ERROR PREVENTION AND TROUBLESHOOTING

Introduction

This chapter summarizes the common errors haunting the beginner modelers. Modeling is a tricky, error-inviting business, as you would probably agree through doing the homework problems. The errors can be as small as a typo, or as big as capturing the relationships between quantities incorrectly (i.e. making a wrong flowchart to start with). Some errors are easily detected and reported by Berkeley Madonna through an error message, e.g. divide by zero; but many others are not obvious to locate. The worst case is an error that does not cause any apparent trouble. No error messages, no quantities blowing up. Yet the deceptive results lead us to totally incorrect conclusions.

Although there is no error-proof way to build a model, full awareness of the common errors teaches us some good habits to avoid many of them, like opposite signs of flows, inappropriate DT, STOPTIME, parameter range, etc. Also, when our model does not work, we have some clues where to find the problem. They are summarized in the list at the end of the chapter.

Last but not least, it is highly recommended to **build models block by block** if possible. You can find and correct errors in a simpler block much more easily than in the whole model.

Starting with good flowcharts/equations

We begin with the common errors in the flowchart or equations. This is to assume that the framework of the model is correct, yet might have not been translated correctly into a flowchart or equations. Troubleshooting the framework is specific with particular models, and generally limited by the current knowledge available on the system of interest. This is beyond summary.

The common mistakes in this translation are the misuse of flow arrows and dependence arrows, and the wrong assignment of the signs of flows. The Goodwin genetic circuit model in *Biochemical Control* is used here to illustrate these points. Before those, we will brief on which quantities should be chosen as variables (tanks) along the way.

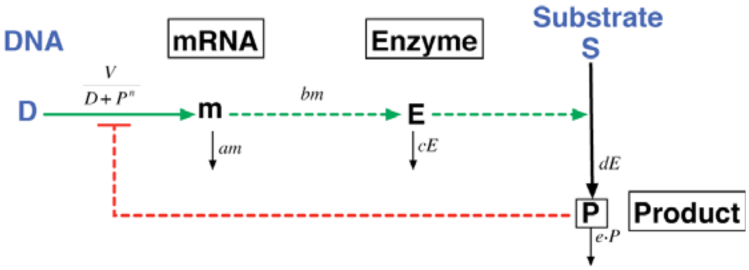


Figure 1: The Goodwin genetic circuit. Solid lines are flows, dashed lines are signals (green = enhancers, red = inhibitors). The variable quantities (reservoirs) are black, and the parameters are blue.

Tanks or buttons?

This subsection is not exactly a troubleshooting one. Yet it addresses a primary concern at the very start of a modeling job. Figure 1 is a typical cartoon diagram we normally see in a biology textbook or paper. In this case, DNA, mRNA, enzyme, substrate and product form a feedback loop. Do we need to assign a tank for each of them?

A model with many tanks takes a long time to compute and contains many parameters to be investigated. Therefore, we would like to use **as few tanks as possible** in a model, for the sake of computational efficiency and reduction of parameters. The following two categories of quantities are usually dispensable as tanks:

1. Quantities that do not change with time or change very slightly on the time scale to be studied;
2. Quantities whose changes closely follows the changes of other quantities. They can then be written as functions of other quantities.

In the Goodwin genetic circuit, the author studied a process consisting of DNA transcription, mRNA translation, and an enzymatic reaction. This process occurs on a much shorter time scale than the replication of DNA. Therefore, the amount of DNA can be considered constant in this model and is not assigned with a tank. The amount of substrate is also considered constant in this model, probably because the author was trying to describe an experiment in which the substrate is supplied as a controlled external signal. In this experiment the substrate may be maintained on a constant level or supplied much more than the cells can consume on the resolved time scale. These are good examples for the first category above.

As for the second category, imagine the experimenter adds a certain detector molecule that binds with the product and fluoresces. Suppose this detection happens very fast, so that the fluorescence level reflects the instantaneous concentration of the product. Then, this new quantity, fluorescence level, does not need to be assigned a tank. We can use a button and set it as a function of the product concentration.

So we start the model with mRNA, enzyme and product in tanks, and the other two in quantities, as shown in Figure 2.

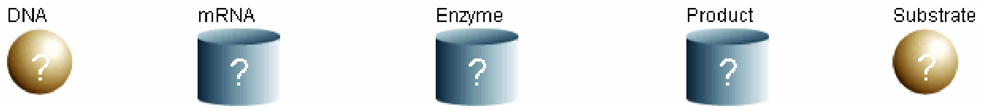


Figure 2: Setting up the tanks.

Please note that a tank choice like this might not be final. When we start modeling a system from scratch, the time scales of some steps may not be clear. Our initial choice of tanks may turn out to be inappropriate. In that case, we should change the tanks.

Flow arrow or dependence arrow?

After setting up the tanks, we need to make connections between them. Let's briefly review the difference between the flow arrow and the dependence arrow:

1. The **flow arrow** connects only tanks. It means molecule A transform into molecule B, or substances flow from compartment A into compartment B.

2. The **dependence arrow** connects tanks and buttons. It means B depends on A.

A cartoon diagram like Figure 1 usually does not distinguish an enhancement relationship vs a flow, creating confusions in model making. For example, in Figure 1 it looks as if DNA transforms into mRNA and mRNA transforms into enzyme. But this is not true. The DNA molecule only serves as a template for the transcription. It does not turn into an mRNA molecule. The helpful way to describe this relationship is that *DNA enhances the production of mRNA*. We should therefore put a dependence arrow from DNA to the production flow of mRNA. Similar connection for mRNA and enzyme.

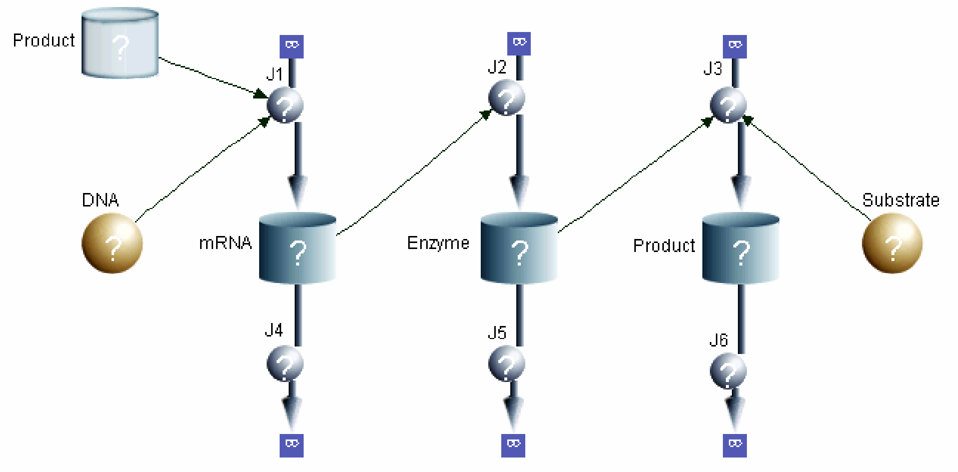


Figure 3: Tanks and connections.

Now that DNA is connected to the production flow of mRNA, what is the source of this production flow? Biologically the source is the nucleotides. But we assume, based on reasonable ground, that nucleotide is a constant supply inside the cell under normal conditions. We do not even keep track of its change. So we substitute this untracked source with an infinity sign. The same applies for the production and removal of all three tanks in this model (Figure 3).

Positive or negative flow?

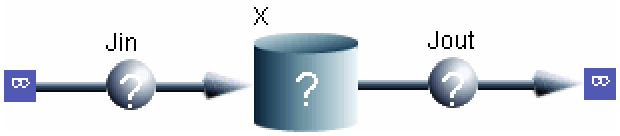


Figure 4: d/dt(X) = Jin - Jout.

Berkeley Madonna automatically generates the differential equations governing each tank by summing up the flows: the signs of incoming arrows kept and those of outgoing arrows reversed, i.e. d/dt(X) = Jin - Jout (Figure 4). In the genetic circuit flowchart shown in Figure 3, Berkeley Madonna generates the following equations:

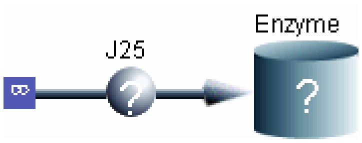
d/dt (mRNA) = + J1 - J4
d/dt (Enzyme) = + J2 - J5
d/dt (Product) = + J3 - J6

The J1, J2, J3 terms correspond to the production flows, and J4, J5, J6 correspond to the removal flows. Whatever expressions you have for J1, J2, J3 are kept their signs; the ones for J4, J5, J6 get a negative sign in front of them.

It is easy to see that this is not a unique way to draw the flow arrows. Take the enzyme tank as an example. In the above model,

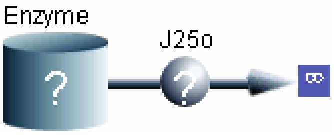
J2 = b*mRNA
J5 = c*Enzyme

You can group the production and the removal into one inflow, placing a negative sign before the removal flow:



d/dt (Enzyme) = + J25
J25 = b*mRNA - c*Enzyme

You can also use just one outflow as follows, but the sign has to be reversed:



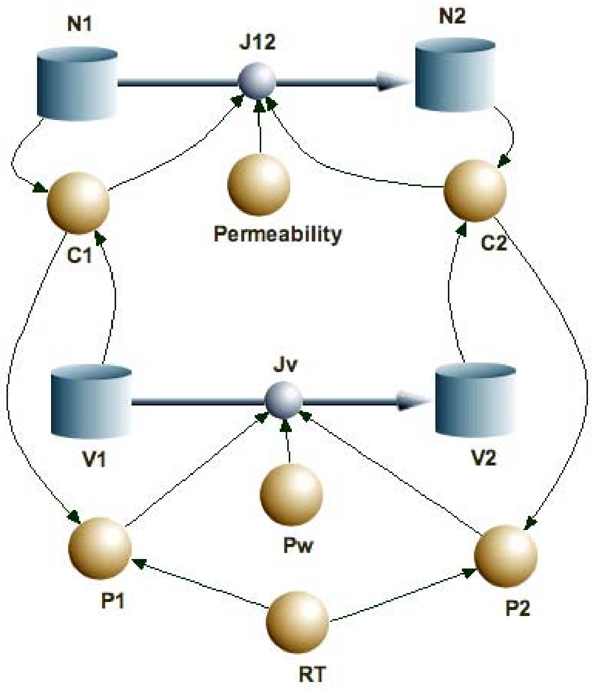
d/dt (Enzyme) = - J25o
J25o = c*Enzyme - b*mRNA

You can even do the crazy thing of exchanging J2 and J5, and placing a negative sign in both flows. But no matter which way you go, the equations of the enzyme is always d/dt (Enzyme) = b*mRNA - c*Enzyme. When you try to reproduce some model from published papers, make sure the terms in your equations end up with the same signs with those in the papers.

The above illustration shows that the directions of the arrows themselves are not important; but the signs we use for the flows have to match the directions of the arrows to represent the real flows. The genetic circuit example has clear production and removal flows. But some other cases can be more confusing in this aspect.

A typical confusing case is when the flow depends on the difference between the starting tank and the end tank. A good example is given by the model of osmotic flow through a semipermeable membrane in *Follow the Money* (Figure 5). The flow of solute J12, and the flow of water Jv, are both dependent on the difference of the concentrations in the two compartments (P1 and P2 are proportional to C1 and C2). But they have opposite signs, J12 ~ C1-C2, Jv ~ C2-C1. This can be understood through a physical reasoning. When C1>C2, the solute tends to flow from compartment 1 to compartment 2, i.e. J12>0, and the water tends to flow in the opposite direction, i.e. Jv<0. Thus, J12 ~ C1-C2, Jv ~ C2-C1.

In these confusing cases, we need to check the signs through physical reasoning shown above. There is no absolute rule to follow. If the wrong signs are used, as you may remember from the first exercise before the typo was corrected, the result of this model behaves weirdly, giving negative numbers.



```
{Top model}
{Reservoirs}
d/dt (N1) = - J12
INIT N1 = 1
d/dt (N2) = J12
INIT N2 = 0
d/dt (V1) = - Jv
INIT V1 = 1
d/dt (V2) = Jv
INIT V2 = 1
{Flows}
J12 = Permeability*(C1-C2)
Jv = -Pw*(P1-P2)
{Functions}
Permeability = 1
C1 = N1/V1
C2 = N2/V2
Pw = 1
P1 = RT*C1
P2 = RT*C2

RT = 1
```

Figure 5: The osmotic flow model, flowchart and equations. Notice the opposite sign of the highlighted flows, J12 and Jv.

Now you shall be able to build a good flowchart to start with. When you suspect about structural errors in your flowchart, you probably want to first check with the arrows and the signs of flows, which are most error-prone.

DT, STOPTIME and parameter plot

Although less fundamental than the structural errors, improper choice of DT, STOPTIME and parameter range can cause a lot of troubles, and even give deceptive results. These errors happen more often than not. They tend to be overlooked in model building because Berkeley Madonna does not require these inputs. The default choices, however, are usually unfit for the time scales or parameter ranges of an arbitrary model. Therefore, we shall pay special precautions to these preset gadgets.

DT

DT refers to the time step of the computation. All the models we study in this course are governed by differential equations given in the form of d/dt (x) = f(x). Berkeley Madonna estimates the increment of x by each time step DT. But no matter how accurate the algorithm is, such estimations produce errors. Smaller DT reduces the integration error. Therefore, we should choose a DT small enough for the computation.

But how small is small enough? The easiest check is to see if the plot alters when DT is reduced. If DT is small enough, the computation error should be smaller than resolvable on the graph, and thus the plot should NOT change with the reduced DT. Berkeley Madonna has built this function in **Compute -> Check DT/TOLERANCE**. It is highly suggested to run this check before using the model for studies of interest.

When DT is too large, the estimation errors in each step build up quickly and may significantly compromise the result. For example, in the cyclin model in *Cell Cycle Oscillations*, the default DT = 0.02 gives no oscillation, while DT = 0.002 does. Check DT/TOLERANCE at DT = 0.02 shows a very large error, as well as highly discrepant results.

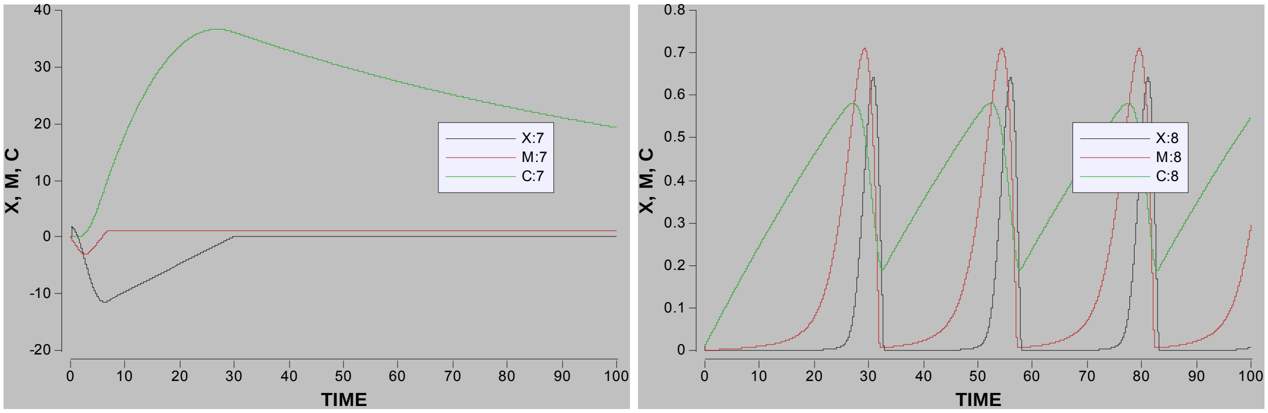
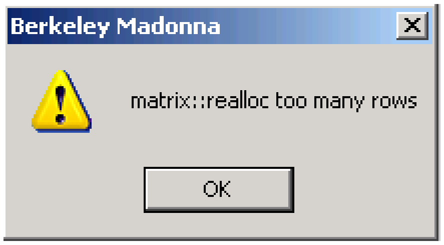


Figure 6: In the cyclin model, DT = 0.02 (left) gives a significantly compromised result, compared to the one obtained with DT = 0.002 (right).

On the other hand, a DT too small slows down the computation and requires large memory space to store the results of each time step. When the computer's memory is overloaded, it stops computing and gives a warning message shown in Figure 7. So we shall choose a DT that is **small enough and just enough**. It is not a the-smaller-the-better situation.

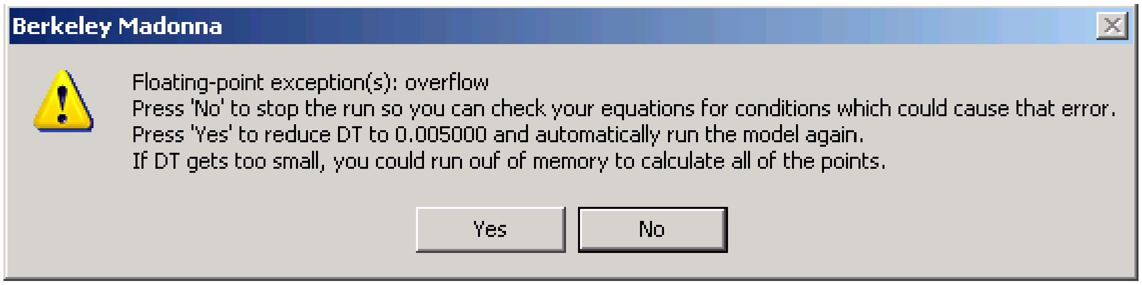


matrix::realloc too many rows

Figure 7: Warning message showing short of memory because DT is too small.

What if the DT small enough for the model already overloads the computer memory? In this case, you can turn to DTOUT. DTOUT defines the space between stored computed points. For example, if DT = 1e-5, DTOUT = 1e-3, we only store 1 point out of every 100, thus cutting off 99% of the storage space needed.

Sometimes Berkeley Madonna pops up a window like Figure 8, asking for reduction of DT. This happens when the results reach some huge number beyond the computation capacity (e.g. 3.4E38 on my computer). A DT too large may cause such blow-ups as a serious divergence from the true trajectory. In this case the problem can be rescued by reducing DT. But the blowing up of results is more likely to be caused by other errors in the model, like wrong signs of flows, zero denominators, etc.



Floating-point exception(s): overflow
Press 'No' to stop the run so you can check your equations for conditions which could cause that error.
Press 'Yes' to reduce DT to 0.005000 and automatically run the model again.
If DT gets too small, you could run ouf of memory to calculate all of the points.

Figure 8: Warning message when the result blows up beyond range. The system then asks for reduction of DT.

STOPTIME

The STOPTIME defines the time range that the model is computed. This parameter is very easy to overlook because it almost never causes any warning message. The computation runs well as long as DT is defined properly. But STOPTIME determines the time scale on which we observe the system. Different time scales may demonstrate vastly different dynamics. Let's review two examples from the previous lessons.

First, let's look at the Michaelis-Menton enzyme reaction model. The short-time graph (left, Figure 9) shows the details of the initial formation of the E-S complex within 0.5 unit time. S and P almost do not change on such a short time scale. The long-time graph (right, Figure 9) shows the long-term changes of S, P and ES. P is produced at almost the same constant rate as S is consumed. The two graphs show different aspects of the model. The choice of STOPTIME depends on which aspect we are interested in studying. If we want to study both the short-term and long-term behavior, we have to separately study the two scenarios with different STOPTIMEs.

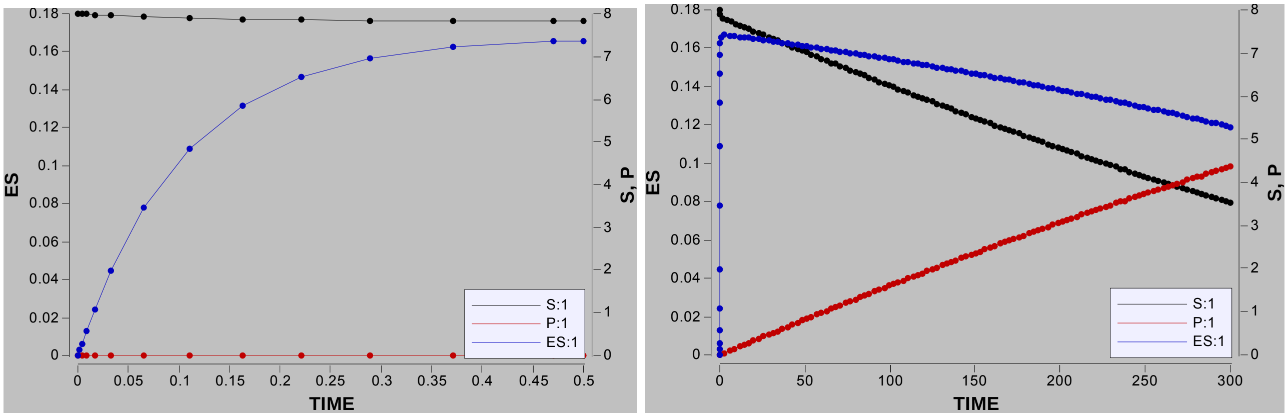


Figure 9: Michaelis-Menton enzymatic reaction model run with different STOPTIME.

Another example is again the cyclin model. With the default STOPTIME = 10 it looks as though there is no oscillations. This is because this stoptime is even shorter than the period of oscillation. Setting STOPTIME = 100 immediately reveals the oscillation. In this case, improper choice of STOPTIME leads to a wrong conclusion!

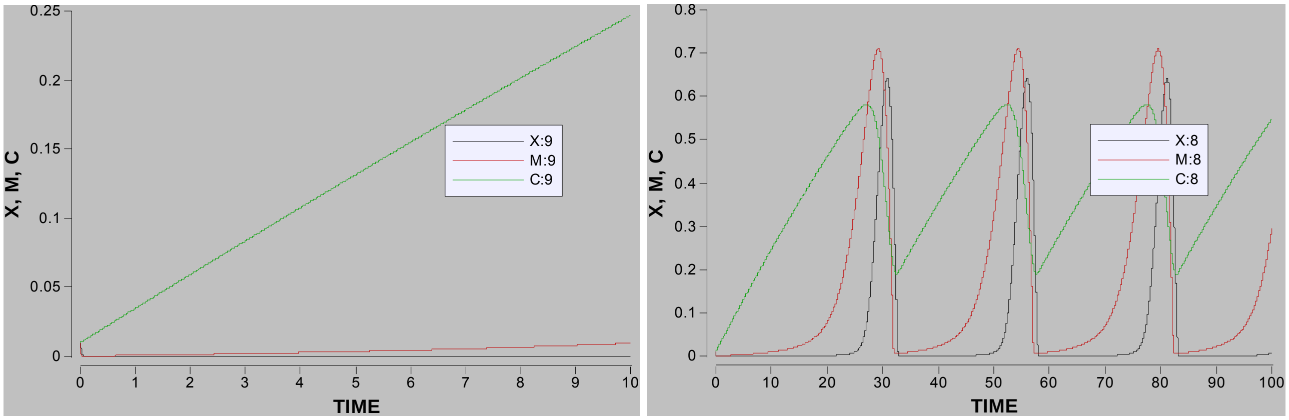


Figure 10: Cyclin models shows deceptive results when STOPTIME is too short.

How do we choose the proper STOPTIME from scratch? Usually, the experimental data from the literature provides some clues about the proper time scale. If we need to compare the model to certain data in time series, then the time range of the model just has to match that of the data. Or if the rate constant(s) of the main flow(s) are available, then we can estimate the time scale as the reciprocal of the rate constant, and choose the STOPTIME several times larger than this reciprocal to capture the whole dynamics.

Of course it never hurts to use a slider to scan a range of STOPTIMEs (more efficient on the logarithm scale). We might discover some interesting dynamics not yet known to the experimenters. But in this case, notice that your choice of tanks might limit the meaningful range of STOPTIMEs. Recall that a quantity that almost does not change within the time scale of interest can be reduced to a button. Such reduction may not be appropriate on a time scale much longer. In this case the model itself does not hold for large STOPTIME.

Parameter Plot

The parameter plot is a useful tool to show how the behavior of the model system, like the steady state values, oscillation frequencies, etc., changes with a certain parameter. But it needs to be set up carefully to give an accurate result. The appearance of a parameter plot is often highly affected by the following three factors:

- # of runs
- The range of parameter
- The STOPTIME of the model

The HIV model in *Virus Dynamics* is used here to illustrate the effects of each factor. In the model, we plot out a parameter plot of V(final) vs. β to find out the critical β value such that the virus becomes self-sustaining, i.e. V(final)>0. The desired plot is shown in Figure 11, giving a bifurcation point at β =5×10^{-8}. This plot is to be compared with the plots generated with improper choices of the above three factors.

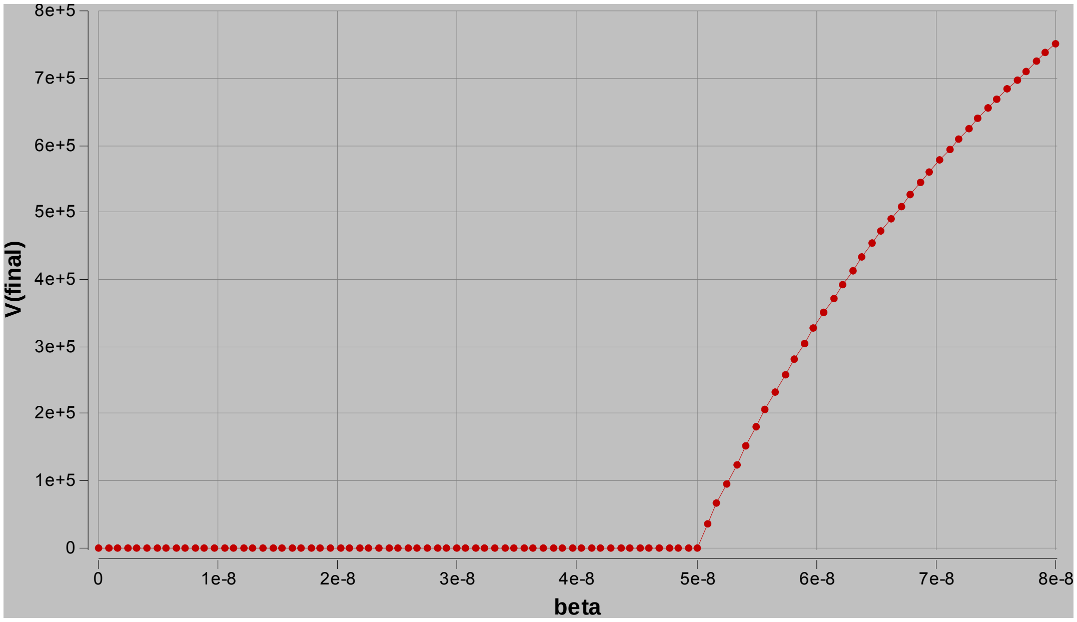


Figure 11: V(final) vs. β plot of the HIV model. The resulting critical β = 5×10-8. This plot is computed with # runs = 100, range of β = 0 ~ 8×10-8 and STOPTIME = 3000.

The # of runs determines the coarseness of the parameter plot. The default number 3 is usually way too coarse. With 3 runs the HIV parameter plot looks like Figure 12. The critical point apparently shifts to β = 4×10^{-8}, and the plot after the critical point appears linear. This distortion happens because of the lack of points. Usually several hundred runs are needed for a decent plot.

The range of parameter plays a similar role in parameter plot as STOPTIME does for a time plot. It determines the scale on which we study the parameter of interest. Improper choice of the range causes loss of interesting behaviors from the plot. Figure 13 shows two cases of improper ranges, one too large (left) and one too small (right). The left graph misses the flat part of the graph and the right graph misses the increasing part. Both graphs fail to show the bifurcation.

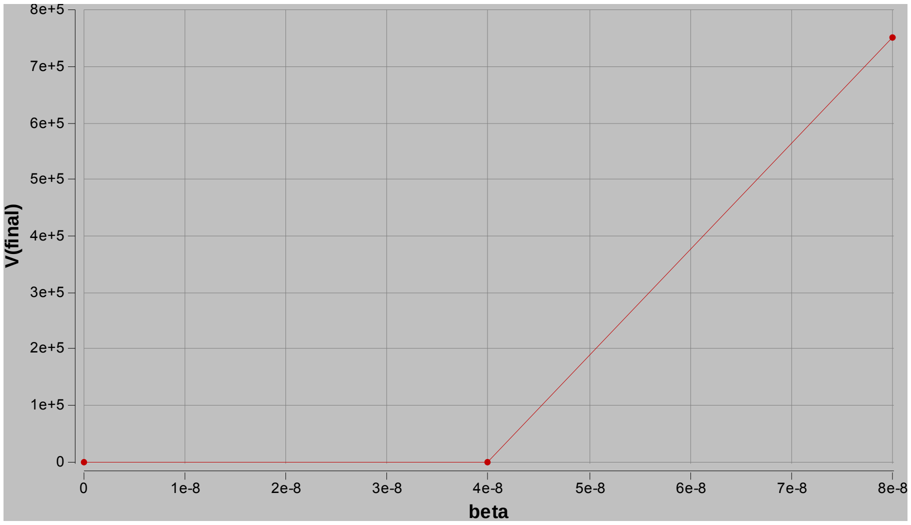


Figure 12: The parameter plot with # of runs = 3 does not show enough details.

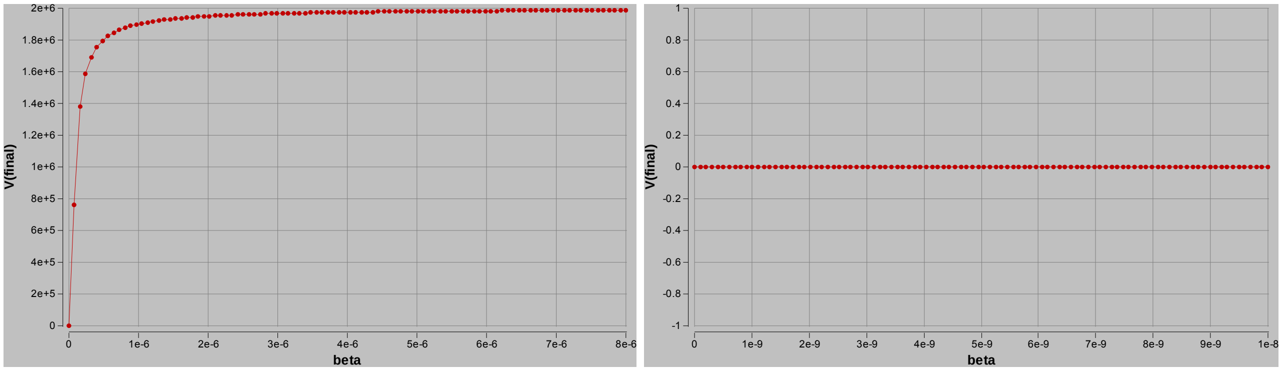


Figure 13: The parameter plot with improper ranges. Left: $\beta = 0 \sim 8\times10^{-6}$; Right: $\beta = 0 \sim 2\times10^{-8}$

The range of parameter is usually chosen to accommodate the following needs. We need to plot separate parameter plots if some of these needs conflict each other.

i. Match the real range of the parameter controlled or measured in experiments;
ii. Show a whole range of changes, e.g. increases until saturation;
iii. Locate the bifurcation points.

The first two needs are easy to understand. The third is explained a bit as follows.

Although most time plots we obtain are smooth curves, parameter plots often show abrupt changes. These abrupt changes usually correspond to the so-called bifurcation points. They are critical parameter values upon which the qualitative behavior of the model system changes, e.g. emergence/vanishing of oscillation. In the HIV model, the critical β is the bifurcation point for the emergence of self-sustaining virus population.

Because such bifurcation points incur abrupt changes on the parameter plot, the accuracy of locating them pretty much depends on the density of points we run the parameter plot with. The density of points is determined by the # of runs and the range of runs. To better show the bifurcation points, therefore, we have to choose large enough # of runs and a range closely surrounding the bifurcation point. We often need several trials to optimize the choice of the range.

The same awareness would also be helpful even when we are playing around with the parameters using a slider. We can scan on a wide range first, and then zoom in on subareas that show interesting changes of dynamics. It usually takes several rounds of readjustment. It is no good to jump into conclusions too fast.

Last but not least, STOPTIME can significantly distort the parameter plot, too. The HIV parameter plot becomes Figure 14 when STOPTIME is set to100, instead of 3000. The change of V(final) is no longer abrupt, and occurs around 6×10^{-8}. Some funny-shaped curve also happens after 7×10^{-8}. This deceptive result is a consequence of the system not resting to the steady state value within the STOPTIME for part of the parameters.

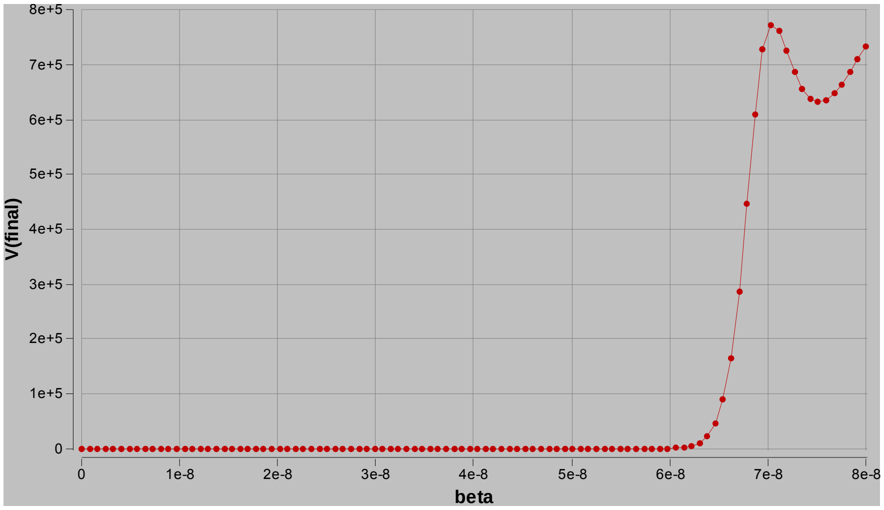


Figure 14: The parameter plot with STOPTIME = 100 (too short).

When we plot out a parameter plot of V(final), we virtually want to plot out the steady state value of the model. Figure 15 shows that V has not reached steady state by TIME = 100 for $\beta > 6\times10^{-8}$. What Figure 14 plots out for this range of β are various transient values. No wonder the result is distorted.

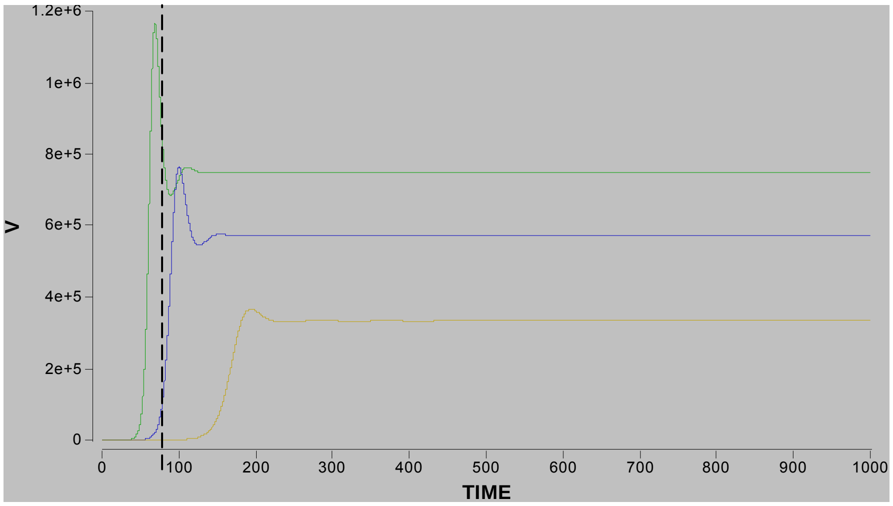


Figure 15: The system has not reached steady state by TIME = 100 for some of the parameters. β = 6×10-8 (green), 7×10-8 (blue), 8×10-8 (yellow).

The lesson learned here is that we have to choose long enough STOPTIME so that **the system can reach the steady state value at STOPTIME. And this has to be true for the entire range of the parameter**. To avoid obtaining distorted plot like Figure 15, we can use a slider to first explore the range of parameter and find out the maximum STOPTIME needed. Figure 16 shows how this is done. The model needs extremely long time (>2000*) to reach steady state right after the critical point at β = 5×10^{-8}. Therefore, we have to choose STOPTIME > 2000 to make an accurate parameter plot.

* In fact, you will find the equilibration time longer and longer when β gets closer to 5×10^{-8}. Theoretically, the equilibration time goes to infinity when β tends to 5×10^{-8}. But since our parameter plot has a limited resolution, with points spaced by about 10^{-9}, STOPTIME long enough for β = 5×10^{-8} is sufficient to make this parameter plot look accurate.

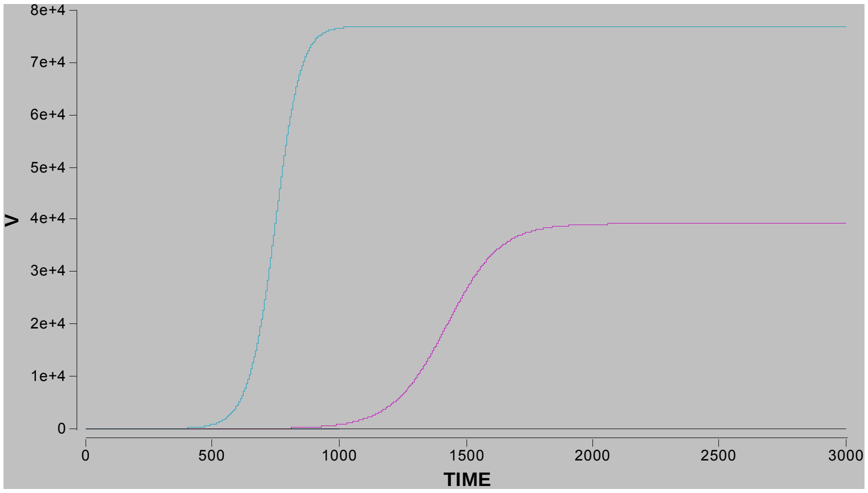


Figure 16: Using slider to explore the range of parameter. β = 5×10-8 (black), 5.1×10-8 (magenta), 5.2×10-8 (cyan). Right after the critical β = 5×10-8, the system takes an extremely long time to reach the steady state. The parameter plot can only be accurate when the chosen STOPTIME accommodates the longest cases.

Similar problems happen when we try to plot out the oscillation frequency/amplitude on a parameter plot. Figure 17 compares two parameter plots for the phosphorylation model in *Biochemical Control*. Choosing STOPTIME = 100 results in apparent steps that disappears in the plot with STOPTIME = 2000.

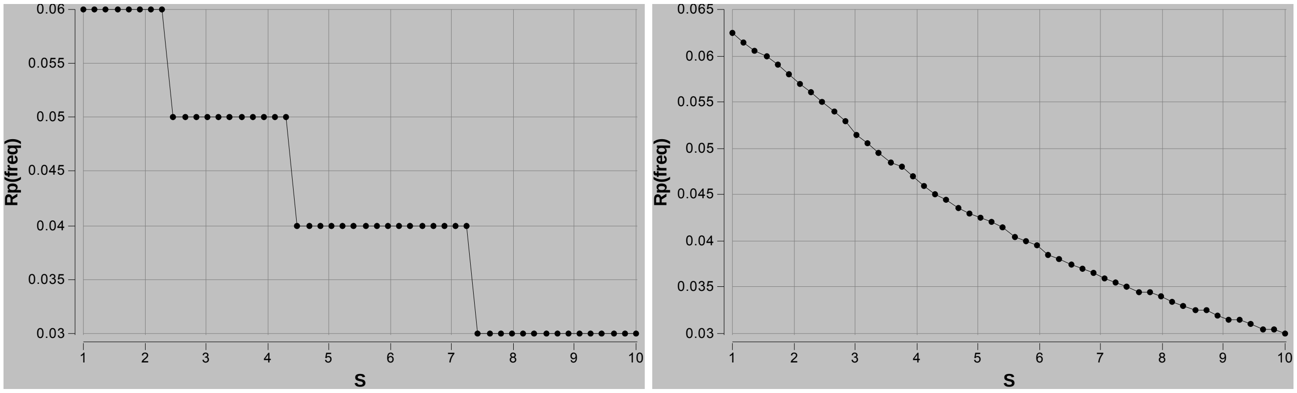


Figure 17: The parameter plot, Rp(freq) vs. S, in the phosphorylation cascade model. Left: STOPTIME = 100; Right: STOPTIME = 2000.

This happens because Berkeley Madonna somehow averages over all the computed oscillation periods to obtain oscillation frequency/amplitude. As the HIV model takes time to rest to a steady state value, the phosphorylation cascade also takes time to rest onto a limit cycle oscillation. Before that, the initial periods deviate from the final limit cycle in time and/or amplitude. This deviation causes errors in the averaging, and thus generates spurious steps in the parameter plot. In this model, the oscillation frequency ranges from 0.03-0.06. Thus, STOPTIME =2000 carries 60-120 cycles to quench the spurious steps. Even so, you can still see a little bit of steps at the right end of the plot. So, the rule of thumb is to choose STOPTIME for about 100 cycles to get an accurate parameter plot for oscillation.

Is the STOPTIME the longer the better for parameter plots? There is never an absolute yes in model making. The STOPTIME choice should depend on what we are trying to plot out. For example, if we want to plot the initial rate of some biochemical reaction, we only need the initial point. In this case computing a long STOPTIME is just a waste of time. Or if we want to plot out the peak value of some quantity, which usually happens at the early stage of the process, we would only need a STOPTIME that can capture that peak stage. There is no need to worry about the later phase.

After reading this section, we hope you are motivated to always check out for your choices of DT, STOPTIME and the parameter range. A good choice caters for the specific aspect of the model you want to show. The choices will be different depending on your specific needs.

Summary

The following list summarizes the errors reviewed in this chapter and the precautions that can help prevent the errors. As shown above, many errors do not cause dramatic problems like results blowing up. They can be deceptive and misleads us to wrong conclusions. With no standard answers to refer to, it is better to use the precautions and kill the errors in the cradle.

Precautions against the error	Potential problems caused by the error

ERROR PREVENTION AND TROUBLESHOOTING

1. Model with as few tanks as possible. Reduce tanks that change negligibly with time or solely depends on other tanks	Too many parameters, slow computation
2. Make flow connections and dependence connections correctly	Wrong equations
3. Watch out for the sign of flows	Wrong equations, results blowing up, etc.
4. Choose DT small enough and just enough. Use the built-in Check DT/TOLERANCE	DT too large: deviated results, may be significant DT too small: slow computation, overloaded memory
5. Choose proper STOPTIME according to the time scale you want to study	Loss of interesting dynamics
6. Choose appropriate parameter range and density of points. Use a slider-scan to aid you.	Inaccurate or incomplete parameter plot, wrong conclusion
7. Choose appropriate STOPTIME for the **entire** range of parameter for a parameter plot. Use a slider-scan to aid you. e.g. steady state value, STOPTIME > equilibration time; oscillation STOPTIME > 100 * period	Inaccurate parameter plot, wrong conclusion

There is no unique way to make a model and tune the parameters. This is what makes model-making exciting and challenging. Down to the earth, the story you want to tell is the guide for everything you do.