5

The Steady State

5.1 Steady State

The steady state is one of the most important states to consider in a dynamical model. In the literature it is also sometimes referred to as the stationary solution or state, singular points, fixed points, or even equilibrium. We will avoid the use of the term equilibrium because of possible confusion with thermodynamic equilibrium.

The steady state is the primary reference point from which to consider a model's behavior. At steady state, the concentrations of all molecular species are constant and there is a net flow of mass through the network. This is in contrast to systems at thermodynamic equilibrium, where, although concentrations are constant there is no net flow of mass across the system's boundaries. We can conveniently illustrate the steady state using a graphical procedure. Consider the simple model below:

$$X_o \xrightarrow{v_1} S_1 \xrightarrow{v_2} X_1$$

where X_o and X_1 are constant boundary species and S_1 is a floating species. For illustration purposes we will assume some very simple kinetics for each reaction, v_1 and v_2. Let us assume that each reaction is governed by simple first order mass-action kinetics,

$$v_1 = k_1 X_o$$
$$v_2 = k_2 S_1$$

where k_1 and k_2 are both first-order reaction rate constants. In Figure 5.1 both reaction rates have been plotted as a function of the floating species concentration, S_1.

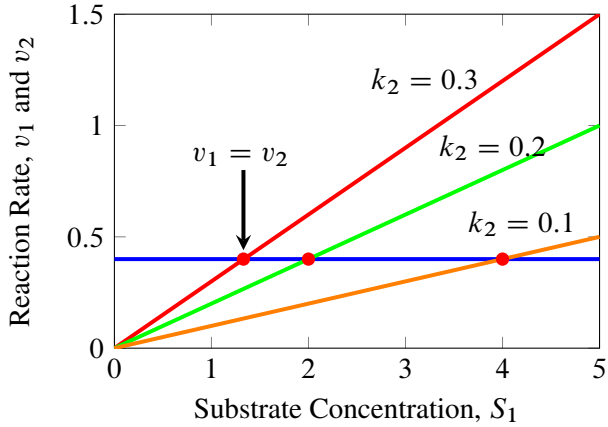

Figure 5.1: Plot of reaction rates versus concentration of S_1 and different values for k_2 for the system $X_o \rightarrow S_1 \rightarrow X_1$. The intersection of the two lines marks the steady state point where $v_1 = v_2$. $X_o = 1, k_1 = 0.4$. Note that as k_2 is decreased the steady state level of S_1 increases.

Note that the reaction rate for v_1 is a horizontal line because it is unaffected by changes in S_1 (no product inhibition). The second reaction, v_2 is

shown as a straight line with slope, k_2. Notice that the lines intersect. The intersection marks the point when both rates, v_1 and v_2 are equal. This point marks the steady state concentration of S_1. By varying the value of k_2 we can observe the effect it has on the steady state. For example, Figure 5.1 shows that as we **decrease** k_2 the concentration of S_1 increases. This should not be difficult to understand, as k_2 decreases, the activity of reaction v_2 also decreases. This causes S_1 to build up in response.

In this simple model it is also straight forward to determine the steady state of S_1 mathematically which amounts to finding a mathematical equation that represents the intersection point of the two lines. We recall that the model for this system comprises a single differential equation:

$$\frac{dS_1}{dt} = k_1 X_o - k_2 S_1$$

At steady state, we set $dS_1/dt = 0$, from which we can solve for the steady state concentration of S_1 as:

$$S_1 = \frac{k_1 X_o}{k_2} \tag{5.1}$$

This solution tells us that the steady state concentration of S_1 is a function of **all** the parameters in the system. We can also determine the steady state rate, usually called the pathway flux and denoted by J, by inserting the steady state value of S_1 into one of the rate laws, for example into v_2:

$$J = k_2 \frac{k_1 X_o}{k_2} = k_1 X_o$$

This answer is identical to v_1 which is not surprising since in this model the pathway flux is completely determined by the first step and the second step has no influence whatsoever on the flux. This simple example illustrates a rate limiting step in the pathway, that is one step, and one step only, that has complete control over the pathway flux.

A slightly more realistic model is the following:

$$X_o \xrightarrow{v_1} S_1 \xrightarrow{v_2} S_2 \xrightarrow{v_3} X_1$$

where the rate law for the first step is now reversible and is given by:

$$v_1 = k_1 X_o - k_2 S_1$$

The remaining steps are governed by simple irreversible mass-action rate laws, $v_2 = k_3 S_1$ and $v_3 = k_4 S_2$. The differential equations for this system are:

$$\begin{aligned} \frac{dS_1}{dt} &= (k_1 X_o - k_2 S_1) - k_3 S_1 \\ \frac{dS_2}{dt} &= k_3 S_1 - k_4 S_2 \end{aligned}$$

The steady state solution for S_1 and S_2 can be obtained by setting both differential equations to zero to yield:

$$\begin{aligned} S_1 &= \frac{k_1 X_o}{k_2 + k_3} \\ S_2 &= \frac{k_3 k_1 X_o}{(k_2 + k_3) k_4} \end{aligned}$$

The steady state flux, J, can be determined by inserting one of the solutions into the appropriate rate law, the easiest is to insert the steady state level of S_2 into v_3 to yield:

$$J = \frac{k_3 k_1 X_o}{k_2 + k_3}$$

Once the first step is reversible we see that the steady state flux is a function of all the parameters except k_4 indicating that the first step is no longer the rate limiting step. The equation shows us that the ability to control the flux is shared between the first and second steps. There is no rate limiting step in this pathway. Note that if we set $k_2 = 0$ then the solution reverts to the earlier simpler model.

We can also make all three steps reversible ($k_f S_i - k_r S_{i+1}$), so that the solution is given by:

$$\begin{aligned} S_1 &= \frac{X_o k_1(k_4 + k_5) + X_1 k_4 k_6}{k_3 k_5 + k_2(k_4 + k_5)} \\ S_2 &= \frac{X_1 k_6(k_2 + k_3) + X_o k_1 k_3}{k_3 k_5 + k_2(k_4 + k_5)} \end{aligned}$$

The last example illustrates the increase in complexity of deriving a mathematical solution after only a modest increase in model size. In addition, once more complex rate laws as used, such as Hill equations or Michaelis-Menten type rate laws, the solutions become exceedingly difficult to derive. As a result, in most cases, steady states are computed numerically rather than analytically.

5.2 Computing the Steady State

In those (many) cases were we cannot derive an analytical solution for the steady state we must revert to numerical methods. There are at least two methods that can be used here. The simplest approach is to run a time course simulation for a sufficiently long time so that the time course trajectories eventually reach the steady state. This method works so long as the steady state is stable, it cannot be used to locate unstable steady states because such trajectories diverge. In addition, the method can sometimes be very slow to converge depending on the kinetics of the model. As a result, many simulation packages will provide an alternative method for computing the steady state where the model differential equations are set to zero and the resulting equations solved for the concentrations. This type of problem is quite common in many fields and is often represented mathematically as the quest to find solutions to equations of the following form:

$$f(x, p) = 0 \tag{5.2}$$

where x is the unknown and p one or more parameters in the equations. All numerical methods for computing solutions to equation 5.2 start with an initial estimate for the solution, say x_o. The methods are then applied iteratively until the estimate converges to the solution. One of the most well known methods for solving equation 5.2 is called the Newton-Raphson method. It can be easily explained using a geometric argument, Figure 5.2. Suppose x_1 is the initial guess for the solution to equation 5.2. The method begins by estimating the slope of equation 5.2 at the value x_1, that is df/dx. A line is then drawn from the point $(x_1, f(x_1))$, with slope df/dx until it intersects the x axis. The intersection, x_2, becomes the next guess for the method. This procedure is repeated until x_i is sufficiently close to the solution. For brevity the parameter is omitted form the following equations. From the geometry shown in Figure 5.2 one can write down the mathematical equivalent of this procedure as:

$$\frac{\partial f}{\partial x_k} = \frac{f(x_k)}{x_k - x_{x+1}}$$

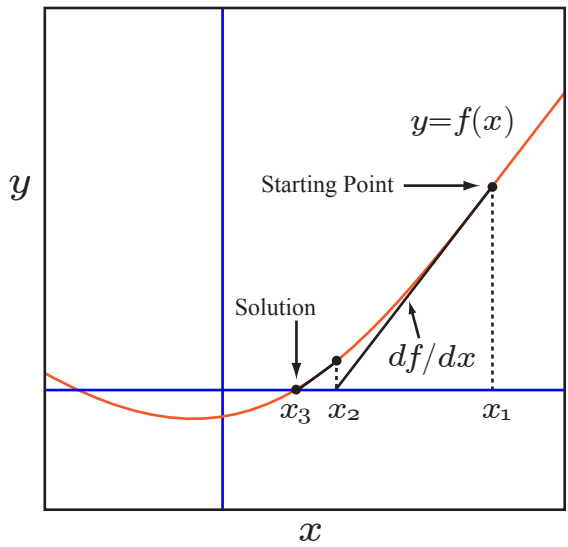

Figure 5.2: The geometry of Newton-Raphson's method

Iteration	Estimate
0	15
1	8.33333
2	5.666
3	5.0392
4	5.0001525
5	5.0

Table 5.1: Newton method used to compute the square root of 25

or by rearrangement:

$$x_{k+1} = x_k - \frac{f(x_k)}{\partial f / \partial x_k} \tag{5.3}$$

In this form we see the iterative nature of the algorithm.

Before the advent of electronic calculators that had a specific square root button, calculator users would exploit the Newton method to estimate square roots. For example, if the square root of a number, a is equal to x, that is $\sqrt{a} = x$, then it is true that:

$$x^2 - a = 0$$

This equation looks like an equation of the form 5.2. We can therefore apply the Newton formula (equation 5.3) to this equation to obtain

$$x_{k+1} = \frac{1}{2}\left(x_k + \frac{a}{x_k}\right)$$

Table 5.1 shows a sample calculation using this equation to compute the square root of 25. Note that only a few iterations are required for convergence.

One importance point to bear in mind, the Newton-Raphson method is not guaranteed to converge to the solution, this depends heavily on the start

point and the nature of the system being solved. In order to prevent the method from continuing without end in the case when convergence fails if is often useful to halt the method after a maximum of iterations, say one hundred iterations. In a case like this, a new initial start is given and the method tried again. In biochemical models we an always run a time course simulation for a short while and use the end point of that as the starting point for the Newton method. This approach is much more reliable. If the method does converge to a solution there are various ways to decided whether convergence has been achieved. Two such tests include:

1. Difference between Successive Solutions Estimates. We can test for the difference between solution x_i and the next estimate, x_{i+1}, if the absolute difference, $|x_i - x_{i+1}|$ is below some threshold then we assume convergence has been achieved. Alternatively we can measure the relative error is less than a certain threshold (say, 1%). The relative error is given by

$$\epsilon = \frac{x_{i+1} - x_i}{x_{i+1}} \times 100\%$$

   The procedure can be made to stop at the i-th step if $|f(x_i)| < \epsilon_f$ for a given ϵ_f.

2. Difference between Successive dS_i/dt Estimates. Here we estimate the rates of change as the iteration proceeds and assume convergence has been archived when the different between two successive rates of change are below some threshold.

The Newton method can be easily extended to systems of equations so that we can write the Newton method in matrix form:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \left[\frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right]^{-1} \boldsymbol{f}(\boldsymbol{x}_k) \tag{5.4}$$

If m is the number of floating species in the model, then $\boldsymbol{x}_k$ is an m dimensional vector of species concentrations, $\boldsymbol{f}(\boldsymbol{x})$ is a vector containing the m

rate of change and $\partial \boldsymbol{f}(\boldsymbol{x})/\partial \boldsymbol{x}$ the $m \times m$ Jacobian matrix.

Newton Algorithm

1. Initialize the values of the concentrations, $\mathbf{s}$, of the molecules species to some initial guess, obtained perhaps from a short time course simulation.

2. Compute the values for $\boldsymbol{f}(\mathbf{s})$, that is the left-hand side of the differential equations $(\boldsymbol{ds}/dt)$.

3. Calculate the matrix of derivatives, $\partial \boldsymbol{f}/\partial \mathbf{s}$ that is $d(\boldsymbol{ds}/dt)/\boldsymbol{ds}$, at the current estimate for $\mathbf{s}$.

4. Compute the inverse of the matrix $\partial \boldsymbol{f}/\partial \mathbf{s}$

5. Using the information calculated so far, compute the next guess $\mathbf{s}_{k+1}$

6. Compute the new value of $\boldsymbol{f}(\mathbf{s})$ at $\mathbf{s}_{k+1}$. If the value is less than some error tolerance then assume the solution has been reached, else return to step 3, using $\mathbf{s}_{k+1}$ as the new starting point.

Although the Newton method is seductively simple, it requires the initial guess to be sufficiently close to the solution in order for it to converge. In addition convergence can be slow or not occur at all. A common problem is that the the method overshoot the solution and then beings to rapidly diverge.

A further strategy that is frequently used to compute the steady state is to first use a short time course simulation to bring the initial estimate closer to the steady state. The assumption here is that the steady state is stable. The final point computed in the time course is used to seed a Newton like method, if the Newton method fails to converge then a second time course simulation is carried out. This can be repeated as many times as desired. If there is a suspicion that the steady state is unstable, one can also run a time course simulation backwards in time. In general there is no sure way of computing the steady state automatically and sometimes human intervention is required to supply good initial estimates.

As a result of these issues the unmodified Newton method is rarely used in practice for computing the steady state of biochemical models. One common variant, called the Damped Newton method is sometimes used. Both Gepasi and SCAMP use the Damped Newton method for computing the steady state. This method controls the derivative, df/dx by multiplying the derivative by a factor α, $0 < \alpha < 1$ and can be used to prevent overshoot. There are many variants on the basic Newton method and good simulation software will usually have reasonable methods for estimating the steady state.

In the last ten years more refined Newton like methods have been devised and one that is highly recommended is NLEQ2. This is used by both Jarnac and PySCeS for computing the steady state. The stiff solver suite sundials also incorporates an equation solver, however experience has shown that is it not as good as NLEQ2.

Solving the Steady State for a Simple Pathway

We are going to use the Newton-Raphson method to solve the steady state for the following simple pathway. We will assume that all three reactions are governed by simple mass-action reversible rate laws. Species X_o and X_1 are assumed to be fixed and only S_1 and S_2 and floating species.

$$X_o \xrightarrow{v_1} S_1 \xrightarrow{v_2} S_2 \xrightarrow{v_3} X_1$$

The differential equations for the model are as follows:

$$\begin{aligned}
\frac{dS_1}{dt} &= (k_1 X_o - k_2 S_1) - (k_3 S_1 - k_4 S_2) \\
\frac{dS_2}{dt} &= (k_3 S_1 - k_4 S_2) - (k_5 S_2 - k_6 X_1)
\end{aligned} \tag{5.5}$$

The values for the rate constants and the boundary conditions are given in Table **??**.

Parameter	Value
k_1	3.4
k_2	0.2
k_3	2.3
k_4	0.56
k_5	5.6
k_6	0.12
X_o	10
X_1	0

Table 5.2: Values for example (5.5).

This is a problem with more than one variable (S_1 and S_2) which means we must use the Newton-Raphson matrix form (5.4) to estimate the steady state. To use this we require two vectors, $\boldsymbol{x}_k$ and $\boldsymbol{f}(\boldsymbol{x}_k)$ and one matrix, $\partial \boldsymbol{f}(\boldsymbol{x})/\partial \boldsymbol{x}$. The $\boldsymbol{x}_k$ vector is simply:

$$\boldsymbol{x}_k = \begin{bmatrix} S_1 \\ S_2 \end{bmatrix}$$

The $\boldsymbol{f}(\boldsymbol{x}_k)$ vector is given by the values of the differential equations:

$$\boldsymbol{f}(\boldsymbol{x}_k) = \begin{bmatrix} (k_1 X_o - k_2 S_1) - (k_3 S_1 - k_4 S_2) \\ (k_3 S_1 - k_4 S_2) - (k_5 S_2 - k_6 X_1) \end{bmatrix}$$

The $\partial \boldsymbol{f}(\boldsymbol{x})/\partial \boldsymbol{x}$ matrix is the 2 by 2 Jacobian matrix. To compute this we need to form the derivatives which in this case is straight forward given that the differential equations are simple. In cases involving more complex rates laws, software will usually estimate the derivatives by numerical means. In this case however it is easy to differential algebraically to obtain:

$$\frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}} = \begin{bmatrix} \dfrac{d(dS_1/dt)}{dS_1} & \dfrac{d(dS_1/dt)}{dS_2} \\ \dfrac{d(dS_2/dt)}{dS_1} & \dfrac{d(dS_2/dt)}{dS_2} \end{bmatrix} = \begin{bmatrix} -k_2 - k_3 & -k_4 \\ k_3 & -k_4 - k_5 \end{bmatrix}$$

Notice that the elements of the Jacobian contain only rate constants. This is because the model we are using is linear. This also means we need only evaluate the Jacobian and its inverse once. If we used nonlinear rate laws such as the Michaelis-Menten rate law, the Jacobian matrix would also contain terms involving the species concentrations and in this case the Jacobian would need to be reevaluated at each iteration because the value for the species concentration will change at each iteration. For the current problem the Jacobian and its inverse is given by:

$$\text{Jacobian} = \begin{bmatrix} -2.86 & 5.6 \\ -0.56 & -11.2 \end{bmatrix} \quad \text{Jacobian}^{-1} = \begin{bmatrix} -0.3876 & -0.1938 \\ -0.01938 & -0.09898 \end{bmatrix}$$

Table 5.3 shows the progress of the iteration as we apply equation 5.4. What is interesting is that convergence only takes one iteration. This is because the model is linear. Nonlinear models may require more iterations. We can also see that after the first iteration the rates of change have very small values, this is due usually to very small numerical errors in the computer arithmetic but anything as small as 10^{-14} can be considered zero.

Computing the Steady State Using Simulation Software

The previous section showed how to compute the steady state using the Newton method. In practice we would not write our own solver but use existing software to accomplish the same thing. To illustrate this, the following Jarnac script will define and compute the steady state all at once:

```
// Define model
p = defn cell
```

Iteration	S_1	S_1	dS_1/dt	dS_2/dt
0	1	1	36.74	-10.64
1	13.18	0.6589	2.8×10^{-14}	-1.16×10^{-13}

Table 5.3: Newton-Raphson applied to a Three Step Pathway with Linear Kinetics. Starting values for S_1 and S_2 are both set at one. Convergence occurs within one iteration. Note that the values for the rates of change are extremely small at the end of the first iteration, indicating we have converged.

```
    $Xo -> S1;  k1*Xo - k2*S1;
     S1 -> S2;  k3*S1 - k4*S2;
     S2 -> $X1; k4*S2 - k6*X1;
end;

// Initialize value
p.Xo = 10;  p.X1 = 0;
p.k1 = 3.4; p.k2 = 0.2;
p.k2 = 2.3; p.k3 = 0.56;
p.k4 = 5.6; p.k6 = 0.12;

// Initial starting point
p.S1 = 1; p.S2 = 1;

// Compute steady state
p.ss.eval;
println p.S1, p.S2;
```

Running the above script yields steady state concentrations of 13.1783 and 0.658915 for S_1 and S_2 respectively, which is the same if we compare these values to those in Table 5.3. Other tools will have other ways to compute the steady state, for example graphical interfaces will generally have a button marked steady state then when selected will compute the steady state for currently loaded model.

When using Matlab the function `fsolve` can be use to solve systems of nonlinear equation and in Mathematica one would use `FindRoot`.

5.3 Stability and Robustness

Biological organisms are continually subjected to perturbations. These perturbations can originate from external influences such as changes in temperature, light or the availability of nutrients. Perturbations can also arise internally due to the stochastic nature of molecular events or by genetic variation. One of the most remarkable and characteristic properties of living systems is their ability to resist such perturbations and maintain very steady internal conditions. For example the human body can maintain a constant core temperature of 36.8°C $\pm$0.7 even though external temperatures may vary widely. The ability of a biological system to maintain a steady internal environment is called **homeostasis**, a phrase introduced by Claude Bernard almost 150 years ago. Modern authors may also refer to this behavior as **robustness**.

A biochemical pathway is dynamically stable at steady state if small perturbations in the floating species concentrations relax back to the original state.

We can illustrate a stable system using a simple two step model.

Let us assume that the two step pathway has the following form:

$$X_o \xrightarrow{v_1 = k_1 Xo} S_1 \xrightarrow{v_2 = k_2 S_1} X_1$$

Figure 5.3 illustrates the results from a simulation of a simple two step biochemical pathway with one floating species, S_1. The Jarnac script used to generate this graph is given in Table 5.4. A perturbation is made to the concentration of S_1 at a certain time by adding 0.25 units of S_1. This could be accomplished by injecting 0.25 units of S_1 into the volume where the pathway resides. The system is now allowed to evolve further. If the

```
p = defn newModel
     $Xo -> S1;  k1*Xo;
      S1 -> $X1; k2*S1;
end;

p.k1 = 0.2;
p.k2 = 0.4;
p.Xo = 1;
p.S1 = 0.5;

// Simulate the first part up to 20 time units
m1 = p.sim.eval (0, 20, 100, [<p.time>, <p.S1>]);

// Perturb the concentration of S1 by 0.35 units
p.S1 = p.S1 + 0.35;

// Continue simulating from last end point
m2 = p.sim.eval (20, 50, 100, [<p.time>, <p.S1>]);

// Merge and plot the two halves of the simulation
graph (augr(m1, m2));
```

Table 5.4: Jarnac script used to generate Figure 5.3

system is stable, the perturbation will relax back to the original steady state, as it does in the simulation shown in Figure 5.3. This system is therefore appear stable.

The differential equation for the single floating species, S_1, is given by

$$\frac{dS_1}{dt} = k_1 Xo - k_2 S_1 \tag{5.6}$$

with a steady state solution $S_1 = k_1 Xo/k_2$. The question we wish to ask here is whether the steady state is stable or not? We can show that the two step model is stable be using the following mathematical argument. The differential equation describing the two step model is given by,

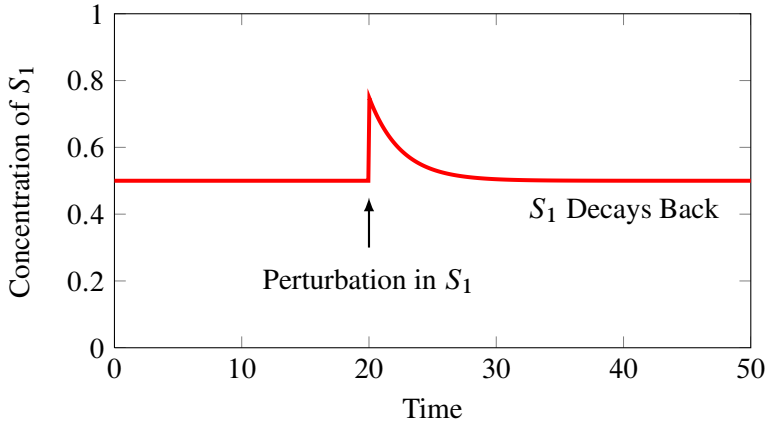

Figure 5.3: Stability of a simple biochemical pathway at steady state. The steady state concentration of the species S_1 is 0.5. A perturbation is made to S_1 by adding an additional 0.25 units of S_1 at time = 20. The system is considered stable because the perturbation relaxes back to the original steady state. See Table 5.4 for Jarnac listing.

$$dS_1/dt = k_1 X_o - k_2 S_1$$

If the system is at steady state, let us make a small perturbation to the steady state concentration of S_1, δS_1 and ask what is the rate of change of $S_1 + \delta S_1$ as a result of this perturbation, that is what is $d(S_1 + \delta S_1)/dt$? The new rate of change equation is rewritten as follows:

$$\frac{d(S_1 + \delta S_1)}{dt} = k_1 X_o - k_2(S_1 + \delta S_1)$$

If we insert the solution for S_1 (equation 5.1) into the above equation we are left with:

$$\frac{d\delta S_1}{dt} = -k_2 \delta S_1 \tag{5.7}$$

In other words the rate of change of the disturbance itself, δS_1 is negative, that is, the system attempts to reduce the disturbance so that the system re-

turns back to the original steady state. Systems with this kind of behavior are called **stable**. If the rate of change in S_1 had been positive instead of negative however, the perturbation would have continued to diverge away from the original steady state and the system would them be considered **unstable**. The important concept of stability and instability will be considered in more detail in a later section.

Dividing both sides of equation 5.7 by δS_1 and taking the limit, we find that $\partial(dS_1/dt)/\partial S_1$ is equal to $-k_2$. The stability of this simple system can therefore be determined by inspecting the sign of $\partial(dS_1/dt)/\partial S_1$ which can be easily determined by taking the derives of the differential equations with respect to the species concentrations. For larger systems the stability of a system can be determined by looking at all the terms $\partial(dS_i/dt)/\partial S_i$ which are given collectively by the expression:

$$\frac{d(d\mathbf{s}/dt)}{d\mathbf{s}} = \boldsymbol{J} \tag{5.8}$$

where $\boldsymbol{J}$ is called the Jacobian matrix containing elements of the form $\partial(dS_i/dt)/\partial S_i$. Equation **??** can be generalized to:

$$\frac{d(\delta\mathbf{s})}{dt} = \boldsymbol{J}\delta\mathbf{s} \tag{5.9}$$

where $\boldsymbol{J}$ is given by

$$\begin{bmatrix} \frac{\partial S_1/dt}{\partial S_1} & \cdots & \frac{\partial S_1/dt}{\partial S_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial S_m/dt}{\partial S_1} & \cdots & \frac{\partial S_m/dt}{\partial S_m} \end{bmatrix}$$

Equation 5.9 is an example of an unforced linear differential equation and has the general structure:

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{A}\boldsymbol{x}$$

Solutions to such equations are well known and take the form:

$$x_j(t) = \sum_{k=1}^{n} \beta_{jk} e^{\lambda_k t}$$

That is the solution to an unforced linear differential equations involves the sum of exponentials, $e^{\lambda_k t}$. The exponents of the exponentials are given by the eigenvalues (See Appendix B) of the matrix, $\boldsymbol{A}$. If the eigenvalues are negative then the exponents decay (stable) whereas if they are positive the exponents grow (unstable). We can therefore determine the stability properties of a given model by computing the eigenvalues of the Jacobian matrix and looking for any positive eigenvalues. Note that the elements of the Jacobian matrix will often be a function of the species levels, it is therefore important that the Jacobian be evaluated at the steady state of interest.

There are many software packages that will compute the eigenvalues of a matrix and there are a small number packages that can compute the Jacobian directly from the biochemical model. For example, the script below is taken from Jarnac, it defines the simple model, initializes the model values, computes the steady state and then prints out the eigenvalues of the Jacobian matrix. For a simple one variable model, the Jacobian matrix only has a single entry and the eigenvalue corresponds to that entry. The output from running the script is given below showing that the eigenvalue is -0.3. Sine we have a negative eigenvalue, the pathway must be stable to perturbations in S_1.

```
p = defn model
     $Xo -> S1;  k1*Xo;
      S1 -> $X1; k2*S1;
end;

// Set up the model initial conditions
p.Xo = 1;   p.X1 = 0;
p.k1 = 0.2; p.k2 = 0.3;

// Evaluation the steady state
p.ss.eval;
```

```
// print the eigenvalues of the Jacobian matrix
println eigenvalues (p.Jac);

// Output follows:
{    -0.3}
```

5.4 Effect of Different Kinds of Perturbations

Effect of Perturbing Floating Species

Figure 5.3 illustrates one kind of perturbation that can be made to a biochemical pathway, in this case perturbing one of the floating molecular species by physically adding a specific amount of the substance to the volume in which the pathway resides. In many cases we will find that systems will recover from such perturbations as we see in Figure 5.3. As mentioned before systems such as this are called stable. In part 2 of the book we will discuss some very interesting unstable systems where a perturbation in a floating species results in the system diverging and moving to a completely different steady state. In part 2 we will also discuss special cases where the total mass of a group of floating molecular species are constrained due to mass conservation. In such cases, perturbing one of the species in the group will cause the total group mass to change and in this case the system will not restore itself but settle slightly away from the original steady state. We will leave a more detailed discussion of these kinds of systems to part 2 of the book.

Effect of Perturbing Model Parameters

In addition to perturbing floating species we can also perturb the model parameters. Such paramors include kinetic constants and boundary species. Equation 5.1 shows how the concentration of species S_1 depends on all the parameters in the model. Moreover, changing any of the parameters results in a change to the steady state concentration of S_1 and in turn the steady state flux. When changing a parameter we can do it in two ways, we can make a permanent change or we can make a change then at some

time later can return the parameter value to its original value. Assuming that the steady state is stable, a temporary change will result in the steady state changing then recovering to the original state when the parameter is changed back. Figure 5.4 shows the effect of perturbing the rate constant, k_1 and then restoring the parameter to its original value at some time later. Table 5.5 shows the Jarnac script that was used to generate this plot. In some applications other types of perturbations are made. For example in studying the infusion of a drug where the concentration of the drug is a model parameter, one might use a slow linear increase in the drug concentration. Such a perturbation is called ramp. More sophisticated analyses might require a sinusoidal change in a parameter, an impulse or an exponential change. The main point to remember is that parameter changes will usually result in changes to the steady state concentrations and fluxes.

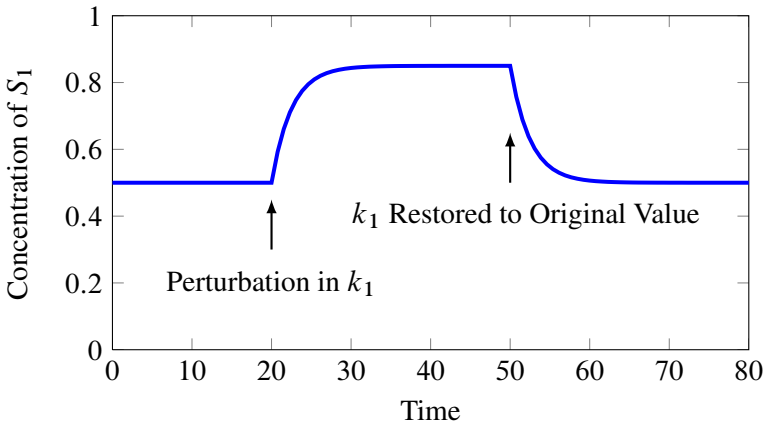

Figure 5.4: Effect of Perturbing Model Parameters using the model in Table 5.5.

5.5 Sensitivity Analysis

Sensitivity analysis at steady state looks at how particular model variables are influenced by model parameters. There are at least two main reasons why it is interesting to examine sensitivities. The first is a practical

```
p = defn newModel
     $Xo -> S1;  k1*Xo;
      S1 -> $X1; k2*S1;
end;

p.k1 = 0.2;
p.k2 = 0.4;
p.Xo = 1;
p.S1 = 0.5;

// Simulate the first part up to 20 time units

m1 = p.sim.eval (0, 20, 5, [<p.time>, <p.S1>]);

// Perturb the parameter k1
p.k1 = p.k1*1.7;

// Simulate from the last point
m2 = p.sim.eval (20, 50, 40, [<p.time>, <p.S1>]);

// Restore the parameter back to orginal value
p.k1 = 0.2;

// Carry out final run of the simulation
m3 = p.sim.eval (50, 80, 40, [<p.time>, <p.S1>]);

// Merge all data sets and plot
m4 = augr(augr(m1, m2), m3);
graph (m4);
```

Table 5.5: Jarnac script used to generate Figure 5.4

one. Many kinetic parameters we use in building biochemical models can have a significant degree of uncertainty about them. By determining how much each parameter has an influence on the model's state we can decide whether we should improve our confidence in the particular parameter. A parameter that has considerable influence but at the same time has significant uncertainty is a parameter that should be determined more carefully by additional experimentation. On the other hand a parameter that has little influence but has significant uncertainly associated with it, is relatively unimportant. A sensitivity analysis can therefore be used to highlight parameters that need better precision.

The second reason for measuring sensitivities is to provide insight. The degree to which a parameter can influence a variable tells us something about about the network is responding to perturbations and it responds to the degree it does. Such a study can be used to answer questions about robustness and adaptation. We will delay further discussion of this important topic to part 2 of the book.

How are sensitivities represented? Traditionally there are two way, one based on absolute sensitivities and the second based on relative sensitivities. Absolute sensitivities are simply given by the ratio of the absolute change in the variable to the absolute change in the parameter. That is:

$$S = \frac{\Delta V}{\Delta p}$$

where V is the variable and p the parameter. This equation shows finite changes to the parameter and variable. Unfortunately because most systems are nonlinear, the value for the sensitivity will be a function of the size of the finite change. To make the sensitivity independent of the size of the change, the sensitivity is usually defined in terms of infinitesimal changes:

$$S = \frac{dV}{dp}$$

Although absolute sensitivities are simple they have one drawback namely that the value can be influenced by the units used to measure the variable

and parameter. Often in making experimental measurements we won't be able to measure the quantity using the most natural units, instead we may have measurements in terms of fluorescence, colony counts, staining on a gel and so on. Is is most likely that the variable and parameter units will be quite different and each laboratory may have its own way particular units is uses. Absolute sensitivities are therefore quite difficult to compare.

To get round the problem of units, many people will use relative sensitivities, These are simple scaled absolute sensitivities:

$$S = \frac{dV}{dp}\frac{p}{V}$$

The sensitivity is defined in terms of infinitesimal changes for the same reason cited before. The reader may also recall that elasticities are measured in this way. Relative sensitivities are immune from the units we use to measure quantities but also relative sensitivities correspond more closely to how many measurements are made, often in terms of relative or fold changes. In practice steady state relative sensitivities should be measured by taking a measurement at the operating steady state, making a perturbation (preferable a small one), waiting for the system to reach a new steady state then measuring the system again. It is important to be aware that the steady state sensitivities measure how a perturbation in a parameter moves the system from one steady state to another.

Further Reading

1. Sauro HM (2011) Enzyme Kinetics for Systems Biology. ISBN: 978-0982477311

2. Kipp E, Herwig R, Kowald A, Wierling C and Lehrach H (2005) Systems Biology in Practice, Wiley-VCH Verlag

Exercises

1. Explain what is meant by a stable and unstable steady state.

2. The steady state of a given pathway is stable. Explain the effect in general terms on the steady state if:

   a) A bolus of floating species is injected into the pathway

   b) A permanent change in a kinetic constant.