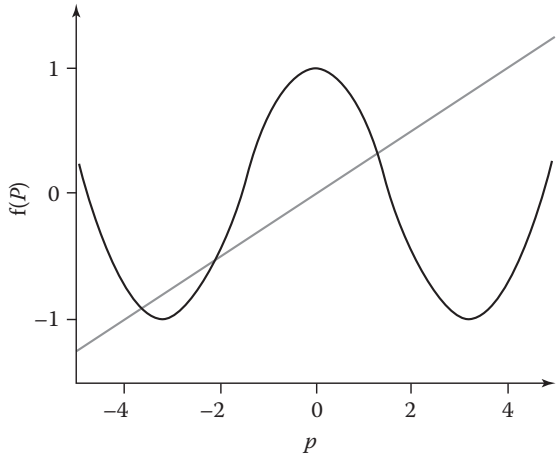
This problem is more illustrative than biological in order to build your skills. Consider a protein whose concentration over time can be described with the following ODE:

$$\frac{dp}{dt} = a \cdot p - \cos(p)$$

where a is a constant equal to 0.25, p is the protein concentration, and t is time. As you learned, we can plot the two terms on the right side of this equation separately to determine fixed points and to generate a vector field. In other words, we can plot $f(p)$ versus protein concentration, where:

$$f(p) = \begin{Bmatrix} 0.25 \cdot p \\ \cos(p) \end{Bmatrix}$$

The resulting plot shows the upper term in gray and the lower term in black.



a. By hand, draw the vector field describing the change in protein concentration. Make sure to note both direction and magnitude in your vector field.

b. Circle all of the steady states, if applicable. Fill in the steady state(s) that are stable.

c. By hand, sketch the changes in protein concentration over time for at least five initial concentrations of protein, including 3.5, 1.0, –1.8, –3.2, and –4.5.

d. What happens to the fixed points as you increase the value of a? You may use graphs to illustrate your answer.

PROBLEM 4.2
An Autoregulatory Circuit with Positive and Cooperative Feedback

In Chapter 3, we considered circuits that contain positive feedback. We also considered the effects of cooperativity, for example when multiple copies of the protein were required to interact at the DNA promoter for the feedback to be effective. Here, we consider positive and cooperative feedback together, as shown in the figure below.

In this case, a protein dimer (Hill coefficient $H = 2$) is required to bind to DNA for transcription to occur. The ODEs describing this system are:

$$\frac{d[Protein]}{dt} = k_{trl} \cdot [mRNA] - k_{ploss} \cdot [Protein]$$

$$\frac{d[mRNA]}{dt} = \frac{k_{trs,\,max} \cdot [Protein]^2}{\left(\left(\frac{k_d}{k_a}\right)^2 + [Protein]^2\right)} - k_{mloss} \cdot [mRNA]$$

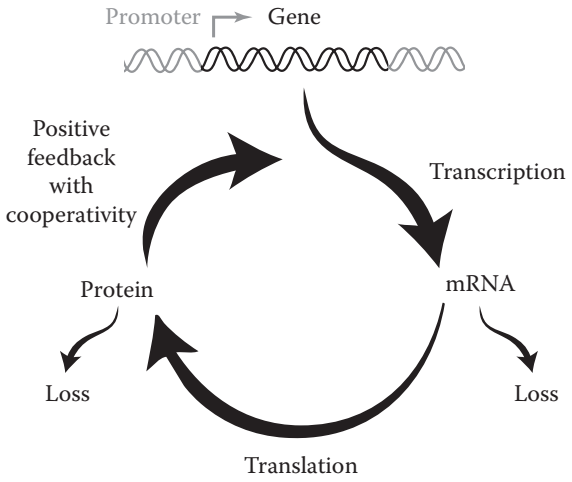
For the following questions, let $k_{trl} = k_{trs,\,max} = k_d = k_a = 1$; $k_{ploss} = 0.02$; and $k_{mloss} = 20$. Don't worry about units for now (they've got plenty of time to break your heart).

a. Use MATLAB to plot the nullclines of the equations in this problem. Hints: Let mRNA be your y axis and protein be your x axis; let your protein concentration range from 0 to 3.

b. How many fixed points does this system have? Circle them on your graph from (a).

c. Using your graph from (a), draw the vector field for several points of your choice by hand. Which of the fixed points are stable?

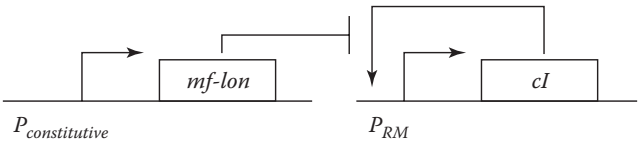
d. Explain what the stable points mean in terms of the biology of the system. How can changes in the parameter k_{ploss} lead to different numbers of fixed points? Provide at least two examples.



PROBLEM 4.3
A Single Feedback Loop System with Enzymatic Degradation

We have mostly focused on production terms in our ODEs when considering the regulation of gene expression. However, protein degradation rates are also important in regulatory circuits and can be critical for the regulation of bistable networks. In this problem, we look at the use of dynamic protein degradation to create a synthetic bistable switch, using the Lon protease found in *Mesoplasma florum* (*mf*-Lon).

In the hybrid switch (described in Huang, Holtz, and Maharbiz, 2012), the positive autoregulatory loop is identical to the one found in lambda phage, where promoter P_{RM} expresses the lambda repressor cI, which is also a transcriptional activator of P_{RM}. Bistable behavior may be possible using only this single positive-feedback loop along with constitutive expression of *mf*-Lon, which quickly degrades cI. However, the Michaelis constant for *mf*-Lon would need to be significantly smaller than the value reported by Huang et al. for bistable behavior to occur, as you will see. We will analyze and model this simplified circuit in three ways to uncover its important dynamic properties.



The equations that govern the system are:

$$\frac{d[m_{cI}]}{dt} = k_{trs,\,max}\left(\frac{1}{1+\left(\frac{K_{Diss}}{[p_{cI}]}\right)^{H}} + l_{P_{RM}}\right) - k_{mloss}[m_{cI}]$$

$$\frac{d[p_{cI}]}{dt} = k_{trl}[m_{cI}] - k_{ploss}[p_{cI}] - k_{cat}[p_L]\left(\frac{1}{1+\frac{K_M}{[p_{cI}]}}\right)$$

where H equals the Hill coefficient of the system, K_{Diss} is the dissociation constant of cI from P_{RM}, l_{PRM} is the fraction of mRNA transcription that is caused by leakage from the P_{RM} promoter (the basal expression level), k_{cat} is the catalytic rate of *mf*-Lon, p_L is the protein concentration of *mf*-Lon, and K_M is the Michaelis constant for *mf*-Lon (Michaelis constants are discussed in more detail in Chapter 9). As described in Problem 3.4, when multiple repressor binding sites occur in front of a gene, the Hill coefficient describes the cooperative binding properties of that system (for two binding sites, $H = 2$, etc.).

To start, let's set:

$$k_{trs,\,max} = 1.35 \times 10^{-9}\ \text{M}^2\ \text{s}^{-1}$$

$$K_{diss} = 2.5 \times 10^{-8}\ \text{M}$$

$$H = 1$$

$$l_{PRM} = 0.1$$

$$k_{mloss} = 2.38 \times 10^{-3}\ \text{s}^{-1}$$

$$k_{trl} = 5 \times 10^{-5}\ \text{s}^{-1}$$

$$k_{ploss} = 2 \times 10^{-4}\ \text{s}^{-1}$$

$$k_{cat} = 0.071\ \text{s}^{-1}$$

$$p_L = 1 \times 10^{-10}\ \text{M}$$

$$K_M = 3.7 \times 10^{-9}\ \text{M}$$

a. List at least two of the assumptions we've made to produce the two equations that describe the system.

b. In Problem 3.4 in Chapter 3, the protein production rate was proportional to:

$$\frac{1}{1+\left(\frac{[Protein]}{K_{Diss}}\right)^{H}}$$

The production term in our ODE for m_{cI} is different in that K_{Diss} is in the numerator and p_{cI} is in the denominator. Where do these changes come from? Show mathematically how this term was derived.

c. What does K_M mean in biochemical terms in this system? Explain why each of the variables appears the way it does in the term describing the enzymatic degradation of cI by *mf*-Lon.

d. Plot the nullclines of these two equations in MATLAB. Circle any fixed points.

e. Plot the vector field using the MATLAB commands `quiver` and `meshgrid`. Are the fixed points you identified stable or unstable?

f. So far, we have assumed that $H = 1$. In fact, the P_{RM} promoter has two adjacent operator sites; binding of one cI protein to the upstream "helper" operator helps recruit another cI protein to the "activator" operator, which when bound stimulates transcription. Thus, positive cooperativity occurs in this system, and $H = 2$ for P_{RM}. Plot the new nullclines in MATLAB and circle the fixed points for this scenario.

g. Plot a vector field on your graph from (f) and determine which of the fixed points are stable. What do these fixed points mean for the biology of the system?

h. Now, vary the value of K_M. How does varying the value of K_M change the stability of the system? Use the MATLAB `subplot` command to add these new plots to one figure to obtain a single figure that allows visual inspection of the effect of changing K_M.

PROBLEM 4.4
A Dual-Repressor System

Here we return to the toggle switch from Problem 2.2 (Gardner, Cantor, and Collins, 2000), which is diagrammed below.

The equations that govern the system are:

$$\frac{d[Repressor_1]}{dt} = k_{trl} \cdot [mRNA_1] - k_{ploss} \cdot [Repressor_1]$$

$$\frac{d[mRNA_1]}{dt} = \frac{k_{trs,\,max1}}{\left(\left(\frac{[Repressor_2]}{K_{Diss2}^2}\right)^H + 1\right)} - k_{mloss} \cdot [mRNA_1]$$

$$\frac{d[Repressor_2]}{dt} = k_{trl} \cdot [mRNA_2] - k_{p,\,loss} \cdot [Repressor_2]$$

$$\frac{d[mRNA_2]}{dt} = \frac{k_{trs,\,max2}}{\left(\left(\frac{[Repressor_1]}{K_{Diss1}^2}\right)^H + 1\right)} - k_{mloss} \cdot [mRNA_2]$$

where $k_{ploss} = k_{mloss} = k_{trl} = 1/s$; $k_{trs,\,max1} = 3/s$; $k_{trs,\,max2} = 4/s$; $K_{Diss1} = 0.6\ \mu M$; $K_{Diss2} = 0.8\ \mu M$; and $H = 1$.

a. For both repressors, assume that [*mRNA*] quickly reaches a steady state with respect to [*Protein*]. Use this assumption to derive a new, reduced set of equations for the change in protein concentrations over time.

b. In MATLAB, plot the nullclines of the two equations you found in (a).

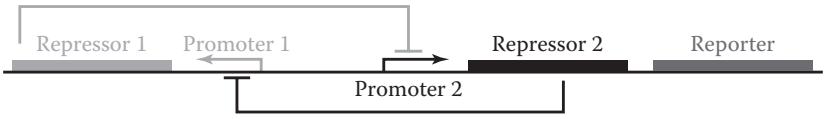
c. Plot a vector field on the graph for (b) and use it to determine the stability of any fixed points you identified.

d. Now let's analyze what would happen if we added an additional DNA binding site to the promoter for each repressor, such that $H = 2$. Plot the new nullclines in MATLAB and circle any fixed points.

e. Draw a vector field on the graph from (d) and determine which of the fixed points are stable.

f. Explain what these fixed points mean for the biology of the system.

g. Now vary K_{Diss} for one of the repressors. How does this change affect the stability of the system?



CHAPTER 5

Variation: Numerical Integration

LEARNING OBJECTIVES

- Calculate the numerical solutions to sets of ODEs using the Euler, midpoint, and Runge–Kutta methods
- Approximate the error of these methods using a Taylor series expansion
- Understand the sensitivity of each method to the time-step size and how this sensitivity relates to the method order

We conclude our exploration of methods for solving ODEs by discussing numerical methods. Even a few years ago, I would have been tempted to cover this entire chapter with two short sentences:

> Open MATLAB on your computer.
>
> Type "`help ode45`" at the command line.

In other words, the methods I am about to describe have already been implemented in MATLAB, so why do you need to learn them? The first reason is that you will be better able to use the MATLAB functions if you really understand how they work and what their limitations are. That is not the most important reason, however. I want you

to learn these methods so that you can "roll your own" versions of the methods, adapting them to your own needs as necessary. In particular, your ability to alter and, when necessary, subvert these numerical solving techniques will be critical to creating **hybrid models** that combine ODEs with non-ODE-type methods. I come back to this point in Chapter 10.

THE EULER METHOD

Numerical integration of ODEs depends on a fairly straightforward problem: Given a function $f(x) = dx/dt$ and values for x and t at a particular instant (I'll call them x_0 and t_0, respectively), determine the value of x at a later time point (x_1 and t_1). If you recognize that $dx/dt = f(x)$ is the form of an ODE, then you can probably also see that our protein and mRNA equations from Chapters 3 and 4 can be solved using these same approaches.

Not to drag this book back into the gutter, but let's revisit the analogy I developed in Chapter 3 (Figure 5.1, a refresher, was originally Figure 3.2). Remember, I was estimating the amount of water in my rain gutter over time by measuring the rates of rainfall and drainage using this mass balance equation:

$$GW(t + \Delta t) = GW(t) + Rain(t) \cdot \Delta t - Drain(t) \cdot \Delta t \tag{5.1}$$

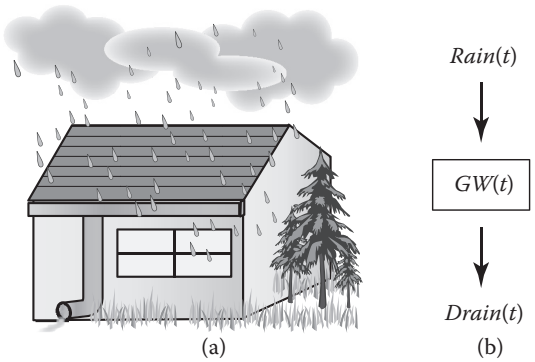


FIGURE 5.1 Conceptualizing ODEs with a rain gutter analogy (See also Figure 3.2). (a) During a rainstorm, water enters the gutter at rate *Rain*, flows out of the drainpipe at rate *Drain*, or builds up in the gutter to an amount *GW*. (b) The compartment diagram for the analogy indicates that water only enters the gutter from the rain and only exits the gutter from the drain; the diagram also captures the dependence on time of these processes.

I then derived an ODE from this equation:

$$\frac{dGW}{dt} = Rain(t) - Drain(t) \tag{5.2}$$

We can cast the central problem of this chapter in terms of this rain gutter equation by recognizing that GW is analogous to x, $Rain(t) - Drain(t)$ to $f(x)$, t in Equation 5.1 to t_0, and $(t + \Delta t)$ to t_1. In that context, let's rearrange Equation 5.1:

$$GW(t + \Delta t) = GW(t) + (Rain(t) - Drain(t)) \cdot \Delta t \tag{5.3}$$

In other words, we can numerically predict the amount of water in the gutter at a future time if we know the present amount of water, the present rates of rainfall and drainage, and the interval between the present and future time. Now, let us apply our analogy to approximate x_1 based on what we know about x_0:

$$x_1 = x_0 + f(x_0) \cdot \Delta t \tag{5.4}$$

To get a feeling for what Equation 5.4 really means, take a look at Figure 5.2. To calculate x_1 using Equation (5.4), you draw a line extending from, and tangent to, the point (x_0, t_0). You then approximate x_1 as the

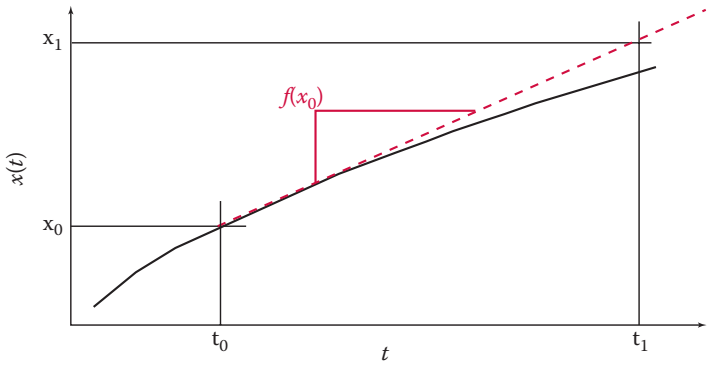


FIGURE 5.2 The Euler method. By drawing a line tangent to our curve (red dashed line) at point (x_0, t_0), we can approximate the value of point (x_1, t_1) from the slope of the tangent line. This approximation becomes less valid as the curve diverges from the tangent, so we want to take the smallest step possible between t_0 and t_1 to generate the most accurate value for x_1. Note that when $t_0 = 0$, Equation 5.4 is the equation of a line with intercept x_0 and slope $f(x_0)$.

value of x where the tangent line intersects with $t = t_1$. In fact, Equation 5.4 is the equation of a line, and if t_0 is set as the origin, x_0 is the intercept and $f(x_0)$ is the slope.

The method that outlined here is called the **Euler method**. You should note two things about applying the Euler method. First, you can see from Figure 5.2 that the accuracy of the approximation depends on the magnitude of Δt. When t_1 is far enough from t_0 (in Figure 5.2, about halfway between t_0 and t_1), the approximation starts to diverge significantly from the actual solution. The second thing to notice is that our function f is evaluated at x_0 when we use the Euler method. I elaborate on how to choose a useful time step, as well as other ways to evaluate f, in further discussion.

Now that we have established how to implement the Euler method for a single time step, it will not be too difficult to describe the numerical solution for a large number of time steps. I will use a MATLAB-esque pseudocode here:

```
(1) Define simulation timespan, starting at (t0, x0),
    and divide the timespan into N steps such that
    deltaT = timespan/N is sufficiently small.
(2) Define two arrays to hold the time and
    concentration data: tarr = [t0]; xarr = [x0];
(3) for i = 1:N
                tnew = tarr(i) + deltaT;
                xnew = xarr(i) + f(xarr(i))* deltaT;
                tarr = [tarr tnew];
                xarr = [xarr xnew];
     end
(4) plot(tarr, xarr)
```

What it means to be "sufficiently small" is addressed in the next section.

ACCURACY AND ERROR

Let's apply the Euler method to a familiar example to explore how it works. Revisit the equation most recently seen as Equation 4.1 (and previously in Equations 3.17 and 3.18): the simple expression of protein without feedback. The two kinetics constants in this equation govern the rate of transcription and the rate of decay. To make the analysis relatively straightforward, let's set both of these rates equal to 1, obtaining the following equation:

$$\frac{d[Protein]}{dt} = 1 - [Protein] \tag{5.5}$$

We determined the general solution to this equation analytically in Equation 3.25; with the constants equal to 1 and initial conditions of (0,0), this solution simplifies to:

$$[Protein](t) = 1 - e^{-t} \tag{5.6}$$

The analytical solution to Equation 5.6 is plotted in Figure 5.3. Now let's try using the Euler approximation. We start with the same initial conditions. Our function $f(Protein)$ is determined from Equation 5.5: $f(Protein) = 1 - Protein$. If we try a time step of 0.01, we can calculate the value of *Protein* at time = 0.01 as:

$$[Protein]_{(t=0.01)} = [Protein]_0 + 0.01 \cdot (1 - [Protein]_0) \tag{5.7}$$

which evaluates to $0 + 0.01 \cdot 1 = 0.01$. Interestingly, if we substitute $t = 0.01$ into Equation 5.6, we also obtain ~0.01. So, for the first time step at least, the Euler method provides a good approximation. We follow the same procedure to determine the rest of the solution, and it turns out that the line from the Euler method completely overlaps the line from the analytical solution (Figure 5.4).

So far, so good: Apparently, a time step of 0.01 is sufficiently small to give us a good approximation in this case. How did I know to choose 0.01 as the time step? I just tried a few possibilities, and the fit that I saw

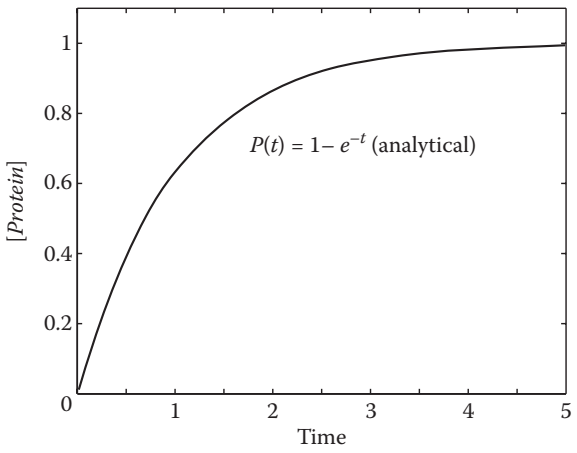


FIGURE 5.3 The analytical solution for our simple protein system. The ODE (Equation 5.6) describes simple protein expression and decay, with the rate constants of both set to 1, without feedback.

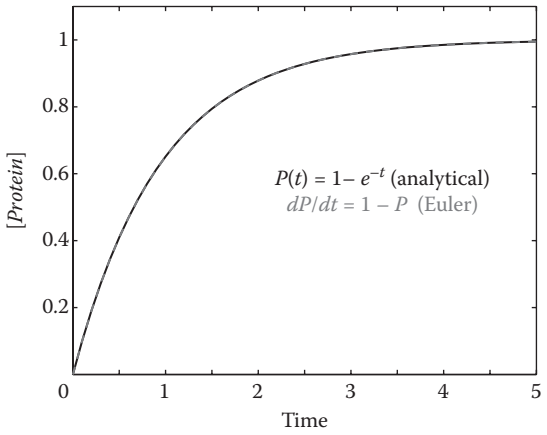


FIGURE 5.4 The Euler method for $t = 0.01$ provides an accurate approximation for Equation 5.6. Here, the line we generated using the Euler method (dashed gray) completely overlaps the line from the analytical solution (Figure 5.3; black line). A time step of $\Delta t = 0.01$ is therefore sufficiently small to return a reasonable approximation via the Euler method.

in Figure 5.4 looked good to me! I can also show you what some of my other attempts looked like. For example, what happens when we move to a bigger time step? The answer appears in Figure 5.5, which plots the Euler approximation of Equation 5.5 for increasingly large time steps. Notice that the approximation quickly becomes inaccurate as the time step is increased.

It would be nice to dig a little deeper with this observation, especially to examine ways to describe the accuracy of a given method and time step. Two ways to look at the error are discussed; the first is specific to our example, and the second is more generalizable.

First, let's look specifically at our example. For this case, we can define an error by determining the **Euclidean distance** between the analytical solution and the Euler approximation at time = 1, 2, 3, 4, and 5 as follows:

$$error = \sum_{i=1}^{5} \left([Protein]_{analytical,\, t=i} - [Protein]_{Euler,\, t=i} \right)^2. \qquad (5.8)$$

Table 5.1 shows the error (Equation 5.8) for four time step lengths. As you saw in Figure 5.5, the error increases substantially with time step length, and the increase is nonlinear: As the time step increases from 0.01 to 0.1, the error increases by 10 fold, but with the increase from 0.1 to 1.0, the error increases by 15 fold.

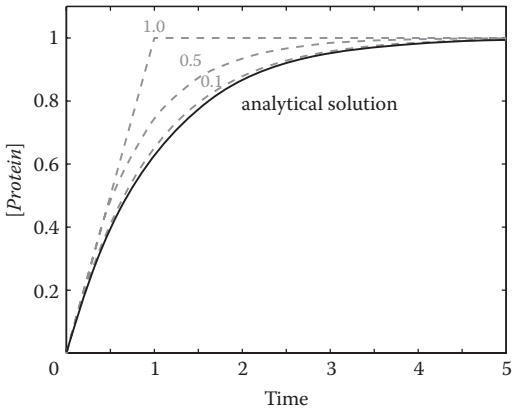


FIGURE 5.5 Increasing the size of the time step generates less accurate approximations via the Euler method. Time steps of Δt = 0.1, 0.5, and 1.0 (dashed gray lines) lead to larger and larger deviations from the curve of the analytical solution to Equation 5.6 (black line). Recall that a time step of Δt = 0.01 did not deviate from the analytical solution in Figure 5.4.

TABLE 5.1 Error Introduced by the Euler Approximation at Various Time Steps, Calculated Using Equation 5.8

Time Step	Error
0.01	0.0045
0.1	0.0455
0.5	0.2450
1.0	0.5781

Equation 5.8 is admittedly a bit artificial (we cannot apply it to compare numerical techniques for every equation we want to solve), so we will also use a more general approach for determining error. One way involves the use of a **Taylor series approximation**. We will not discuss the details of the approach here, but briefly, if you have a function S defined in terms of some variable y and want to approximate the value of $S(y)$, when y is near a value a for which $S(a)$ is known, the Taylor series approximation is:

$$S(y) \approx S(a) + \frac{S'(a)}{1!}(y-a) + \frac{S''(a)}{2!}(y-a)^2 + \frac{S'''(a)}{3!}(y-a)^3 + \ldots + \frac{S^n(a)}{n!}(y-a)^n \tag{5.9}$$

where $S'(a) = dS/dy$, evaluated at a. For this approximation to work, S must be "infinitely differentiable" (S', S'', and so on), even though in practice we only use the first handful of terms in the approximation to simplify the calculation.

The Taylor series approximation gives us a second, independent numerical method (the first method in this case is the Euler method itself) to predict future values of a function. Another way to calculate error, particularly when the analytical solution is not known, would therefore be to compare approximations from a variety of methods. For example, we could approximate the "error" of the Euler method as the difference between values calculated using it and the Taylor series approximation.

Referring to Equation 5.4, recall that we determined a new value of x (x_1) in terms of a previous value x_0, a function that described the time dynamics $f(x)$ and the time step Δt. If we substitute $x(t)$ for $S(y)$, let $t_1 = t_0 + \Delta t$, and set $a = t_0$, Equation 5.9 becomes:

$$x(t_1)_{Taylor} \approx x(t_0) + \frac{1}{1!}\frac{dx}{dt}\bigg|_{x(t_0)}(t_0 + \Delta t - t_0) + \frac{1}{2!}\frac{d^2x}{dt^2}\bigg|_{x(t_0)} (t_0 + \Delta t - t_0)^2 + \ldots \tag{5.10}$$

Simplification leads to:

$$x(t)_{Taylor} \approx x_0 + \Delta t \cdot f(x_0) + \tfrac{1}{2}\Delta t^2 \cdot f'(x_0) + \ldots \tag{5.11}$$

Now let's define the error of the Euler method simply as the absolute difference between the values of x_1 determined using the Euler method and using the Taylor series approximation:

$$error_{Euler} = \left| x_{1,Taylor} - x_{1,Euler} \right| \tag{5.12}$$

Substituting Equations 5.4 and 5.11 and ignoring terms past the second derivative in Equation 5.11 (multiplying these terms by higher powers of Δt makes these terms relatively small), we obtain:

$$error_{Euler} = \tfrac{1}{2} f'(x_0)\Delta t^2 \tag{5.13}$$

Note that the $f'(x_0)$ term is a constant because it is evaluated at a specific value of x (x_0). Equation 5.13 therefore reduces to:

$$error_{Euler} \propto \Delta t^2 \tag{5.14}$$

In plain English, the error depends on the square of the time step length, as we noticed in Figure 5.5 and Table 5.1. The Euler method is called "first order" (the **order** of the method is the power of Δt in Equation 5.14 minus 1) because of this dependence.

THE MIDPOINT METHOD

The Euler method is useful for teaching the concepts of numerical integration, but with real-world problems like ours for which error is a concern, we will need higher-order methods. "Higher order" means that the error depends on a higher power of Δt, so higher-order methods are generally more accurate at a given Δt. Let's consider a second-order approach called the midpoint method (Figure 5.6).

The left panel of Figure 5.6 is an illustration of the Euler method that is similar to Figure 5.2 except the curve is a little more dramatic so that you can see the error more clearly. As you can see, what's causing the error in the Euler method is that big slope, which in the example is larger at x_0 than at any later time point. If we approximated x_1 using a smaller slope

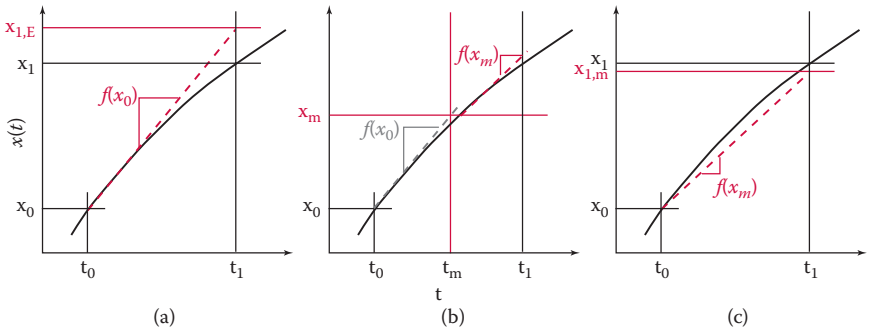


FIGURE 5.6 Illustration of the midpoint method to approximate the value of x_1. Left, application of the Euler method leads to an assignment of $x_{1,E}$ based on the slope of the tangent to $f(x)$ at x_0, denoted as $f(x_0)$. The error of the Euler method versus the analytical solution is $x_1 - x_{1,E}$, which is large here. Middle, use of the Euler method to approximate x_m (Equation 5.16). Right, solving Equation 5.15 with x_m and $f(x_m)$ yields a value of x_1 that is too small, but with smaller error than that obtained by Euler's method alone.

determined from elsewhere in the curve, it could yield a more accurate calculation.

The midpoint method is simple in principle: Determine the slope at the midpoint and use that slope to calculate x_1. First, we modify Equation 5.4 to include the midpoint:

$$x_{1,m} = x_0 + f(x_m) \cdot \Delta t \tag{5.15}$$

where $x_{1,m}$ is the approximation of x_1 calculated using the midpoint method, and x_m is the value of x halfway across the time interval (at time t_m, equal to $(t_0 + t_1)/2$).

Wait a minute, you're thinking (I hope)—if we do not know x_1, how do we know x_m? Well, we don't know x_m! We can only approximate it, and the tool that we have is … the Euler method! x_m is therefore calculated as:

$$x_m = x_0 + f(x_0) \cdot \frac{\Delta t}{2} \tag{5.16}$$

The calculation of x_m is also shown in the middle panel of Figure 5.6. Notice that the Euler method predicts a value for x_m that is also too large. As a result, $f(x_m)$ will not actually be the slope of the curve at t_m but at a higher value of t, which in this case has a smaller slope. Once x_m and $f(x_m)$ are determined, they can be used to solve Equation 5.15 to yield $x_{1,m}$ (Figure 5.6, right). Alternatively, Equations 5.15 and 5.16 can be combined into a single equation:

$$x_{1,m} = x_0 + f\left(x_0 + \tfrac{1}{2} f(x_0)\Delta t\right) \cdot \Delta t \tag{5.17}$$

Figure 5.7 illustrates how the midpoint and Euler methods compare over a larger span of the curve from the analytical solution. Note that in our case, the Euler method overpredicts x_1, while the midpoint method underpredicts it. This effect is caused by the difference in slope discussed earlier. Using the Taylor series approximation, we can estimate the error of the midpoint method as before. The error turns out to be proportional to Δt^3, so the midpoint method is second order. For Figure 5.7, the error (Equation 5.8) is 0.3907 for the midpoint method, which is 60% of the Euler method error at $\Delta t = 1$.

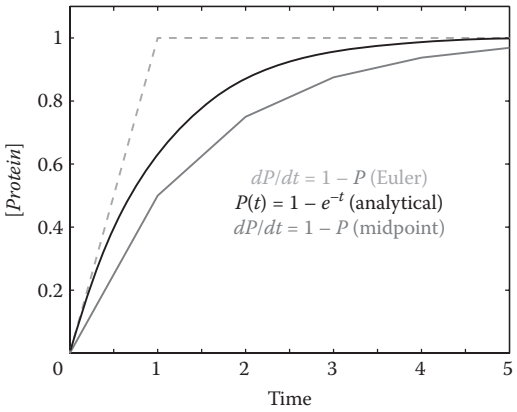


FIGURE 5.7 The midpoint method (dark gray) versus the Euler method (dashed light gray) and the analytical solution (black) for $\Delta t = 1$. (meaning that only the points {1, 2, 3, 4, 5} were calculated using the midpoint and Euler methods; the lines connecting the points were added by MATLAB).

PRACTICE PROBLEM 5.1

Given the equation:

$$\frac{dp}{dt} = \frac{p^3 + 2p}{3}$$

and that $p(0) = 2$, $t(0) = 0$, and $\Delta t = 0.5$, use the midpoint method to find $p(t = 0.5)$. Show your work.

SOLUTION

First we find p_m using the Euler method and Equation 5.16:

$$p_m = p_0 + f(p_0) \cdot \frac{\Delta t}{2}$$

$$p_m = 2 + \left(\frac{2^3 + 2 \cdot 2}{3} \right) \cdot \frac{0.5}{2} = 3$$

Next, we substitute the value for p_m into Equation 5.15:

$$p_{1,m} = p_0 + f(p_m) \cdot \Delta t$$

$$p_{1,m} = 2 + \left(\frac{3^3 + 2 \cdot 3}{3} \right) \cdot 0.5 = 7.5$$

PRACTICE PROBLEM 5.2

The transition from one to two equations can be particularly difficult for those who are new to numerical integration. Using the midpoint method, numerically integrate the following two equations:

$$\frac{dm}{dt} = p$$

$$\frac{dp}{dt} = 2m$$

given that $m(0) = 4$, $p(0) = 2$, $t(0) = 0$, and $\Delta t = 1$.

SOLUTION

First, find m_m and p_m. The equations are:

$$m_m = m_0 + f_m(m_0, p_0) \cdot \frac{\Delta t}{2}$$

$$p_m = p_0 + f_p(m_0, p_0) \cdot \frac{\Delta t}{2}$$

where $f_m = p$, and $f_p = 2m$. Substitution yields:

$$m_m = m_0 + (p_0) \cdot \frac{\Delta t}{2}$$

$$m_m = 4 + 2 \cdot \frac{1}{2} = 5$$

$$p_m = p_0 + (2m_0) \cdot \frac{\Delta t}{2}$$

$$p_m = 2 + (2 \cdot 4) \cdot \frac{1}{2} = 6$$

Next, use m_m and p_m to find m_1 and p_1:

$$m_{1,m} = m_0 + f_m(m_m, p_m) \cdot \Delta t$$

$$m_{1,m} = 4 + (6) = 10$$

$$p_{1,m} = p_0 + f_p(m_m, p_m) \cdot \Delta t$$

$$p_{1,m} = 2 + (2 \cdot 5) = 12$$

THE RUNGE–KUTTA METHOD

We can carry the principles of the midpoint and Euler methods still further using a higher-order approximation. Carl Runge and Martin Kutta, of the Technische Hochschule Hannover and the Rheinisch-Westfälische Technische Hochschule Aachen, respectively, developed a robust implementation of this approach, which is named for them. As we have seen, the error in both the Euler and midpoint methods essentially depends on an imperfect approximation of the slope (Figure 5.6). The Runge–Kutta method addresses this problem using a weighted average of slopes.

The first slope, which we will call f_1, is the slope calculated using the Euler method:

$$f_1 = f(x_0) \tag{5.18}$$

In other words, f_1 is simply the slope of the curve at x_0 (Figure 5.8, left). To simplify the equations, let's define a set of variables k as $k_i = \Delta t \cdot f_i$. The second slope is then calculated using the following equation:

$$f_2 = f\left(x_0 + \tfrac{1}{2}k_1\right) \tag{5.19}$$

If you compare Equation 5.19 with Equation 5.17, you will see that the second slope is the slope that would be calculated using the midpoint

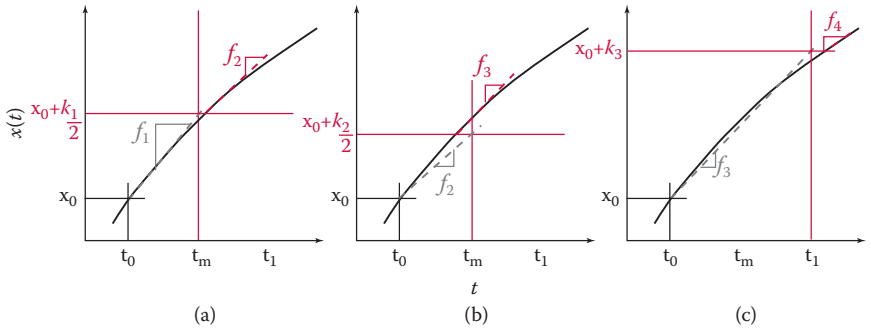


FIGURE 5.8 Calculation of the four slopes for implementation of the Runge–Kutta method. Left, f_1 is the slope of the curve at x_0; f_2 is the slope that would have been calculated using the midpoint method in Figure 5.6. Middle, f_3 is also calculated via the midpoint method, but with f_2 in place of f_1. Right, f_4 is based on the estimated end point. These four calculated slopes are weighted and averaged to estimate x_1.

method (Figure 5.8, left). The third slope is also calculated via the midpoint method (Figure 5.8, middle), with the exception that the second slope is used in the calculation in place of the first:

$$f_3 = f\left(x_0 + \tfrac{1}{2}k_2\right) \tag{5.20}$$

Finally, the fourth slope is not determined at the beginning of the time interval, or at estimated midpoints, but at an estimated end point (Figure 5.8, right):

$$f_4 = f(x_0 + k_3) \tag{5.21}$$

The four slopes are weighted and averaged to calculate the new value of x_1:

$$x_{1,RK} = x_0 + \tfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{5.22}$$

Figure 5.9 depicts the analytical solution for Equation 5.5, together with the three numerical methods discussed in this chapter. In this figure, the time step was equal to 1, and so only six points were calculated for each numerical method (your comparison of the methods in Figure 5.9 should focus on the points, not the lines between the points). You should see that for these six points, the Runge–Kutta solution

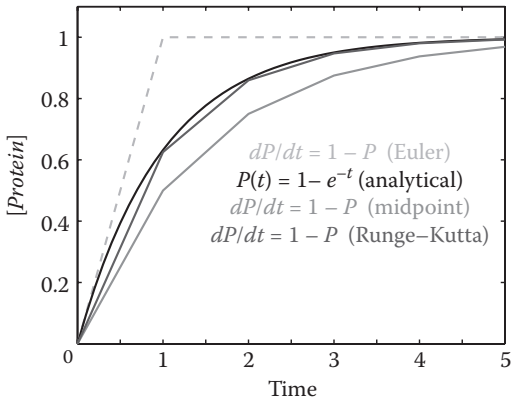


FIGURE 5.9 Comparison of the analytical solution (black) with the Euler (dashed light gray), midpoint (gray), and Runge–Kutta methods (dark gray) for $\Delta t = 1$. The Runge–Kutta method generates a curve that is extremely similar to the analytical solution.

TABLE 5.2 Values of k Calculated Using the Runge–Kutta Method

i	$k_i = \Delta t \cdot f(x_{i-1})$
1	$1 - 0 = 1$
2	$1 - 1 \cdot ½ = ½$
3	$1 - ½ \cdot ½ = ¾$
4	$1 - ¾ = ¼$

is nearly indistinguishable from the analytical solution; the other numerical methods are clearly different.

To help illustrate the Runge–Kutta method more clearly, let's calculate the value of [*Protein*] at $t = 1$, $[Protein]_0 = 0$, and $f([Protein]) = 1 - [Protein]$, as given in Equation 5.5. With $\Delta t = 1$, we can simply consider the four slopes in Table 5.2 using Equations 5.18–5.21.

With these values, x_1 is determined using Equation 5.22 as $0 + 1/6 \cdot (1 + 1 + 6/4 + 1/4) = 5/8$ or 0.625. For comparison, the value of $x_{t=1}$ determined analytically is ~0.632. Using Equation 5.8 to estimate the error of the Runge–Kutta method for this particular case yields a value of 0.0175, which is 33-fold more accurate than the Euler method and over 20-fold more accurate than the midpoint method. Furthermore, the Taylor series approximation approach to determining error reveals that the Runge–Kutta method error is proportional to Δt^5.

The Runge–Kutta method is therefore fourth order and in fact, is the `4` in MATLAB's `ode45` function. The `5` comes from adaptively controlling the step size, which is a nice addition but often unnecessary. For most uses, the standard Runge–Kutta method has stood the test of time.

You now have all three major tools for modeling and understanding ODEs in your tool belt—congratulations! These numerical approaches are particularly important because the size of the system you can study increases dramatically. But why did I force you to learn something that MATLAB can easily do for you? The answer is that knowing how the algorithm works will give you the power to change it when necessary. Without this ability, my lab would never have been able to create a model of a whole cell, for example, because we wanted to integrate ODE-based approaches with other approaches as far-flung as Boolean logic, linear optimization, and stochastic methods (Chapters 9 and 10). This kind of integration is discussed further, and you will see that your familiarity with these numerical methods will be critical.

CHAPTER SUMMARY

Numerical integration is the most common way to solve sets of ODEs, in particular when more than a few ODEs need to be solved simultaneously. We first considered the Euler method; we began with a set of initial conditions and then used the instantaneous rate of change at those conditions to estimate the new conditions. For a one-dimensional ODE, we showed how the instantaneous rate of change gives us the slope of a line, which we multiplied by the time step length and added to the initial value (at t_0) to obtain the next value (at t_1). The Euler method is highly sensitive to the step size (the error is supposed to be proportional to the square of the time step, as we determined by comparison to a Taylor series expansion), so other methods have been devised that are associated with a smaller error. The Euler method is called a first-order method because the error is proportional to the square of the time step.

The midpoint method uses the Euler method to estimate the variable value at t_m (halfway between t_0 and t_1) and to determine the instantaneous rate of change at that value to generate a slope. The midpoint method is called second order because its error is estimated to be proportional to the cube of the time step.

The Runge–Kutta method builds on the Euler and midpoint methods by calculating a weighted average of four slopes (the Euler and midpoint slopes, together with a second estimate of the instantaneous rate of change at t_m based on the midpoint, and an estimate of the instantaneous rate of change at t_1 based on the third slope). The error associated with the Runge–Kutta method is proportional to the fifth power of the time step, so this method is called fourth order. This method is a commonly used numerical integrator and has been implemented in MATLAB (with a variable step size) as `ode45`. Even though this function is simple to implement, it is important for us to learn how numerical integration works so that we can adapt the methods to our own purposes as necessary.

RECOMMENDED READING

Ellner, S. and Guckenheimer, J. *Dynamic Models in Biology*. Princeton, NJ: Princeton University Press, 2006.

Moore, H. *MATLAB for Engineers* (3rd edition). Englewood Cliffs, NJ: Prentice Hall, 2011.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. *Numerical Recipes Third Edition: The Art of Scientific Computing*. Cambridge, UK: Cambridge University Press, 2007.

PROBLEMS

PROBLEM 5.1
A Simple ODE

Problem 5.1 is more illustrative than biological in order to build your skills. Consider the differential equation

$$\frac{dp}{dt} = p^3$$

with the initial condition $p_0 = 0.5$ at $t = 0$.

a. Calculate the value of $p(t = 1\text{ s})$ by hand using the Euler, midpoint, and Runge–Kutta methods.

b. Now, solve the equation analytically. How does the analytical solution compare with your answers from (a)?

c. Write MATLAB code to simulate the dynamics of p over time using the Runge–Kutta method. Simulate 1 s with $\Delta t = 0.01$ s, beginning with $p_0 = 0.5$. Plot your result. Did the smaller time step make the Runge–Kutta calculation more accurate?

PROBLEM 5.2
Two Simple ODEs

Using the Runge–Kutta method, numerically integrate the following two equations:

$$\frac{dm}{dt} = p$$

$$\frac{dp}{dt} = 2m$$

given that $m(0) = 4$, $p(0) = 2$, $t(0) = 0$, and $\Delta t = 1$.

a. Calculate by hand the values of $m(t = 1)$ and $p(t = 1)$. Show your work.

b. Write MATLAB code to simulate the dynamics of m and p over time for 100 time steps with $\Delta t = 0.1$. Don't use `ode45`, of course! Plot your result.

c. At a party, a crowd of people watches from behind as you code your own numerical integrator. Under that pressure, you forget how to code a Runge–Kutta integrator and are forced to use the Euler method. What parameter would you change to make your integrator more accurate? What negative consequences would that have compared to higher-order methods?

PROBLEM 5.3
Our Favorite Autoregulatory, Negative-Feedback System

Recall that the system we have been studying throughout these chapters is governed by two ODEs (Equations 4.7 and 4.8):

$$\frac{d[Protein]}{dt} = k_{trl}[mRNA] - k_{ploss}[Protein]$$

$$\frac{d[mRNA]}{dt} = k_{trs,max}\left(\frac{1}{1+[Protein]/K}\right) - k_{mloss}[mRNA]$$

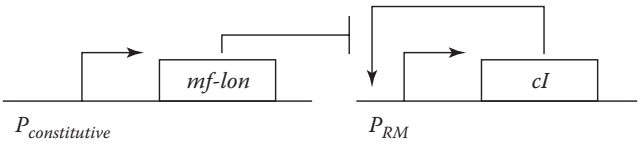
Let $K = k_{ploss} = k_{trl} = k_{trs,max} = k_{mloss} = 1$ and start with $[Protein]_0 = [mRNA]_0 = 0$.

a. Write three numerical integrators to solve this pair of ODEs using the Euler, midpoint, and Runge–Kutta methods. Use $\Delta t = 0.01$ and integrate from $t = 0$ to $t = 10$. Plot $[mRNA]$ and $[Protein]$ on the same plot for each integrator.

b. Now, run the same simulations at larger time steps, $\Delta t = 0.1$, 0.5, 1.0, and 5.0. How do your integrations compare with one another at these different time steps?

PROBLEM 5.4
A Single Feedback Loop System with Enzymatic Degradation

Problem 4.3 introduced you to the use of dynamic protein degradation to create a bistable switch using the Lon protease found in *Mesoplasma florum* (*mf*-Lon). As mentioned in Chapter 4, the positive autoregulatory loop is identical to the one found in lambda phage, for which promoter P_{RM} expresses the lambda repressor cI, which is also a transcriptional activator of P_{RM}.



The equations that govern the system depicted in the figure are:

$$\frac{d[m_{cI}]}{dt} = k_{trs,\,max}\left(\frac{1}{1+\left(\frac{K_{Diss}}{[p_{cI}]}\right)^{H}} + l_{P_{RM}}\right) - k_{mloss}[m_{cI}]$$

$$\frac{d[p_{cI}]}{dt} = k_{trl}[m_{cI}] - k_{ploss}[p_{cI}] - k_{cat}[p_L]\left(\frac{1}{1+\frac{K_M}{[p_{cI}]}}\right)$$

where H equals the Hill coefficient of the system, K_{Diss} is the dissociation constant of cI from P_{RM}, l_{PRM} is the fraction of mRNA transcription from the P_{RM} promoter that is caused by leakage (basal expression), k_{cat} is the catalytic rate of *mf*-Lon, p_L is the protein concentration of *mf*-Lon, and K_M is the Michaelis constant for *mf*-Lon (Michaelis constants are discussed in more detail in Chapter 9). As described in Problem 3.4, when multiple repressor binding sites occur in front of a gene, the Hill coefficient describes the cooperative binding properties of that system (for two binding sites, $H = 2$, etc.).

As before, let's use the following parameter values:

$$k_{trs,\,max} = 1.35 \times 10^{-9}\ \mathrm{M^2\ s^{-1}}$$

$$K_{diss} = 2.5 \times 10^{-8}\ \mathrm{M}$$

$$l_{PRM} = 0.1$$

$$k_{mloss} = 2.38 \times 10^{-3}\ \mathrm{s^{-1}}$$

$$k_{trl} = 5 \times 10^{-5}\ \mathrm{s^{-1}}$$

$$k_{ploss} = 2 \times 10^{-4}\ \mathrm{s^{-1}}$$

$$k_{cat} = 0.071 \text{ s}^{-1}$$

$$p_L = 1 \times 10^{-10} \text{ M}$$

$$K_M = 3.7 \times 10^{-9} \text{ M}$$

$$H = 2$$

Let the initial amounts of cI mRNA be 200 nM and cI protein be 50 nM.

a. Calculate the steady-state quantities of cI mRNA and protein analytically.

b. Using the midpoint method, find the concentrations of mRNA and protein at $t = 0.1$, using $\Delta t = 0.1$. Show your work.

c. Now build a Runge–Kutta numerical integrator in MATLAB to solve the ODEs. Generate plots of mRNA and protein concentrations over time. How do your results compare to your answer for (a)?

d. Plot the nullclines for the two ODEs (hint: you may have already done this as part of Problem 4.3). Now, on the same nullcline graph, plot the paths taken by m_{cI} and p_{cI}. Make sure you run enough time steps to visualize the behavior you expect! You will find the `hold` command to be useful here.

e. Change the initial conditions as follows, plotting the concentrations of cI mRNA and protein over time and also on the nullcline plot: Try $[m_{cI}] = 70$ nM and $[p_{cI}] = 10$ nM, and then $[m_{cI}] = 151.53875097$ nM and $[p_{cI}] = 11.2000778972$ nM (yes, I chose these specific numbers on purpose). Are the results what you expect? Again, you will find the `subplot` command useful here.