Chapter 10
The Network Within: Signaling Pathways

UPINDER S. BHALLA

10.1 Introduction

In the preceding chapters, we have taken the building blocks for neuronal models in the form of ion channels, membrane compartments and synapses (Chapters 4, 5 and 6) and put them together to form neurons, small neural circuits, and then networks (Chapters 7, 8 and 9). We conclude this section of the book by opening the lid on the building blocks, and seeing what happens inside them. There used to be a view of the atom as a tiny solar system, which led to exotic visions of an infinite series of ever-decreasing worlds as one examined matter at finer and finer scales. Neurobiology now has to face a similar unsettling concept: that hidden within "atomic" compartments and under all the ion channels, there exists a whole new universe of subcellular networks. These networks, of course, are the multitudes of interacting biochemical signaling pathways. They profoundly affect everything from the properties of single channels to the morphology of neurons and the wiring of the brain itself. Recent work on biochemical signaling reactions has emphasized the sheer complexity of these networks. There are dozens of known major signaling pathways, each having at least five to ten enzyme isoforms, each of which communicates with a different subset of other messengers and pathways. As with neuronal networks themselves, these biochemical networks cannot be analyzed in isolation. Neuronal signaling events from above, and nuclear and metabolic events from below, provide a barrage of signals that this network must process.

What role do signaling pathways play in the life of a cell? In a word: everything. Metabolism, cytoskeleton formation, gene regulation, response to external inputs, differentiation, cell cycle, synaptic computation — all these operations are in the domain of signaling pathways. Consider a large industrial chemical complex. The control system for such a system is typically a huge computer, with hundreds of control points and sensors. A living cell is a chemical system whose complexity is orders of magnitude beyond anything man-made, and its control systems are correspondingly complex. Although a lot of the "software" is encoded in DNA, the role of "hardware" (more properly, "wetware") is carried out by biochemical pathways.

The goal of this chapter is to introduce you to biochemical computation, with particular emphasis on aspects important for neurobiology. These include gating and modulation of ion channels, signaling by diffusible and retrograde messengers, and the myriad processes involved in long term potentiation (LTP). After a brief introduction to a few major pathways, the chapter covers some of the theory underlying biochemical models. The second half of the chapter describes the specifics of building such models using GENESIS and *Kinetikit*.

10.1.1 Nomenclature

A *signaling pathway*, in the sense I employ it, is any series of reactions that manipulate information of importance to a cell. This definition is rather broad, and might, for instance, be considered to include metabolic regulatory pathways that do not have much to do with the outside world. We are mostly concerned with pathways that do involve interaction with external events, particularly neuronal events.

It is sometimes difficult to decide where one pathway ends and another begins. I will use the term *pathway* to describe a group of reactions influencing a single signaling enzyme. In this sense, then, the classical cyclic AMP pathway shown in Fig. 10.1 (Gilman 1987) consists of at least four pathways: the receptor-ligand pathway, the G protein activation pathway, the adenylyl cyclase pathway, and the protein kinase A activation pathway. Another way of looking at it is to consider a pathway as a node at which information can converge, be processed, and sent on to multiple other nodes.

The term *second messenger* is typically used for a small signaling molecule (i.e., not a protein) that is produced indirectly through the action of some primary messenger such as a neurotransmitter. Examples are cAMP (cyclic adenosine monophosphate), DAG (diacylglycerol), IP_3 (inositol 1,4,5 triphosphate), AA (arachidonic acid), and, of course, Ca (calcium). Given the proliferation of interactions among signaling pathways, it is sometimes difficult to decide if a given molecule is a second, third, or *n*th messenger. We will apply the term to most small non-protein signaling molecules.

The term *signal transduction* will be applied in a restrictive manner to receptors that change or transduce one signaling modality (such as light) to another, such as biochemical

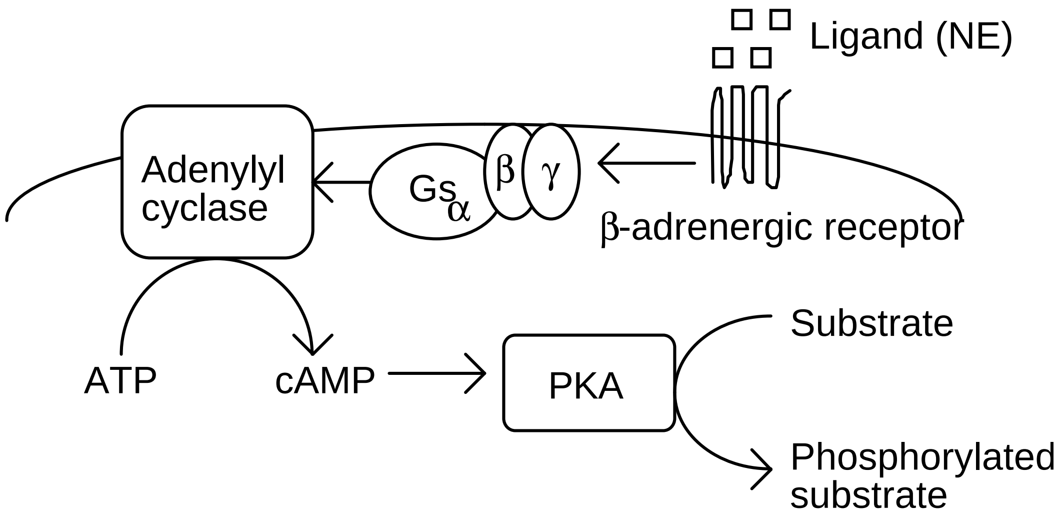


Figure 10.1 The cyclic AMP pathway. In this chapter we treat it as four successive pathways: (1) activation of the β-adrenergic receptor, (2) activation of the G protein G_s, (3) activation of adenylyl cyclase, and (4) activation of protein kinase A. This pathway was one of the first to be examined in detail, and incorporates many of the interactions subsequently found in other pathways.

signals. In this sense a Ca channel is a sort of signal transduction mechanism, but the more specific term *ion channel* will be used in this case. *Receptors* will be used to describe proteins that detect neurotransmitters, usually at a synapse.

10.1.2 A Short Short Course in Biochemistry

This little section is meant for non-biologists and should be skipped by anyone who has survived a modern biology course. Any of the standard textbooks on biochemistry or molecular biology of the cell are recommended reading (e.g., Alberts, Bray, Lewis, Raff, Roberts and Watson 1994, Kandel, Schwartz and Jessel 1991).

DNA (deoxyribonucleic acid) is the genetic material of the cell. It encodes information in the sequence of four chemical building blocks called *bases*. These are labeled A, T, C and G.

A *protein* is a chain of amino acids (another building block). There are 20 amino acids. The sequence of amino acids is specified by DNA, and in turn determines the three-dimensional structure and biochemical properties of the protein. Proteins frequently associate in pairs to form *dimers*, or triplets (*trimers*), or larger numbers (*multimers*).

An *isoform* of a protein is another protein, with a different but usually closely related sequence. Isoforms usually have similar enzymatic properties, but may be regulated differently.

An *enzyme* is a protein with catalytic activity. It speeds up reactions. Enzymes usually exhibit great specificity in their *substrates*, the molecules they act upon, and also in the *products* they generate.

ATP (adenosine triphosphate) is the major source of chemical energy in the cell. The removal of one or two phosphates from ATP releases energy. This reaction is called *hydrolysis*. *GTP* (guanosine triphosphate) is a chemical cousin of ATP, and shares its energetic properties but is not involved in as many reactions in the cell.

A *protein kinase* is an enzyme that adds a phosphate from ATP to an amino acid on a protein. This process is called *phosphorylation*. The added phosphates frequently modify the activity of the target protein, which makes this a process of great importance for signaling.

A *phosphatase* reverses the action of a kinase: it removes the phosphate group.

Buffering is a way of holding the free concentration of a given molecule close to a desired set point by a suitable combination of chemical sources and sinks for the molecule. It is most commonly used for holding the *pH*, that is, the acidity, of a solution at a desired level.

10.1.3 Common Signaling Pathways

Regrettably, this section looks rather like an alphabet soup. It can be skimmed through on a first reading, but it is actually the barest of introductions to the topic. Anyone contemplating research simulations on signaling will need to go into much greater breadth and depth than this incomplete list. You should assume that there are tens of known examples in each category, each of which has tens of known isoforms. You should also be aware that the field is in an exponential growth phase: new isoforms are reported almost daily, and entirely new pathways and signaling mechanisms turn up every few months.

Receptors

These collect information from outside the cell, and couple it into intracellular pathways. There are two main subclasses of receptors. Ligand-gated ion channels are directly gated by receptors such as the familiar NMDA receptor, and pass signaling ions such as calcium into the cell when suitably stimulated by the presence of a ligand. G protein coupled receptors, such as the beta-adrenergic receptor, belong to a second category that gates channels indirectly. These receptors activate G proteins when stimulated by ligands. The G protein may then either modulate channel activity by binding directly to the channel, or regulate the activity of an enzyme involved in a second messenger pathway that modulates channel activity.

G Protein Pathways

The term *G protein* refers to proteins that bind guanine nucleotides, such as GTP. There are at least four different major G protein families, G_s, G_i, G_o, and G_q. G_t (transducin), in the visual receptor pathway, is a G protein in the G_i family that is stimulated by light. As usual, each family has five to ten known isoforms. These trimeric, membrane-associated signaling proteins contain α, β, and γ subunits. The β and γ subunits have their own sets of isoforms, so there are plenty of permutations. Only a small fraction of these are believed to occur in nature. Activated α wanders around on its own, activating further pathways, but the $\beta\gamma$ complex remains dimerized. The prototypical G protein reaction is shown in Fig. 10.2.

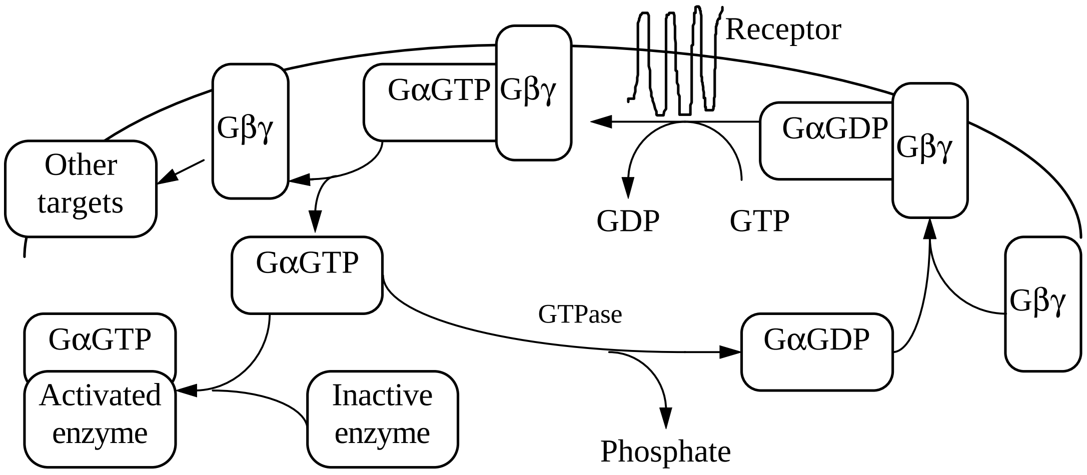


Figure 10.2 G protein signaling cycle.

The main steps in the cycle are as follows. First, the ligand activates the G protein coupled receptor. The activated receptor promotes the replacement of GDP by GTP, following which, the active G_α-GTP splits off from the $G_{\beta\gamma}$ dimer. The active G_α binds to an effector enzyme such as adenylyl cyclase, and turns it on. In parallel, $G_{\beta\gamma}$ binds to its own target enzymes or channels. After a while (a few seconds to minutes), the slow GTPase activity of the G_α hydrolyzes the GTP to GDP. Now the pieces reassemble: G_α-GDP and the $G_{\beta\gamma}$ associate, and the inactive heterotrimer can bind to another receptor to repeat the cycle. There are two major features to note here. First, the G protein loop *amplifies* signals. A receptor can catalyze the activation of numerous G proteins while the ligand is bound. Each G_α remains active until the GTP is hydrolyzed, and this takes a minute or so. During this time, the activated effector enzyme can process a large amount of substrate. Second, there are *two* signaling outputs from a G protein: the G_α and the $G_{\beta\gamma}$. Each can potentially influence several signaling pathways.

There is also a large subfamily of "small G proteins," which lack $\beta\gamma$ subunits. These include Ras, Rab, and many others. They have their own family of regulatory proteins that affect the rate of GTP turnover.

Protein Kinase Pathways

Protein kinases add phosphate groups to specific sites on other proteins, and thereby affect their activity. There are dozens of kinases known, which fall into two main classes: serine/threonine kinases and tyrosine kinases. These classes indicate the amino acids that are phosphorylated by the kinase. The major serine/threonine kinases are PKC (protein kinase C), PKA (protein kinase A), CaM-KII (calcium-calmodulin regulated type II kinase), and MAPK (mitogen-activated protein kinase). Receptor tyrosine kinases (RTKs) are an important class of tyrosine kinase. Beyond the specificity for a particular amino acid substrate, each kinase has its own range of sequence preferences, which can be exquisitely specific or very broad. Kinase activity is regulated in many ways, including second messengers such as Ca, DAG or cAMP, phosphorylation, membrane association, or even direct ligand binding.

Phosphatases

These reverse the activity of kinases: they remove phosphate groups. Like kinases, they have their own classes, isoforms, substrate specificities and so on. Major phosphatases are PP-2A (protein phosphatase 2 A), PP-1 (protein phosphatase 1) and calcineurin. It used to be thought that their only role was to balance out kinase function, but it is now clear that they are in turn regulated in a wide variety of ways and contribute actively to signaling.

Phospholipases

These take phospholipids, which are a major constituent of the cell membrane, and turn them into active signaling molecules. Examples include PLA_2 (phospholipase A_2), PLC-β (phospholipase C-β), and phospholipase D. PLC-β is particularly interesting as it produces two signals: diacylglycerol (DAG) and IP_3 (inositol triphosphate). PLA_2 produces AA, which is a membrane-permeable signal and may be involved in retrograde signaling. It should be pointed out that many phospholipids themselves are important in signaling.

Cyclases

The main cyclases are AC and GC (adenylyl and guanylyl cyclases), which produce the cyclic nucleotides cAMP and cGMP from ATP and GTP, respectively. There are at least ten ACs known, each of which has its own regulatory preferences. All ACs are activated by $G_{s\alpha}$; most are inhibited by $G_{i\alpha}$, and there is regulation by $\beta\gamma$, phosphorylation, and so on, in various combinations.

Phosphodiesterases

These degrade cAMP and cGMP, and thereby turn off signals from cyclases. Regulatory mechanisms include CaM-dependence and phosphorylation.

Calcium

Although calcium is just a simple ion, and not really a pathway, it is arguably the most important single signaling molecule. The regulation of Ca alone involves ligand and voltage-gated channels, pumps, intracellular stores, buffers, diffusive microenvironments and more. It is involved in the regulation of almost all kinds of pathways, and there are whole families of proteins such as CaM (calmodulin), whose only role is to detect Ca levels as part of the regulation of other proteins.

10.2 Modeling Signaling Pathways

10.2.1 Theory

Chemical rate theory is an old, well-established subject. A lot of it has to do with analytical derivation of results that we, in our modern-day decadence, leave to computers. For our purposes all we have to understand is a single rate equation:

$$\mathrm{A} + \mathrm{B} \underset{k_b}{\overset{k_f}{\rightleftharpoons}} \mathrm{C} + \mathrm{D}. \tag{10.1}$$

By definition, this is described by a differential equation of the form

$$d[A]/dt = k_b[C][D] - k_f[A][B], \tag{10.2}$$

where the square brackets indicate the concentration of the specified molecule. The mass conservation laws constrain the remaining concentrations, once the starting concentrations and the current value of [A] are known. You have already encountered very similar rate equtions when studying the Hodgkin-Huxley model of activation of sodium channels (Eqs. 4.11–4.13).

Enzyme catalysis is a very important part of biochemical reactions. One of the simplest and best descriptions of enzyme kinetics is due to Michaelis and Menten. Way back in 1913, they described catalysis as a two-step process, where the enzyme and substrate first reversibly combine to form a complex (Michaelis and Menten 1913). This complex can then can go forward to form the product in an essentially irreversible step:

$$\mathrm{Enz} + \mathrm{Sub} \underset{k_2}{\overset{k_1}{\rightleftharpoons}} \mathrm{Enz.Sub} \overset{k_3}{\rightarrow} \mathrm{Enz} + \mathrm{Prd}. \tag{10.3}$$

Although we need three rate constants here, Michaelis and Menten went on to specify only two parameters which do an excellent job of representing enzyme properties. These are:

$$K_m = (k_2 + k_3)/k_1 \tag{10.4}$$

and

$$V_{max} = k_3. \tag{10.5}$$

K_m is the Michaelis-Menten constant for the enzyme. It is the concentration of substrate at which the rate of generation of product is half maximal. I leave the derivation of this as an exercise for the reader.

V_{max} is the maximum velocity of the enzyme, i.e., the rate of formation of product when there are saturating amounts of substrate present. All of the enzyme will then be complexed with the substrate, so V_{max} is just k_3.

10.2.2 Sources of Data

A brief digression on sources of data is in order here. It is almost certain that a modeler in search of biochemical data will have to refer to the original literature, both for the reaction mechanisms and for the parameters. *Journal of Biological Chemistry* accounts for about half of the work on the subject. It provides its entire contents on the Internet (*http://www-jbc.stanford.edu/jbc/*) and on CD-ROM, making literature searches almost pleasant. The remainder is partitioned between many other journals, such as *European Journal of Biochemistry*, *Proceedings of the National Academy of Science*, *Biochemistry*, and many others.

10.2.3 Figuring Out the Mechanisms

Signaling pathways are typically represented as neat little black boxes connected with arrows. The first step in constructing a model of such a pathway is to open up the black box and recoil in horror at the seething mass of interactions inside. These interactions, which are frequently ill-defined in mechanistic terms, must be framed in terms of individual chemical reactions. In GENESIS, these are binding/reversible reactions, and Michaelis-Menten enzyme reactions. Complete mechanistic details for signaling pathways are available in only a few cases. G protein signaling, for example, has been worked out in considerable detail (Fig. 10.2, and Gilman 1987). The classical pathway for cAMP signaling has also been extensively studied and modeled (Levitzki 1984). Lauffenburger and Linderman (1993) provide many examples of signaling pathways that have been modeled with varying degrees of realism. It is much more common to have partial mechanistic details available. For example, several important kinases including PKC and PKA are known to incorporate an enzyme site and a pseudosubstrate region, which binds to and inactivates the enzyme site. The process of activation of these enzymes can then be described in terms of the release of the pseudosubstrate from the enzyme site. As a modeler, one may have to make compromises on several fronts:

- If one is lucky, the mechanistic data may actually go beyond what you need or can handle. PKA regulation is a case where there is such an overabundance of information (e.g., Døskeland and Øgreid 1984). Judicious simplification is usually not difficult in such cases.

- At the other extreme, mechanistic details may be so sparse that you have to develop an "empirical" model, which uses the simplest possible mechanisms that fit the observed data. This approach works surprisingly well, especially if there are plenty of data to constrain the system. A common situation is where there are several known activators for an enzyme, each of which is described by a different concentration-effect curve. The dumbest approach is then to treat each activation process as a separate reaction, resulting in an activated enzyme with different rates. This has been done, for example, for the activation of PLA_2. One can often go a step further and rule out certain possible mechanisms on the basis of such raw concentration-effect curves. This is discussed below.

- The general, and most common case, is that some of the crucial mechanistic details are known, and the rest are a bit fuzzy. One can often fall back on mechanisms for similar pathways to fill in the gaps. Otherwise, one just has to plug the holes with the simplest mechanisms that will work.

As an example of figuring out mechanistic details, consider protein kinase C (Nishizuka 1992). It is known to be regulated by a pseudosubstrate domain that normally binds to and blocks the kinase site. Activators function by releasing this blockage, in assorted ways. This crucial regulatory information enables us to predict that most activators will expose an enzyme site of identical activity. This implies that the strength of the activator is likely to depend on its efficacy in releasing the block, rather than on special properties of the enzyme-activator complex. So, the active enzyme can be represented as a single pool, to which different activators contribute an amount determined by their respective efficacy. To round off the picture, let us consider activation by Ca, DAG, and AA. This lets us draw up the first skeleton of the activation process (Fig. 10.3). We just need to sum up the individual contributions into a total final activity, as we assume that the exposed enzyme site is identical in all cases. We can further elaborate on the activation process by noting that each of the activators works synergistically with the others. One way of modeling this might be to add further reactions where the two activators combine, as illustrated by the Ca-AA synergistic pathway in dotted lines. As it turns out, we can considerably refine the activation mechanisms because the concentration-effect curves require even more complex interactions for the model to fit well.

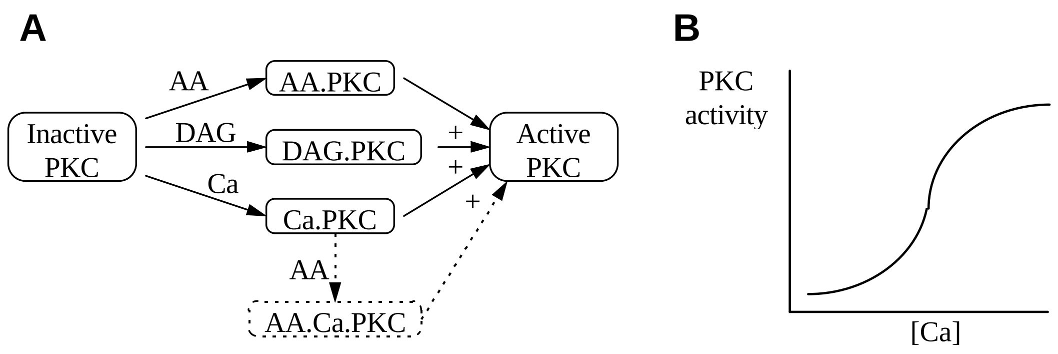


Figure 10.3 (A) Skeleton model of PKC activation. (B) Example concentration-effect curve.

10.2.4 Reaction Rate Constants

When a signaling pathway is modeled in terms of biochemical reactions, the immediate parameters required are rate constants and initial concentrations. Unfortunately, biochemists rarely provide parameters with modelers in mind. The most common form for expressing biochemical data is the *concentration-effect curve* (Fig. 10.3B) which describes an *effect*, such as activation, rate of production, or some similar experimental quantity, in terms of the *concentration* of an activating agent. Mechanistically, the link between the concentration and the effect is quite likely to involve numerous intermediate steps. By the time you get to the point where you want to fill in rates, we assume that you have a specific mechanistic model in mind, so you should have an idea of what these steps are. Obviously, you want to start with the simplest curve involving the fewest unknowns. If one must express a concentration-effect curve as a single number (losing information in the process) it is the concentration at half-maximum, K_d. Assuming that your reaction is first-order, $K_d = k_b/k_f$. If in addition one has information about the time-course τ of the reaction, one can completely define k_b and k_f. This procedure is fine in theory. In practice, if you do not actually run your simulation and compare the concentration-effect plot point-by-point, you will throw away a lot of valuable data and probably lose the chance to catch serious mistakes in your assumptions. Many papers report families of concentration-effect curves under various conditions. Such sets of curves embody enormous amounts of information to help constrain rates, mechanisms, and interactions between different activators.

10.2.5 Enzyme Rate Constants

Enzyme rate constants are usually reported as the maximal enzymatic rate V_{max} and the Michaelis-Menten constant K_m. This is obviously insufficient data for the three rate constants in the standard formulation for an enzyme reaction. As already mentioned, the maximum velocity of the reaction $V_{max} = k_3$. From the Michaelis-Menten definition, $K_m =$

$(k_3 + k_2)/k_1$. One option to fill in the missing rate constant is to assume that k_3 and k_2 are in a fixed ratio:

$$k_2 = xk_3. \tag{10.6}$$

Then,

$$k_1 = (1+x)k_3/K_m. \tag{10.7}$$

I use a value of $x = 4$. Exploring a huge range of such scale factors (from 0.4 to 40) leads me to believe that the behavior of most models is remarkably insensitive to the exact value of x. In other words, the Michaelis-Menten formulation captures most of the essential properties of the enzyme.

There are some exotic complications to consider when obtaining enzyme rates from the literature. Many membrane-bound measurements are conducted in artificial membrane systems. The composition of these membranes may strongly affect rates. One also has to be extremely careful of situations where the enzyme has access to only a limited amount of substrate. This may happen in experiments where both enzyme and substrate are micelle-bound, for example, PLA_2.

10.2.6 Initial Concentrations

Determining initial concentrations for signaling molecules is a task fraught with peril. In principle, one just has to look up purification stages for enzymes, and standard biochemical measurements for substrates and messengers. In practice, one needs to watch out for:

- Tissue and species specificity
- Partitioning between different fractions of tissue (membrane, cytosolic, particulate, etc.)
- Purification methods, and yield and loss of activity during purification
- Loss of activity (or sometimes even gain of activity!) during storage

and many other complications. Given all these hazards, the more references for each concentration, the better.

Typically, one will need to provide only a few initial concentrations and just let the simulation run to steady-state to obtain the remainder. This is essential, as it is experimentally very difficult to obtain steady-state values for many of the reaction intermediates. If one has equilibrium values for some molecules, that is a bonus and an excellent cross-check.

10.2.7 Refining the Model

There is no final step to building a good model of any kind. In signaling models in particular, it takes many iterations to converge to a good representation of the system. All the preceding steps, (except hopefully the theory) will need to be revisited as the model evolves. Most of my models of individual pathways have involved 20 or more cycles of improvement. An important part of this refinement includes repeated scans through the literature to hunt out alternative sources for each parameter, and cross-checks that become feasible as the model becomes more predictive.

10.3 Building Kinetics Models with GENESIS and *Kinetikit*

As with other simulations and demonstrations in this book, we draw a distinction between the underlying GENESIS simulation modules and the user interface based on them. The modules for kinetic modeling are provided by the kinetics library, whereas *Kinetikit* (or *kkit*) is the user interface. *Kinetikit* is similar to *Neurokit* in the sense that it is a research tool rather than a tutorial program. It makes much better use of the features provided by XODUS, and we hope you will find it easier to use.

10.3.1 The kinetics Library

As you will learn in Chapter 12, a GENESIS library contains the definitions of commands that will be accepted by GENESIS, as well as basic building blocks, called *objects*, which are used for the construction of simulations. The kinetics library defines additional commands and objects to extend the capabilities of GENESIS to simple models of biochemical signaling. By "simple" I merely mean that the only interactions considered are chemical. There is ample complexity in biological signaling, even when such vital complications as diffusion, compartmentalization, and enzymatic scaffolding are not addressed.

A simple exponential Euler scheme (Sec. 20.3) is used by the kinetics library. Fortunately, stiffness does not seem to be such a problem in the kinetic computations. Time steps of 2 *msec* are sufficient for most simulations of biochemical reaction systems. It is trivial to implement a crude variable time step scheme when known steep stimuli occur — just lower the time step. About 100 *μsec* will usually give you sufficient stability and accuracy for such cases. A **ksolve** object (cousin to the **hsolve**) is being designed to use much faster implicit and variable time step techniques. Even the current methods work many times faster than real-time for all but the most complex simulations.

All calculations are carried out on numbers of molecules, rather than concentrations. This simplifies computations where molecules are exchanged between chemical compartments of different volumes. However, concentrations are calculated locally by each molecular pool, by dividing the number of molecules by a *vol* field of the **pool** object. The *vol* field is typically scaled to include Avogadro's number so that the concentration units are *μM* or some other familiar units. The kinetics library assumes bulk quantities of all molecules. Markov models are specifically not simululated in the current kinetics library. Caution should therefore be employed when dealing with tiny compartments: it has been estimated that there are about ten Ca ions in a small dendritic spine. Mean rate theory may not apply in such situations.

The GENESIS objects provided by the kinetics library are the **pool** of molecules, the **reac** for simple reactions, and an **enz** which can be attached to a reactant pool to provide an enzyme site. A simple **concchan** object is provided for constructing concentration-driven channels within the kinetics library. It is a much simpler version of the voltage- and ligand-gated channels used elsewhere. These objects are described in much more detail in the GENESIS Reference Manual.

10.3.2 Kinetikit

The method of choice for developing kinetic simulations in GENESIS is to use *Kinetikit*, which is a graphical interface and simulation tool designed specifically to make it easy to build and manage complex kinetic simulations. *Kinetikit* is internally documented in detail. To get you started quickly, we'll go through a demonstration simulation that illustrates many of its features. This demonstration involves a simple reaction where a molecule A reversibly converts into a molecule B:

$$\mathrm{A} \underset{k_b}{\overset{k_f}{\rightleftharpoons}} \mathrm{B}. \tag{10.8}$$

Step 0

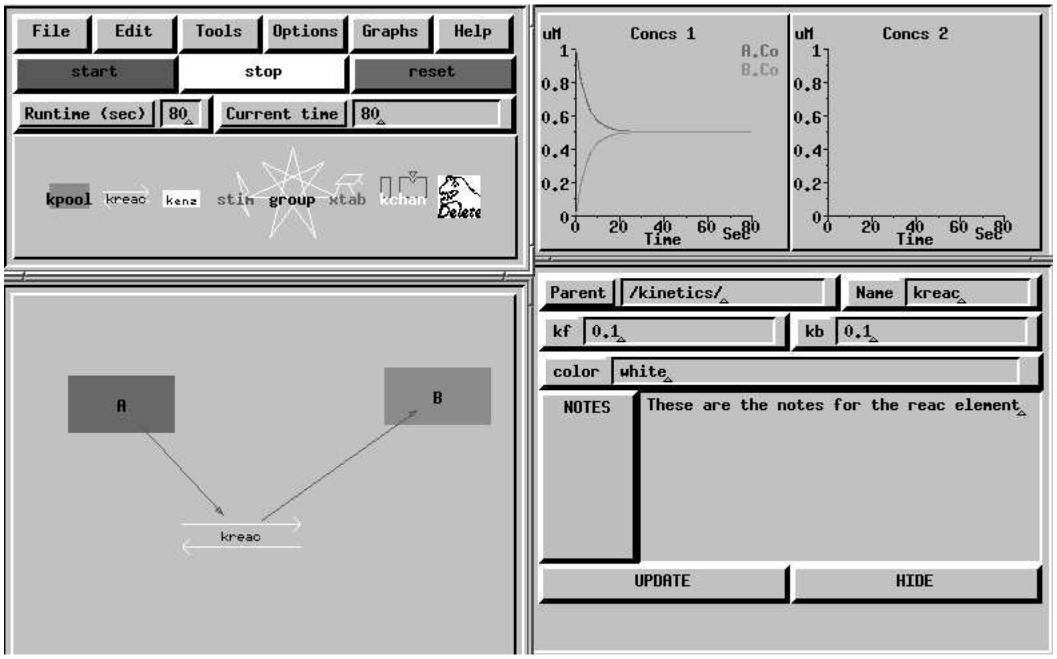
Make sure you have a version of GENESIS that includes the kinetics library. If there is a startup message of the form "The kinetics library is copylefted under the LGPL, see kinetics/COPYRIGHT." then it is there. If not, you will need to follow the installation steps detailed in the kinetics documentation. Once you have it installed, you are ready to go. After changing to the directory in which *Kinetikit* resides (usually *Scripts/kinetikit*), perform the following steps.

Step 1

Type:

```
genesis kkit
```

This will display a start-up window detailing the philosophy behind *Kinetikit* models, and also indicating the licensing terms (free!). As soon as *Kinetikit* has loaded, this window will vanish and will be replaced by a screen similar to that in Fig. 10.4. The control window includes various menu options on the top. The simulation run control buttons in bright traffic-light colors are just below, and dialogs for the simulation duration and current time below them. At the bottom is a row of strange icons. These are the building blocks for a kinetics simulation, and their role will shortly become clear. Below the control window is the *Kinetikit* edit window, where the reactions are set up. It will initially be blank. The graph windows are the only other windows currently visible.



Runtime (sec) 80 Current time 80

Parent /kinetics/ Name kreac

kf 0.1 kb 0.1

color white

NOTES These are the notes for the reac element

Figure 10.4 Screen dump of *Kinetikit.*

Step 2

Use the left mouse button to click and drag the first of the icons, the blue `kpool` icon, into the edit window. Two things should happen: a `kpool` icon should appear in the edit window, and a new window should pop up with lots of dialogs for parameters. What you have just done is to create a new pool of molecules in the simulation, and to help you along, the parameter window has been displayed. Typically the first thing one does is to change the color and name of the icon to something that suits you. Try `red` and `A`. Then hit the `HIDE` button in the parameter window.

Step 3

Familiarize yourself with the properties of the edit window. Move the `kpool` icon in the edit window around a little, using click-and-drag with the mouse. The layout of the reactions has nothing to do with the numerical calculations, but a clean layout makes it much easier to figure out what is going on. Now double-click on the `kpool` icon. The parameter window reappears. Set the initial concentration `CoInit` to 1. Now drag the `kpool` icon to the graph window. You have just made yourself a plot, labeled `A.Co`. The "`Co`" suffix indicates the field that is being plotted, in this case the concentration `Co` of element A.

Step 4

Drag in a `reac` and another `kpool` (let's call it `B`). Add the `B` to the graph as well. Now we can set up a reaction. Drag `A` onto `reac`, and then `reac` onto `B`. Little green arrows should appear, linking them. If your reaction components are too close together, the little arrows don't fit, and so they vanish. You can zoom and pan the edit window using the angle-bracket and arrow keys, to see this effect. Also try rearranging the reaction using click-and-drag within the edit window.

Step 5

Run the reaction. Hit the green `start` button, and watch the reaction proceed. You can hit the `stop` button in mid-stream, and start up again without affecting the calculations. The `reset` button clears the simulation. Play around with the parameters of the pools (esspecially `CoInit`, `nInit` and `vol`), and the reaction (`kf` and `kb`) to get a feel for how it all works. Double-click on `A` or `B` while the simulation is running, or hit the `UPDATE` button in the parameter window, to monitor concentrations. Also try double-clicking on the graph axes and on the plot labels, to pop up parameter windows for specifying display parameters.

Step 6

Edit the reaction. Drag `A` onto `kreac` again. The green arrow disappears. Repeat the drag, and it reappears. This is fine for simple editing, but what happens if you want to make it a second-order reaction $2A \rightleftharpoons B$? You will have to go to the `Options` menu and enable higher-order reactions to accomplish this feat. Now drag `kreac` onto Barney, the evil dinosaur icon in the control window. Chomp! Barney has deleted `kreac`. This works for everything in the edit window, and also for plots.

Step 7

Save the reaction. Click on the `File` menu button, fill in some notes, and enter your chosen filename, e.g., "`reacs.g`." It will be a script file, so you should have the usual "`.g`" suffix. Having saved the file, you can restore your simulation by typing:

```
genesis reacs.g
```

I will assume, for the remainder of the chapter, that you have the good sense to save each version of your simulations frequently.

Exercise for the reader:

Make and test an enzyme reaction. Hint: an enzyme site cannot exist without an enzyme. So, you will need to drag the enzyme onto an existing pool.

10.3.3 A Feedback Model

As a more complete example of the development of a kinetics simulation, we will use *Kinetikit* to come up with a feedback pathway that exhibits some interesting properties such as bistability. We then show how to run this model without the interface, and finally how to hook this up to other GENESIS components.

The Model

The feedback model consists of two enzymes, X and Y, each of which is activated by the product of the other enzyme. The activation process is second-order. We assume that the substrates are buffered: their concentration is fixed no matter how much is used up. The products are degraded in a simple first-order reaction to avoid runaway feedback. The reactions are represented in Fig. 10.5. The figure has almost a one-to-one relationship with the *Kinetikit* implementation, so it should be easy to implement. The saved model is available in the *kinetikit/examples* directory as *feedback.g*.

Notes on setting up the feedback model

1. All enzyme and reaction rates are 1.0, except for the degradation reactions, which have a k_f of 0.16 and a k_b of 0. Note that these are *not* the default values, so you will have to explicitly set them.

2. All initial concentrations are 0.0, except the X and Y substrates, which are buffered to an initial value of 1, and the inactive forms of X and Y, which are at an initial value of 1 but are not buffered.

3. Two molecules of product bind to one molecule of inactive enzyme to activate it. Look up the `Options` menu to see how to do this sort of second- and higher-order reaction. All other reactions are first-order.
4. The arrows with a double bend represent enzyme reactions.
5. The default time step of 0.01 is good for running the model.
6. Plot out the concentrations of X and Y to monitor the simulation. You can also double-click X or Y at any time to read out the concentration from the `Co` dialog.
7. Do not use spaces in the names of the components in the model. It will confuse the simulator when it tries to restore simulations from a file.

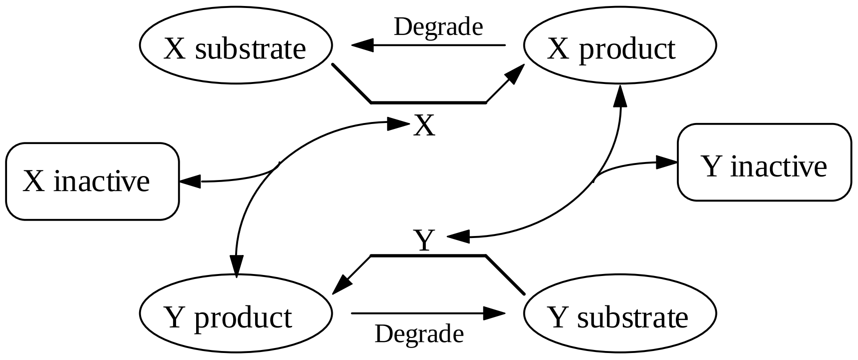


Figure 10.5 Reaction mechanism for feedback model.

Bistability

As a preliminary check on your model, just run it for, say, 100 seconds. X and Y should remain steady at a concentration of 0.0. The steady-state value you have now reached is the lower bistable point. To convince yourself that there is an upper steady-state mode, dump some X product into the system by setting its `Co` to 2, using the dialog box that will appear when you click on `X_prd`. Watch the system climb up to the upper activated state. Let it run a while longer, and convince yourself that the system has now stabilized at a new level. Note this level. If you are ambitious, you can now figure out how to push the system back and forth between the two stable levels. Are they always the same?

Concentration-Effect Curves

This manual fiddling with reaction parameters is a rather cumbersome way of identifying a bistable system, and does not easily generalize to a complex model. There is a simpler, experimentally more feasible, and theoretically informative way of characterizing a biological feedback system of this kind. This technique works by “breaking”’ the loop at some point, and plotting concentration-effect curves of X vs. Y and vice versa. When the two curves are plotted on the same axes, so that the X vs. Y curve now has been flipped around, their intersection points define the behavior of the system. You can convince yourself that a single intersection point always defines the steady-state, and that if you have three intersection points the outer two must be steady-state and the middle one the threshold. This is discussed in more detail in Bhalla and Iyengar (1997). We will use two methods for generating the concentration-effect curves. In each case, we buffer one of the enzymes to break the loop, and step it through a series of concentration values.

Brute force The simple way of doing this is to double-click on `X`, and flip the `buffer` toggle on. Now the concentration of X is set by `CoInit`. Run until it reaches steady-state, and note the concentration [Y]. Increase X, and repeat until Y saturates. Now go through the whole process again, with Y buffered and X free. Hmm . . . The numbers look familiar. Could you have predicted this?

Using xtab If you have to generate the concentration-effect curve more than once, say, for different parameters, you will probably wish to automate the whole process. The **xtab** object is good for this. This is an extended object (see Sec. 14.5) which is created by *Kinetikit*. Drag in an `xtab` icon, and connect it up to `X`. Set up the `xtab` waveform to a “staircase” of levels, with appropriate times for each step. You may need to refer to the *Kinetikit* documentation to help you with this. Activate the `xtab`, and run the simulation again. Your concentration-effect curve will be generated without further effort on your part.

Exercises:

Using the numbers you painstakingly noted down before, plot out the two concentration-effect curves of X vs. Y and Y vs. X. Put them both on the same axis and find the intersection points. You may need to smooth the curves, or find extra points. Do they tally with your previous estimates for the bistable points? Devise a way of testing the accuracy of the estimate of the threshold. Hint: adjust the degradation rates again to set up an initial condition for X and Y. Alternatively, buffer the product concentrationss of X or Y. Why won’t it work if you set up the initial condition by buffering X or Y to the initial desired level, and then release the buffering?

This model is rather unrealistic because the baseline activity of X and Y is zero. You could put in a baseline activity by setting k_b of the degradation pathway to some non-zero number. Find a number that gives reasonable results.

10.3.4 Beyond Kinetikit

The simulation files generated by *Kinetikit* are regular GENESIS script files. This means that you can manipulate them and hook them up to simulations in much the same way as other script files. (You will best understand this section after reading the introduction to GENESIS programming in Chapter 12.)

Batch Mode

The best part of a graphical interface is often the ability to turn it off. In *Kinetikit* this, and many other simulation defaults, can be assigned through the *PARMS.g* file. Make a copy of the *PARMS.g* file in your current directory, and edit it to set the `DO_X` flag to zero. Now when you load your simulation, there will be a few minor complaints which you can ignore. The simulation can be examined as usual through the command line, but no XODUS modules will be displayed. This is an ideal arrangement for those long overnight runs that you want to grind away in the background.

Modifying Parameters

The most common application for batch mode runs is parameter searches. Modifying the parameters in such situations is just a matter of using standard *setfield* and *getfield* commands, usually in batch files. When you combine this with the standard GENESIS script commands for saving values to data files, you have an effective way of automating long runs. For example, suppose we wished to run our concentration-effect curves for the feedback loop with various rates for the degradation pathways. This could be accomplished using the following script file.

```
//genesis
include fb.g // This is the file with the feedback model.
// We assume that the enzyme X is in /kinetics/X; and Y is in /kinetics/Y.

// This function generates the conc-eff curve for the enzyme enz,
function vary_enz(enz)
    str enz
    float conc
    float origconc = {getfield {enz} CoInit}
    float dconc = 0.1
    setfield {enz} slave_enable 4 // buffers the enz to CoInit
    echo % varying enz = {enz} >> conc_eff_file
    reset
    for (conc = 0.0; conc < 2; conc = conc + dconc)
        setfield {enz} CoInit {conc}
        step 200 -t
        echo {getfield /kinetics/X Co} {getfield /kinetics/Y Co} \
            >> conc_eff_file
    end
    setfield {enz} \
        slave_enable 0 \
        CoInit {origconc}
end

float dkf = 0.02
float kfx, kfy
// This loop varies kf for the X and Y degradation pathways, and generates
// a concentration-effect curve for each case.
for (kfx = 0.08; kfx < 0.24; kfx = kfx + dkf)
    setfield /kinetics/degrade_X kf {kfx}
    for (kfy = 0.08; kfy < 0.24; kfy = kfy + dkf)
        setfield /kinetics/degrade_Y kf {kfy}
        echo % kfx={kfx}, kfy={kfy} >> conc_eff_file
        vary_enz   /kinetics/X
        vary_enz   /kinetics/Y
    end
end
quit
```

The output of such a run would be a set of concentration-effect curves that could be examined to find all the intersection points.

Saving Outputs

There are various ways you can save results of simulations in *Kinetikit*. The simplest is to create plots for the desired quantities, run the simulation, and dump the results from the plots. This can be done for individual plots (double-click on the plot) or for all of them (go to the `Graphs` menu). The example script above illustrates another way of saving specific quantities to a file. The standard GENESIS commands for dumping figures to postscript files (by typing Ctrl-p within the window) also work for the graph and edit window. There are special postscript options in the `Tools` menu.

10.3.5 Connecting Kinetic Models to the Rest of GENESIS

Kinetikit allows you to do a lot of things, but only with a few kinds of GENESIS building blocks. It is often desirable to link *Kinetikit* models to the rest of GENESIS, especially to develop models that span multiple levels of neuronal function. Again, one can take advantage of the fact that *Kinetikit* models are stored in regular script files. There are two main ways of hooking kinetic models up to the rest of GENESIS.

1. Tabs A very efficient way of coupling different kinds of simulation is to use tables to provide inputs to *Kinetikit* or to a batch file based on a kinetic model. For example, you might want to feed the Ca levels from a single cell model into a kinetics simulation. Aside from its simplicity and computational efficiency, the advantage here is that the time steps used in the two situations can be radically different. The big drawback of this approach is, of course, that it assumes that the source of the tabulated data doesn't care what the kinetic part of the model does.

2. Direct messaging This is the "proper" way of hooking up kinetic and other simulations. The simplest way to get information into kinetic models is to identify a molecule whose concentration is determined by the non-kinetic part of the model, and that is explicitly modeled there. Consider Ca influx through an ion channel, which may be modeled using a **Ca_conc** object. We just need to create a kinetic "slave" counterpart of this **Ca_conc**, and hook it into the kinetic model. Likewise, messages can be sent from "pools" from the kinetics library to specify concentrations used by other GENESIS modules.

It is often necessary to do unit conversions in such situations. For example, we may need to factor in Faraday's constant, or Avogadro's number, or the volume of different cellular compartments. The **xtab** module in *Kinetikit* (which is based on the GENESIS **table** object) is a good way of implementing linear, logarithmic, or even more exotic conversion operations where necessary.

The kinetics library also allows one to send messages to **reac** elements so as to control their rate constants. This provides a great deal of flexibility. (See Exercise 5 below.)

10.4 Summary: Molecular Computation

At the risk of stating the obvious, a few aspects of the computational properties of signaling pathways are outlined here. Parallels have been drawn between signaling systems and Boolean logic (Bray 1995), and even between neural networks (Alberts et al. 1994). Such analogies, of course, are only part of the story — otherwise, we could avoid the whole messy business of modeling chemical kinetics.

It is easy to see how a signaling pathway could perform logic operations. *And* gates could be represented as enzymes that need two or more simultaneous signals for activation. *Or* gates are enzymes that can be activated by either of two signals. *Inverters* are simply inhibitory signals. Examples of all these situations exist, but the actual biochemistry is much richer than Boolean logic.

Analog computation might be a better way to think of biochemical signaling. This would allow one to consider such signaling properties as *amplification*, as we have seen for

G proteins, *synergy*, where the effect of two signals is more than additive, *cooperativity*, where the response curve is strongly nonlinear, and so on.

Even this viewpoint has its limitations. One of the most important properties of real-world neuronal as well as biochemical signaling is time-delays. Rather than being a limitation of such signaling, this is one of the most interesting elements in the computational repertoire of such networks. *Integration* and *differentiation* of the time-course of signals now becomes possible, as does short-term *storage* of information.

The analogies could be taken to the absurd, by invoking electrical circuitry as the next level of analysis. At this point the analysis is almost as complex as the original biochemistry itself, so it doesn't help much. In any case, we still have not considered the computational implications of diffusion, compartmentalization, and the anchoring of successive enzymes on protein scaffolds. Beyond that, there are all the possibilities inherent in biochemical control of the cytoskeleton, membrane traffic, and of course the genes themselves. To put the scale of this problem into perspective, computer scientists long ago realized that self-modifying code was insanely difficult to analyze. Nature has done much better here. Imagine a computer where you could redesign the CPU, while it was running, from within software. Biochemical control of the cytoskeleton and genes is a *functioning* example of self-modifying hardware, software, and everything in between.

I regard the current state of the art in the field of biochemical signaling as similar to that in electrophysiology fifty years ago: one can begin to analyze the interactions as point processes, but limitations in the available data and the techniques currently make it difficult to construct more complex models. This is likely to change rapidly in the coming years. We are now at the point where a few specific examples of microcompartmentalization of signaling pathways have been found. These have enormous implications for the ways in which we analyze such pathways. Macroscopic rate constants and concentrations become almost irrelevant in such cases. Reaction mechanisms and pathways themselves may change in ways that are nearly impossible to duplicate in a test tube. Computational methods are one possible way in which we can begin to take test-tube data and scale them down to fit inside a dendritic spine. They are also perhaps the only way in which we can tackle the growing mountain of available data, and begin to understand the immensely complex network within the cell.

10.5 Exercises

1. Using the feedback loop model, compare results at different time steps and using explicit conservation rules. You will need to look at the *Kinetikit* on-line help manual to find out how to set up conservation rules in the model.

2. Build the G protein model from Fig. 10.2. Estimate the amplification provided by this G protein. Parameters for G proteins are readily available from the literature,

or from the library of models of signaling pathways available through the GENESIS Users Group (Appendix A.3).

3. One of the most useful operations in developing complex signaling models is to merge multiple pathways into a complex model. Groups are a very useful organizing tool in such situations. Take the feedback model you built previously, and put it all into a group. Now load in the G protein model on top of this. Can you now see a role for groups? Experiment with using the G protein model as an input pathway for the feedback model.

4. In the feedback model, one way of getting the concentration-effect curve was to provide a series of concentration steps using an **xtab** element. Try to replace the step waveform with a linearly or logarithmically increasing table. Why might you have problems here? How long would you have to run to get a reasonable answer?

5. The k_f and k_b of the **reac** object class in the kinetics library can be controlled through messages. This allows one to implement many kinds of rate equations. Implement a Hodgkin-Huxley type ion channel using this facility. This can be done both using some of the special options in *Kinetikit*, and of course directly from the script language. Estimate the execution speed of such a channel, and compare with the tabchannels, which are highly optimized.

6. Exercise on buffering (challenging!): Design a buffer system for Ca that holds its concentration at 1 μM over a wide range of added Ca. Work out how to improve the buffering by changing

   - The order of the reaction
   - Cooperativity
   - The K_d of the buffer
   - The total amount of the buffer vs. its K_d, for a given initial Ca level of 0.1 μM.
   - Estimate the speed of the buffer, by seeing how well it suppresses a large but bufferable influx of Ca lasting for 10 *msec*.
   - Estimate how fast such a buffer would work to lower Ca levels if the buffer were suddenly released (by a flash of light) from a caged compound into a solution containing Ca.