The Steady State

Consider the following very simple two step pathway:

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

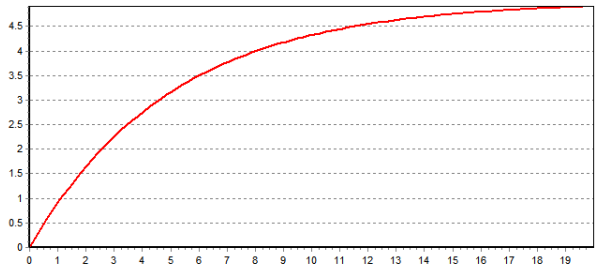
X_o and X_1 are fixed boundary species and S_1 is a floating species. Let us assume simple mass-action kinetics for each step, thus:

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

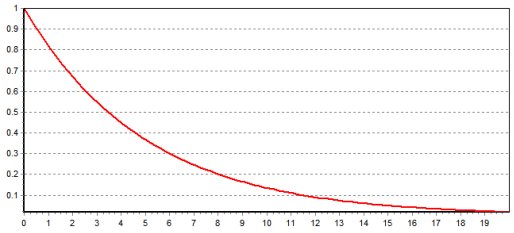
The differential equation for this simple system is then given by:

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

If we solve this equation we will get a time course for S_1 that looks something like:



Notice that the concentration of S_1 rises then reaches a plateau. If we plot the reactions rate, v_2, we would also see that it reaches a plateau. Finally if we plot the **rate of change** of S_1 we get the following:



In summary, as the system evolves it tends to the following condition:

$$\frac{dS_1}{dt} = 0$$

$$(v_1 = v_2) > 0$$

That is, the change in the state variables (floating species) is zero but there is a net flow of mass from one end of the pathway to the other (reaction rates greater than zero). This is in contrast to thermodynamic equilibrium where the net flow of mass is zero.

The state of a model where the variables are unchanging in time but where there is a net flow of mass across its boundaries is called the **steady state.**

For simple models such as the one shown above, it is easy to determine the steady state concentration algebraically. Give the differential equation that describes the system, we first set it to zero:

$$\frac{dS_1}{dt} = k_1 X_o - k_2 S_1 = 0$$

And then solve for the concentration of S_1. If we do this calculation we arrive at:

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

This equation describes how the steady state concentration of $S_1$is a function of the parameters of the system. Note that in general, a concentration or steady state rate (flux) will be a function of all model parameters. We can also find out the steady state rates (steady state rates are usually called **fluxes** in the literature) by substituting the concentration of S_1 into the rate laws:

$$J_1 = k_1 X_o$$

In this case the flux through both steps is simply $k_1 X_o$.

In most cases however it is not possible to evaluate the steady state concentrations of fluxes analytically. Instead we must compute them using numerical methods. Given that the steady state concentrations were evaluated by setting the differential equations to zero, the problem to finding the steady state is equivalent to solving a non-linear equation:

$$f(x) = 0$$

Solving Nonlinear Equations

Have you ever wondered how a pocket calculator works out the square root of a number?

The square root of a number a is given by : $x = \sqrt{a}$ or $x^2 - a = 0$. This is in the form of a nonlinear equation and if we had some way to solve nonlinear equations numerically we could find the square root of a number by inserting a number into a and solving for x. For example if we wanted to find the square root of 9, we would write:

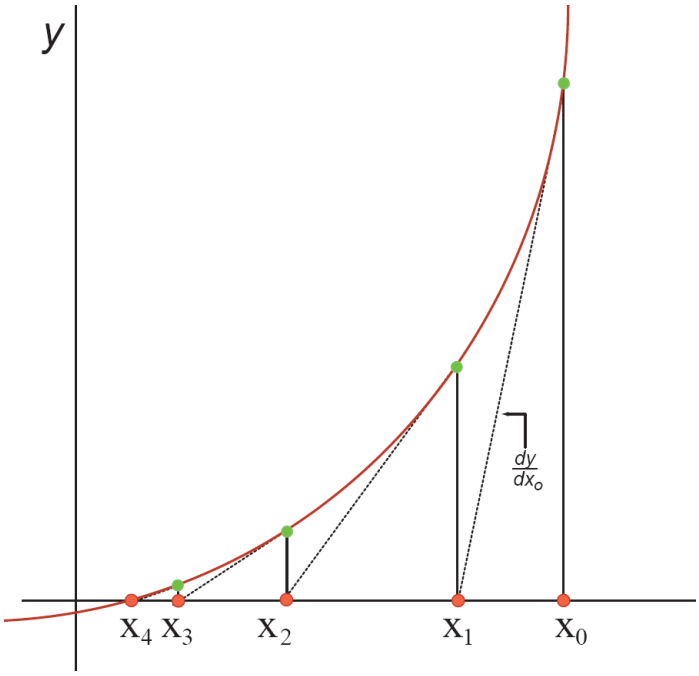
$$x^2 - 9 = 0$$

We can also rewrite this equation so that it looks like:

$$y = x^2 - 9$$

Or $$f(x) = x^2 - 9$$

y (or f(x)) will only be zero when x is the square root of 9. If we plot y versus x we will obtain a curve that looks something like the one below:



Where the line crosses the x axis (at x_4) is when $y = 0$ which is the solution to the equation. Let us assume we don't know the square root of 9 and we being with a guess, say 16. This point corresponds to x_o on the graph. Clearly this is far away from the actual solution at x_4. Let us calculate the slope at $x_o$and draw the slope so that it intersects the x axis at x_1 . Note that $x_1$is closer to the solution, let us therefore use this to find a new slope and draw this to intersect the x axis at x_2. As one can see from the graph, the x axis intersections converge quite quickly to the solution at x_4.

The technique can be formalized into an equation. The slope of the line dy/dx_o is simply the height of the line from x_0 divided by the segment length between x_1 and x_o, that is:

$$\frac{dy}{dx_o} = \frac{y_o}{x_1 - x_o}$$

This can be rearranged to compute the better solution, x_1 as:

$$x_1 = x_o - \frac{y_o}{\frac{dy}{dx_o}}$$

Or in general:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

This equation forms an iterative procedure which, by repeated use, we will converge onto the solution.

For evaluating the square root we wish to solve the nonlinear equation, $f(x) = x^2 - a$, where a is the value for which we want to the find the square root. If we apply the iterative formula we obtain:

$$x_{n+1} = x_n - \frac{x^2 - a}{2\,x_n} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right)$$

The table below shows the method used to evaluate the square root of 25 starting with an initial guess of 15. It only takes about five iterations to converge onto the solution.

Iteration	Estimate
0	15
1	8.33333
2	5.66666
3	5.0392
4	5.0001525
5	5.0

The method just described is called the **Newton-Raphson Algorithm.** The same method can be applied to solving for the steady state solution of a model by setting the rates of change to zero. Determining whether the algorithm has reached convergence can be accomplished by checking whether the relative error is less than a certain threshold, say 1%.

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

The procedure can be made to stop in the i[th] step, if $|f(x_i)| < \epsilon$.

Extension to Multiple Variables

The method can be extended to systems will multiple nonlinear equations, for example, models that have more than one floating species. The derivative that is required in the procedure now become a matrix of derivatives and the quotient must be computed by evaluating the inverse of the matrix.

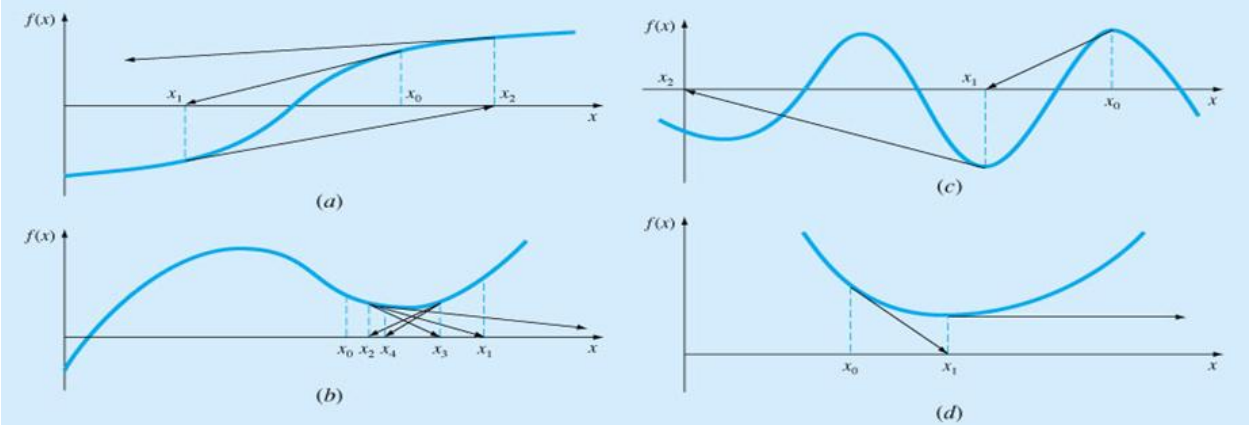
$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \left[\frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right]^{-1} \boldsymbol{f}(\boldsymbol{x}_k)$$

Where $\boldsymbol{x}$ is now a vector of variables to solve and $\boldsymbol{f}(\boldsymbol{x})$ a vector of nonlinear equations. The matrix term, $\partial \boldsymbol{f}(\boldsymbol{x})/\partial \boldsymbol{x}$**,** has special significance in modeling and is called the **Jacobian Matrix.**

The Newton Algorithm for Multiple Nonlinear Equations

1. Initialize the values of the concentrations of the molecules species to some initial guess.
2. Compute the values for $\boldsymbol{f}(\boldsymbol{x})$, that is the left-hand side of the differential equations $(d\boldsymbol{S}/dt)$.
3. Calculate the matrix of derivatives, $\partial \boldsymbol{f}/\partial \boldsymbol{x}$ that is $d(d\boldsymbol{S}/dt)/d\boldsymbol{S}$, at the current estimate for $\boldsymbol{x}$.
4. Compute the inverse of the matrix $\partial \boldsymbol{f}/\partial \boldsymbol{x}$
5. Using the information calculated so far, compute the next guess $\boldsymbol{x}_{k+1}$
6. Compute the new value of $\boldsymbol{f}(\boldsymbol{x})$ at $\boldsymbol{x}_{k+1}$. If the value is less than some error tolerance then assume the solution has been reached, else return to step 3, using $\boldsymbol{x}_{k+1}$ as the new starting point.

Warning: Sometimes the application of the Newton method will not converge either because there is no solution (i.e there is something wrong with the model), or there is a local minimum which can confuse the method. The figure below illustrates some of the bad behavior that can happen.



(From Chapra and Canele: Numerical Methods for Engineers, 2nd Edition, 1988, McGraw-Hill)

Possible reasons why the Newton method might not converge: a) Oscillations around an inflection point; b) Oscillations around a local minimum; c) Multiple solutions; d) No solution at all and the method shoots off to infinity (divide by zero slope). As a result the Newton method can fail and in such cases a short time course simulation before applying the method can greatly help (assuming there is a solution at all).

In summary the Newton method can be an effective way to find the steady state for a model. However it has some advantages and disadvantages:

1. It Requires an initial guess that is sufficiently close to the solution.

2. It is unable to find multiple solutions (assuming there are more than one possible steady state).

3. When it works, the Newton method can converge very quickly and is therefore more efficient that doing a time course simulation to locate the steady state.

Many simulation software have the ability to find the steady state of a pathway and will implement either the Newton-Raphson method directly or some variant of it. In Jarnac the command to find the steady state is simply:

p.ss.eval;

assuming that p is the variable that holds the model.