Modeling Network Motifs, I

Paul M. Magwene

January 29, 2013

Test your Python installation

1. Create a folder on your desktop called `python`
2. Open the terminal or command prompt:
   - on OS X: `Applications > Utilities > Terminal`
   - on Windows: `Start Menu > Accessories > Command Prompt`
3. Navigate to the `python` directory you created, by typing the following commands in the terminal:
   - on OS X: `cd ~/Desktop/python`
   - on Windows: `cd \Users\<username>\Desktop\python` [substitute username as appropriate]
4. Confirm that you can call Python from the command line:
   - Type `python`, which should take you into the interactive Python interpreter. You should see something like this:

     ```
     Python 2.7.3 (v2.7.3:70274d53c1dd, Apr  9 2012, 20:52:43)
     [GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
     Type "help", "copyright", "credits" or "license" for more information.
     >>>
     ```

   - Type `quit()` to quit the Python interpreter.

Copy the network simulation scripts to the `python` folder on your desktop

1. Download the `networksim.zip` zip-file from the class website
2. Unzip the archive and use the Finder/Explorer to copy all of the files in the unzipped folder to the `python` directory you created on your Desktop.

Exploring the Hill Function

The `hill-fxn.py` script generates an interactive plot representing the Hill function:

$$f(X) = \frac{\beta X^n}{K^n + X^n}$$

In the context of modeling transcription, we can think of X as the representing the concentration of a transcriptional activator and $f(X)$ as representing the promoter activity (rate of transcription) of a gene Y that is regulated by X.

- Run the `hill-fxn.py` script from your terminal by typing `python hill-fxn.py`. You should see a plot window like that shown in Figure 1.

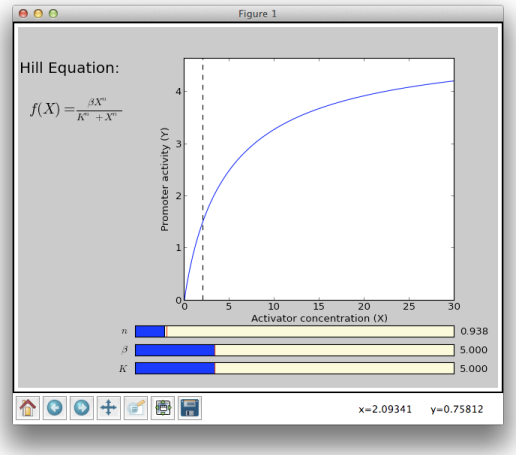

Figure 1: Screenshot of hill-fxn.py script.

- There are three sliders at the bottom of the application window. You can drag the blue regions of these sliders left or right to change the indicated parameter values. The exact values of each parameter are shown to the right of the sliders. As you drag the sliders the plot will update to show you what the Hill function looks like for the combination of parameters you have currently specified.
- Also note there is a dashed vertical line in the plot window. When you move your mouse over the plot window this line will follow your position. As you do so, x- and y-plot values in the lower left of the application window will update to show you the exact position your mouse is pointing to in the plot. The dashed line and the plot readout are useful for reading values off the plot.

Use the `hill-fxn.py` script to answer the following questions:

1. Vary the parameter n over the range 1 to 10.
   a) Describe what happens to the shape of the plot.
   b) How does changing n change the maximum (or asymptotic maximum) promoter activity (V_{max})?
   c) At what value of activator concentration is half of the maximum promoter activity reached?
2. Vary the parameter β. How does changing β change:
   a) the shape of the plot?
   b) the maximum promoter activity?
   c) the activator concentration corresponding to half-maximal promoter activity?
3. Vary the parameter K. How does changing K change:
   a) the shape of the plot?
   b) the maximum promoter activity?
   c) the activator concentration corresponding to half-maximal promoter activity?
4. Similarly explore the parameter space for a Hill function representing a transcriptional repressor using the script `hill-fxn-repressor.py`

Logic approximation to the Hill function

For the purposes of building network models we will use a *logic approximation*, where we will treat the promoter as either OFF (i.e. transcription rate is zero) or ON (transcription rate is β). We will write this as:

$$f(X) = \beta\theta(X > K)$$

where θ represents a function that evaluates to 1 if the statement inside it is True, or 0 otherwise. An alternate way to write this is:

$$f(X) = \begin{cases} 0, & \text{if } X > K; \\ \beta, & \text{otherwise.} \end{cases}$$

1. Run the script `hill-fxn-wlogic.py`
    - This is like the previous `hill-fxn.py` script except it now include a set of buttons for toggling the logic approximation on and off.
2. As before vary the parameters n, β and K. When is the logic approximation a good approximation to the Hill function?

Modeling changes in transcript concentration over time

Up until this point we've been considering how promoter *activity* changes with the concentration of a transcriptional activator or promoter. Now we want to turn to the question of how the amount of transcript of a gene Y changes over time.

How the amount of Y transcript changes over time is a function of two things: 1) a growth term which represents the rate at which the gene is being transcribed; and 2) a decay term which gives the rate at which transcripts are being degraded. We can write down a simple differential equation for this as follows:

$$\frac{dY}{dt} = f(X_1, X_2, \ldots) - \alpha Y$$

The β term represents the growth term, and will be a function of the transcription factors that regulate Y. The term, αY represents the rate at which Y is being broken down. Notice that the decay rate is a proportional to the amount of Y that is present.

We've already seen a couple of ways to model the rate of transcription – using the Hill function or its logic approximation. For the sake of simplicity we'll use the logic approximation to model the growth term. For example, in the case Y is regulated by a single input we might use $f(X) = \beta\theta(X > K_1)$. For the equivalent function where Y was regulated by two transcription factor, X_1 and X_2, and both are required to be above the respective threshold, we could use the function $f(X_1, X_2) = \beta\theta(X_1 > K_1 \text{AND} X_2 > K_2)$.

Simple activation

Now let's explore a simple model of regulation for the two gene network, $X \longrightarrow Y$.

1. Run the `simple-activation.py` script.
    - Here we assume that at time t_0 the activator, X, rises above the threshold, K, necessary to induce transcription of Y at the rate β. X remains above this threshold for the entire simulation.
    - This script includes three sliders:

a) t_{max} which sets the number of time steps over which the simulation is run
b) β
c) α

2. The concentration of Y eventually reaches a steady state, Y_{st}. How does Y_{st} relate to β and α?

3. The *response time* of a dynamical system, $T_{1/2}$is defined as the time it takes for it to go half-way between it's initial and final value.

   a) How does the response time change as you vary β?

   b) How does the response time change as you vary α?

4. One a sheet of paper (or in Excel) create a plot showing the relationship between α and response time, for $0.1 \leq \alpha \leq 2.0$.

Decay from steady state

1. Run the `simple-off.py` script

   - This script models the case where Y is initially at it's steady state, and we suddenly set the growth term, β, to zero. That is, we're modeling the case where the promoter activity is suddenly shut off. So our differential equation simplifies to:

$$\frac{dY}{dt} = -\alpha Y$$

   - The β slider in this case only effects the starting concentration of Y.

2. How does the response time change as you vary α?

Modeling the transcriptional response to a pulsed signal

For the next exercise we're going to assume that the transcriptional activator, X, is present as a pulse of variable duration. This might mimic an experiment where you were growing bacteria in a microfluidic device, and you introduced a small molecule that activated the transcription factor for a brief period of time, and then you quickly flowed the device through with fresh medium to deactivate the response. As before, the regulatory networks is $X \longrightarrow Y$. The rate of change of Y is given by:

$$\frac{dY}{dt} = \beta\theta(X > K) - \alpha Y$$

1. Run the `simple-pulse.py` script.

2. There are four sliders, representing:

   a) The length of the X pulse
   b) The threshold K
   c) β
   d) α

3. Explore the different parameters to get a sense of how they effect the behavior of the system.

Autoregulation

We're now going to contrast simple regulation, of the type $X \longrightarrow Y$, with two types of autoregulation: 1) negative autoregulation ($Y \dashv Y$); and 2) positive autoregulation ($Y \longrightarrow Y$).

We will model autoregulation as follows:

$$\frac{dY}{dt} = g(X > K) - \alpha Y$$

where:

$$g(X) = \begin{cases} \beta_0, & \text{if } X < K; \\ \beta_1, & \text{otherwise.} \end{cases}$$

Simple regulation is the case where $\beta_0 = \beta_1$; negative autoregulation corresponds to $\beta_0 > \beta_1$; positive autoregulation is when $\beta_0 < \beta_1$

1. Run the `autoreg.py` script.
2. There are four sliders, representing:
    a) The threshold K when autoregulation kicks in
    b) The initial growth rate, β_0
    c) The growth rate after feedback starts, β_1
    d) The decay rate, α
3. As long as the threshold K is exceeded, which parameters determine the final steady state of the system?
4. Keeping α and β_1 constant, vary β_0 so that it is both less than $beta_1$ (positive autoregulation) and greater than $beta_1$ (negative autoregulation).
    a) How does negative autoregulation effect the response time of the system?
    b) How does positive autoregulation effect the response time of the system?