ICSB Tutorial 2009
Systems Biology Workbench

Frank T. Bergmann (fbergman@u.washington.edu) and
Kyung Hyuk Kim (kkim@u.washington.edu)

Tutorial website: http://www.sys-bio.org/sbwWiki/tutorials/icsb2009

Survey website: http://shrinkster.com/18ti

Reaction Scheme for JDesigner Introduction

$$\xrightarrow{v_1} S_1,$$

$$S_1 + ES_2^{\gamma} \underset{k_{-1}}{\overset{k_1}{\rightleftarrows}} S_1ES_2^{\gamma} \xrightarrow{k_2} ES_2^{\gamma} + S_2$$

$$S_2 \xrightarrow{v_2}.$$

$$\gamma S_2 + E \underset{k_{-3}}{\overset{k_3}{\rightleftarrows}} ES_2^{\gamma},$$

Hands on Exercises – Part I

This set of exercises will demonstrate the use of Jarnac to create hypothetical biochemical models and how to analyze them with the tools included with the SBW.

Homeostasis

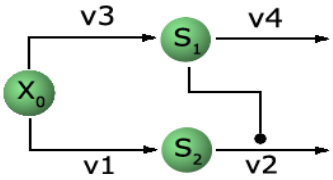
A homeostatic system is one that resists internal change when external parameters are perturbed. We will illustrate two such networks: one that shows perfect adaptation and another that near adaptation.

Perfect adaptation describes a system that recovers from a perturbation without any error (thus perfectly). There are a number of approaches to achieving perfect adaptation, one is via integral control and another, simpler approach, is via coordinate stimulation. In this tutorial we will illustrate perfect adaptation using coordinate stimulation.

A simpler and perhaps more common method for achieving homeostasis is to use negative feedback to resist external perturbations. Unlike systems which show perfect adaptation, systems which employ negative feedback cannot completely restore a disturbance.

Reaction Diagrams

S_2 remains homeostatic

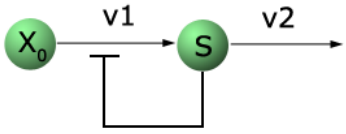


$$v1 = k_1 X_0 \qquad v2 = k_2 S_1 S_2$$

$$v3 = k_3 X_0 \qquad v4 = k_4 S_1$$

Perfect Adaptation

S remains homeostatic



$$v1 = \frac{X_0}{K_m + S^4} \qquad v2 = k_1 S$$

Negative Feedback

Exercise - Perfect adaptation

In the following example we will load a model corresponding to the Perfect Adaptation Reaction Scheme above into Jarnac / JarnacLite, and then perturb the model and see whether / how the model recovers from those perturbations.

1. Enter the following Jarnac code into Jarnac / JarnacLite. (Or download PerfectAdaptation.jan from the tutorial website):

   Jarnac code:

```
p = defn PerfectAdaptation
      $Xo  -> S2;  k1*Xo;
       S2  -> $w;  k2*S1*S2;
       $Xo -> S1;  k3*Xo;
       S1  -> $w;  k4*S1;

end;

// initialize
p.k1 = 1;
p.k2 = 1;
p.k3 = 1;
p.k4 = 1;
p.Xo = 1.0;
```

2. From the SBW menu in Jarnac, select "Simulation Tool: RoadRunner".
3. Here select "Pulse / Scan"
   a. Set Time End to 60.0
   b. Pulse Start Time to 30.0
   c. Pulse Duration to 20
   d. Pulse Value to 3

   After pressing "Pulse" the simulation will be carried out and displayed.

4. A similar effect can be seen with the "Timecourse (continuous)" tab:
   a. Select the "Time course (continuous)" tab. Click Start.
   b. From the Options menu, select "Sliders". Under the boundary variables, you will find X_0
   c. Change X_0 and observe the manner in which the two concentrations are disturbed, and then observe how S_2 recovers to the original state.
5. Of course Jarnac itself could have been easily used to perform the simulation, if we just add the following code lines:

```
m1 = p.sim.eval(0, 10,100, [<p.time>, <p.Xo>, <p.S2>]); graph(m1);
setghold(true)

p.Xo = 5;

m2 = p.sim.eval(10, 20,100, [<p.time>, <p.Xo>, <p.S2>]); graph(m2);
setghold(false)
```

Exercise - Negative feedback system

1. Enter the following Jarnac code into Jarnac / JarnacLite. (Or download NegativeFeedback.jan from the tutorial website):

   Jarnac code:

```
p = defn NegativeFeedback
      $Xo   -> S;   Xo/(km + S^h);
       S    -> $w;  k1*S;

end;


// initialize
p.h   = 4;   // Hill coefficient
p.k1  = 1;
p.km  = 1;
p.S   = 1.5;
p.Xo  = 5;
```

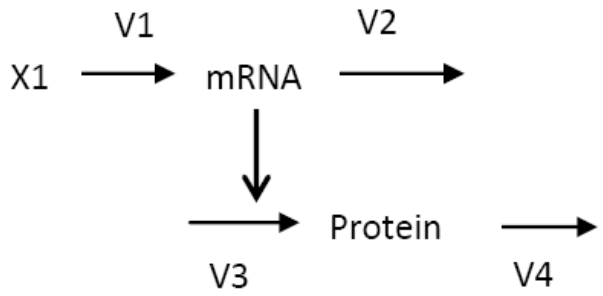
2. From the SBW menu in Jarnac, select “Simulation Tool: RoadRunner”.

1. In order to see why a negative feedback produces homeostatic behavior, one needs to see how the flow into S_2 and the flow out of S_2 are connected to the concentration of S_2. Open the Simulation Service (SBW menu -> Simulation Tool: RoadRunner).
2. Select the “Rate vs. Species” tab. Select S for the “Species”, and plot. The two curves indicate the flow in and flow out of S.
3. Use the sliders (Options -> Sliders) to see how the curves are affected by the parameters.

Hands on Exercises – Part II

Protein Synthesis

You will investigate a model that mimics protein synthesis.



The rate laws for this model are:

$$v_1 = k_1 x_1$$

$$v_2 = k_2 mRNA$$

$$v_3 = \frac{100\, mRNA}{Km_1 + mRNA}$$

$$v_4 = k_4\, Protein$$

Protein and mRNA levels can be set to zero at the start of the simulation. Run a simulation for about 200 time units and plot the graphs for mRNA and Protein.

Investigate the model under two sets of parameters:

1. k1 = 80; k2= 200; Km1 = 10; k4 = 1;
2. k1 = 0.1; k2 = 0.25; Km1 = 10; k4 = 1;

a) What difference do you observe in the two simulations?

b) Explain in mechanistic terms why the two behaviors are different. To help you answer his question, observe closely the difference in the two parameter sets.

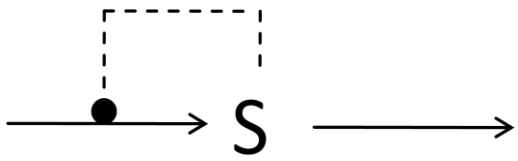
```
p= defn protein_synthesis
   v1: $Xo->mRNA; k1*Xo;
   v2: mRNA -> $X1; k2*mRNA;
   v3: $X2 -> Protein; 100*mRNA/(km1+mRNA);
   v4: Protein -> $X3; k4*Protein;
   end;
p.Xo=1;p.X1=1;p.X2=1;p.X3=1;
p.k1=80;
 p.k2=200;
p.km1=10;
 p.k4=1;

 gillLoadSBML(p.xml);
 m=gillSimulate(0,100,100);
 graph(m);
```

Stochastic Switching

Positive feedback is a common approach to generate a system with two stable states, often called bistable systems. In the stochastic framework, such bistable system can show spontaneous switching from one stable stationary state to the other. This is caused by stochastic fluctuations. We examine this stochastic switching phenomenon.

Reaction Diagram



$$v_1 = \frac{100(1+S^2)}{100+S^2}, \qquad v_2 = 3.8\,S$$

Exercise – Stochastic Simulation of Bistability

[Jarnac file name: bistability.jan, bistability-stoch.jan]

Jarnac code:

```
//Bistability
//model
p = defn bistability
     v1: $Xo -> S1; Xo*(100+100*S1*S1)/(100+S1*S1);
     v2: S1 -> $w;  k2*S1;
end;
//initialize
p.k2 = 3.8; p.Xo=1;
```

1. Let us first run model deterministically. Set the value of S1 equal to 1.0 and simulate the model by executing

```
m1 = p.sim.eval(0,10,100,[<p.time>, <p.S1>])
```

2. Generate its graph by using the command `graph(m)`

3. Choose a different value of S1 and generate a graph of S1 vs. time. Observe that the system can reach two different stationary states depending on the initial value of S1.

4. Next we run the model stochastically by using the gillSimulateOnGrid command and plot the probability density function of S1. [Refer to commands: pdf, pdfPlot, timeslice, getColumn, getColumns]. A summary of these experiments is given in the scripts below.

bistability.jan

```
//Bistability

//model
p = defn bistability
     v1: $Xo -> S1;  Xo*(100+100*S1*S1)/(100+S1*S1);
     v2: S1 -> $w;   k2*S1;
     end;

//initialize
p.k2 = 3.8;
p.Xo = 1;

p.S1 = 1;
m1 = p.sim.eval(0,10,100,[<p.time>, <p.S1>])
graph(m1)
setghold(true);

p.S1 = 10;
m2 = p.sim.eval(0,10,100,[<p.time>, <p.S1>])
graph(m2)
setghold(false);
```

bistability-stoch.jan

```
//Bistability

//model
p = defn bistability
     v1: $Xo -> S1;  Xo*(100+100*S1*S1)/(100+S1*S1);
     v2: S1 -> $w;   k2*S1;
     end;

//initialize
p.k2 = 3.8;
p.S1 =1;
p.Xo=1;

gillLoadSBML(p.xml);
m1 = gillSimulate (0, 500, 1);
graph(m1)
```