The Parts and Compositors Framework

Using MATLAB and ODEs to understand the behavior of multi-part systems

(Kristjan) Eerik Kaseniit (eerik@mit.edu), Jacob Becraft (jbecraft@mit.edu)

Massachusetts Institute of Technology
20.305 Principles of Synthetic Biology

September 25, 2014

http://web.mit.edu/20.305/www/tutorial/

Differential equations – the usual tool

- What are they?
  Equations relating variables with their derivatives.
  Examples:

$$\begin{matrix} & \textit{Differentiation} \rightarrow \\ & \leftarrow \textit{Integration} \end{matrix}$$

Equation	*Differential equation*
$x(t) = c$	$\dot{x} = 0 \Leftrightarrow \frac{dx}{dt} = 0$
$x(t) = ct + d$	$\dot{x} = c$
$x(t) = ct^2 + d$	$\dot{x} = 2ct$

This is all pretty "mathematical". How can we use diff eqs to describe biological processes, chemical reactions?

Setting up differential equations

- Let's consider a chemical reaction:

$$E + S \underset{k_r}{\overset{k_f}{\rightleftharpoons}} ES$$

  Reaction: an enzyme E binds a substrate S to form an enzyme-substrate complex ES.

- Three species in this system, so we'll have three equations describing their *concentrations* changing over time:

$$\frac{d[E]}{dt} = -k_f[E][S] + k_r[ES]$$

$$\frac{d[S]}{dt} = -k_f[E][S] + k_r[ES]$$

$$\frac{d[ES]}{dt} = k_f[E][S] - k_r[ES]$$

  What are the units of $[E], d[S]/dt, k_f, k_r$?

Reading a differential equation

$$\frac{d[ES]}{dt} = k_f[E][S] - k_r[ES]$$

- *What are we describing?*
  The rate of change of the concentration of ES.
- *How many molecules come together to form ES?*
  There are two concentration terms in the forward reaction – so two.
- *Suppose that a second substrate molecule must be present for the complex to form – how would the equation change?*
  It would be $d[ES]/dt = k'_f[E][S]^{\mathbf{2}} - k'_r[ES_2]$
- *What increases the rate of change?*
  Increasing $k_f, [E], [S]$; decreasing $k_r, [ES]$.

Predicting using a differential equation

$$\frac{d[ES_n]}{dt} = k_f[E][S]^n - k_r[ES_n]$$

- Suppose we start with no ES_n, $[E](t = 0) = [E]_0$, $[S](t = 0) = [S]_0$ and we don't make any new E, S. After long enough time, what is the *steady-state concentration* of ES_n?

  Hint: Steady-state $\Rightarrow$ rate of change is zero.

Predicting using a differential equation

$$\frac{d[ES_n]}{dt} = k_f[E][S]^n - k_r[ES_n]$$

$$\frac{d[ES_n]}{dt} = 0 \Rightarrow k_f[E][S]^n = k_r[ES_n]$$

$$[ES_n] = \frac{k_f}{k_r}[E][S]^n$$

Is this it? What are $[E], [S]$? Set:

$[E]_{total} = [E] + [ES_n], \quad [S]_{total} = [S] + [ES_n] \approx [S]$ (if $[S]_0 \gg [E]_0$)

$$\frac{k_r}{k_f}[ES_n] = ([E]_{total} - [ES_n])[S]^n_{total}$$

$$[ES_n]\left(\frac{k_r}{k_f} - [S]^n_{total}\right) = [E]_{total}[S]^n_{total}$$

$$\boxed{[ES_n]_{steady-state} = \frac{[S]^n_{total}}{\frac{k_r}{k_f} - [S]^n_{total}}[E]_{total}}$$

Predicting using a differential equation

$$[ES_n]_{steady-state} = \frac{[S]^n_{total}}{\frac{k_r}{k_f} - [S]^n_{total}}[E]_{total}$$

But $[E]_{total} = E_0, [S]_{total} = S_0$:

$$[ES_n]_{steady-state} = \frac{[S]^n_0}{\frac{k_r}{k_f} - [S]^n_0}[E]_0$$

Hill term:

$$\boxed{\frac{X^n}{K^n + X^n}}$$

n – "cooperativity", Hill coefficient; K – a characteristic constant (X at which the fraction is $1/2$)
Limits as $X \to 0$ or $X \to \infty$? 0, 1.

Describing a biological system

The Central Dogma!

$$\begin{array}{ccccc} & & \emptyset & & \emptyset \\ & & \uparrow k_{mdeg} & & \uparrow k_{pdeg} \\ \text{DNA} & \xrightarrow{k_{txn}} & \text{mRNA} & \xrightarrow{k_{tln}} & \text{Protein} \end{array}$$

Note: this is more of a diagram than a set of equations.

$$\frac{d[mRNA]}{dt} = k_{txn}[DNA] - k_{mdeg}[mRNA]$$

$$\frac{d[Protein]}{dt} = k_{tln}[mRNA] - k_{pdeg}[Protein]$$

What are we leaving out? E.g. no cell division (dilution & replication), are neglecting small-number effects.

Something more interesting: modeling a repressor

Step 1: Model repressor-DNA binding.
Sounds a bit like enzyme-substrate binding...?

Step 2: Apply central dogma.
Ah! I know this!

Modeling a repressor: repressor-DNA binding

$$nR + DNA \underset{k_r}{\overset{k_f}{\rightleftharpoons}} R_n : DNA$$

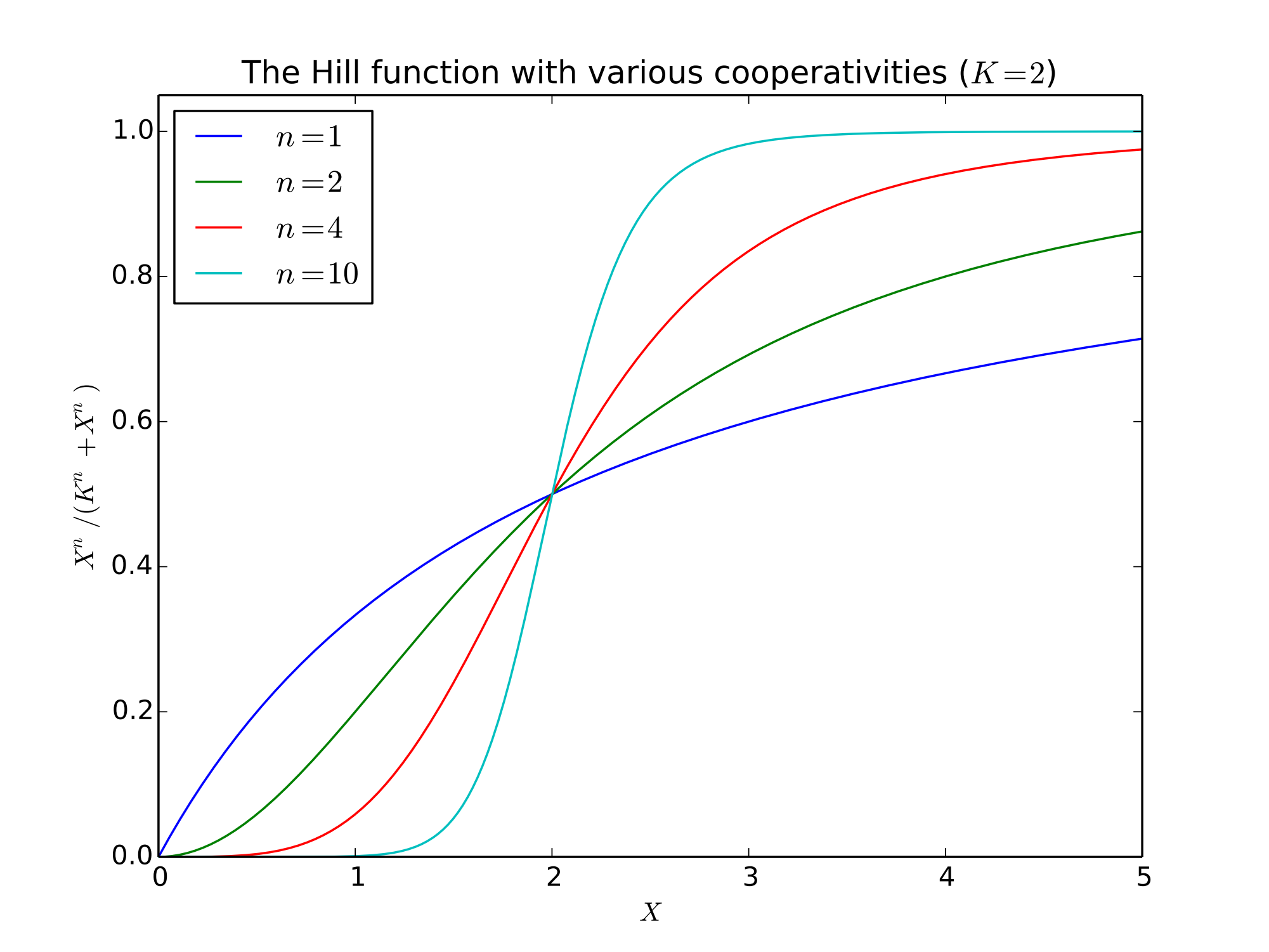
If we assume that equilibrium is achieved rapidly (not unreasonable), then we are at steady-state conditions:

$$\frac{d[R_n : DNA]}{dt} = 0$$

Thinking back to enzyme-substrate binding, we say that $[DNA] \ll [R]$ and so:

$$[R_n : DNA] = \frac{[R]^n}{K^n + [R]^n}[DNA] \qquad K = (k_r/k_f)^{1/n}$$

Note: because of the assumption above, the "enzyme" is the DNA and the "substrate" is the repressor if we think about the equations. $n = 2$ means roughly that the repressor dimerizes.

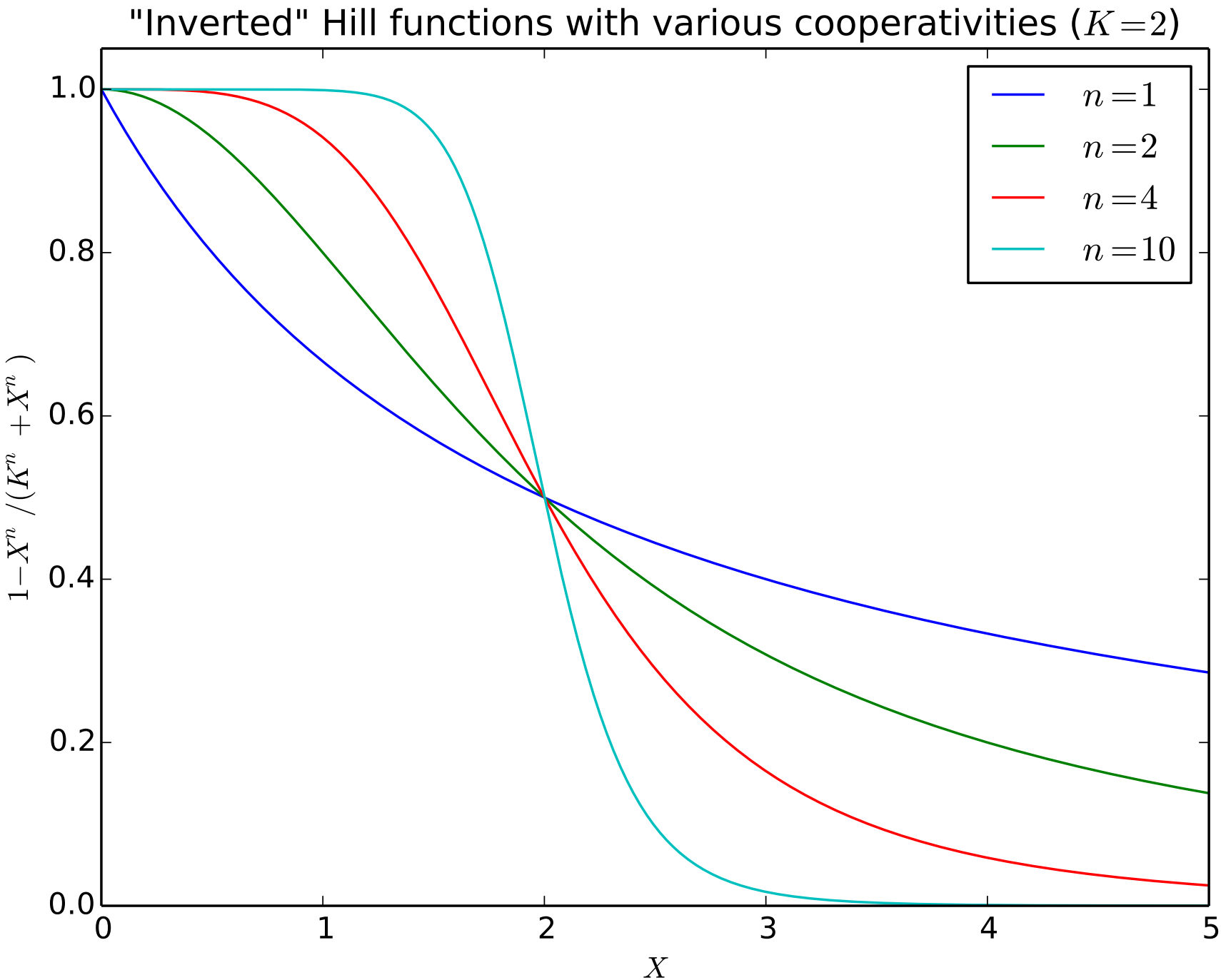

The Hill function with various cooperativities ($K=2$)
$n=1$
$n=2$
$n=4$
$n=10$
1.0
0.8
0.6
0.4
0.2
0.0
$X^n/(K^n+X^n)$
0
1
2
3
4
5
X

Modeling a repressor: relating to central dogma

Only DNA *not* bound the repressor can be transcribed:

$$\frac{d[mRNA]}{dt} = k_{txn}[DNA]_{tot}\left(1 - \frac{[R]^n}{K^n + [R]^n}\right) - k_{mdeg}[mRNA]$$

Fraction not bound by repressor


"Inverted" Hill functions with various cooperativities ($K=2$)
$1-X^n\ /(K^n\ +X^n\ )$
X
$n=1$
$n=2$
$n=4$
$n=10$
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5

Extending the model – how to describe the requirement for activation?

Hint:

A	B	A AND B	$A \cdot B$
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

$$\frac{d[mRNA]}{dt} = k_{txn}[DNA]_{tot}\left(1 - \frac{[R]^{n_R}}{K_R^{n_R} + [R]^{n_R}}\right) \cdot \left(\frac{[A]^{n_A}}{K_A^{n_A} + [A]^{n_A}}\right)$$

$$-k_{mdeg}[mRNA]$$

Numerical analysis of differential equations

Most interesting diff eqs are hard to solve analytically. This leaves two common options:

- Numerical integration
  Start with some initial conditions, and integrate little-by-little. E.g. `ode23s` in MATLAB.
- Stochastic simulations
  Reaction rates are like probabilities: throw weighted dice to figure out a) when the next reaction is happening and b) which one from a set of reactions occurs. E.g. Gillespie algorithm.

MATLAB tips and tricks

- Use up/down arrows to navigate history (Ctrl-R to search)
- Use `;` at the end of a statement to suppress output
- `disp('a')` will print out `a` (can use variables)
- `clear all` will clear the workspace, `clc` just the screen
- MATLAB over SSH from Athena: `ssh -XC -c blowfish -l YOUR_KERBEROS athena.dialup.mit.edu` (might need XQuartz for Macs)
- `help sqrt` will bring up the manual for the $\sqrt{}$ function
- `lookfor square` will search for "square" in docs ($\sim$ slow)
- `f = @(x) (x^2)` creates an "anonymous" function in a script, call e.g. `f(2)`
- `hold on` adds `plot()` results to current figure; `hold all` will also use new color/style automatically
- `ans` holds result of last statement

MATLAB tips and tricks

- `[1:5] == [1 2 3 4 5]`
- `A'` will transpose `A`
- `[1:5]' == [1; 2; 3; 4; 5]`
- `A(:, 2)` gives the second column of the matrix `A`
- `A(:, end)` gives the last column of the matrix `A`
- `...` lets you break up long lines
- Look at the `.m` files that make up our framework! They're short :)

Setup for the part-compositor framework

To download the part-compositor framework, run this once in MATLAB in your work directory:

```
urlwrite(['http://web.mit.edu/20.305/www/' ...
    'part_composition_setup.m'], ...
    'part_composition_setup.m');
rehash;
part_composition_setup('v2pre');
```

Note: for problem submissions, you must include this sort of statement at the top of the problem (always given in the problem template)

To download the scripts we will be looking at here, go to `http://web.mit.edu/20.305/www/tutorial/`

Off we go!

Parts are processes and compositors help connect them

Parts are processes

- **Inputs**: a subset of state variables
- **Outputs**: a set of rates of change of a possibly overlapping set of other state variables

Compositors aggregate effects of parts

- There is one compositor for each state variable of the system
- Multiple parts may act on the same state variable and thus on the compositor
- **Inputs**: certain rates from possibly many parts (to produce a change in the relevant state variable)
- **Output**: the state variable
- We call compositors by their state variable name

Example from class

Part: conversion of $M_1 \rightarrow M_2$ mediated by E_1 with rate law $k_{cat}[E_1]\frac{[M_1]}{K_m+[M_1]}$

Compositors: $\frac{d[E_1]}{dt}$ (named `E1`), $\frac{d[M_1]}{dt}$ (`M1`), $\frac{d[M_2]}{dt}$ (`M2`)

$$v^1_{E_1} = \frac{d[E_1]}{dt} = 0$$

$$v^1_{M_1} = \frac{d[M_1]}{dt} = -k_{cat}[E_1]\frac{[M_1]}{K_m + [M_1]}$$

$$v^1_{M_2} = \frac{d[M_2]}{dt} = k_{cat}[E_1]\frac{[M_1]}{K_m + [M_1]}$$

Initial conditions:

$$[E_1](t = 0\ s) = 10\ \mu M$$

$$[M_1](t = 0\ s) = 10\ \mu M$$

$$[M_2](t = 0\ s) = 0\ \mu M$$

Coding it up in MATLAB

Part: conversion of $M_1 \rightarrow M_2$ mediated by E_1 with rate law $k_{cat}[E_1]\frac{[M_1]}{K_m+[M_1]}$

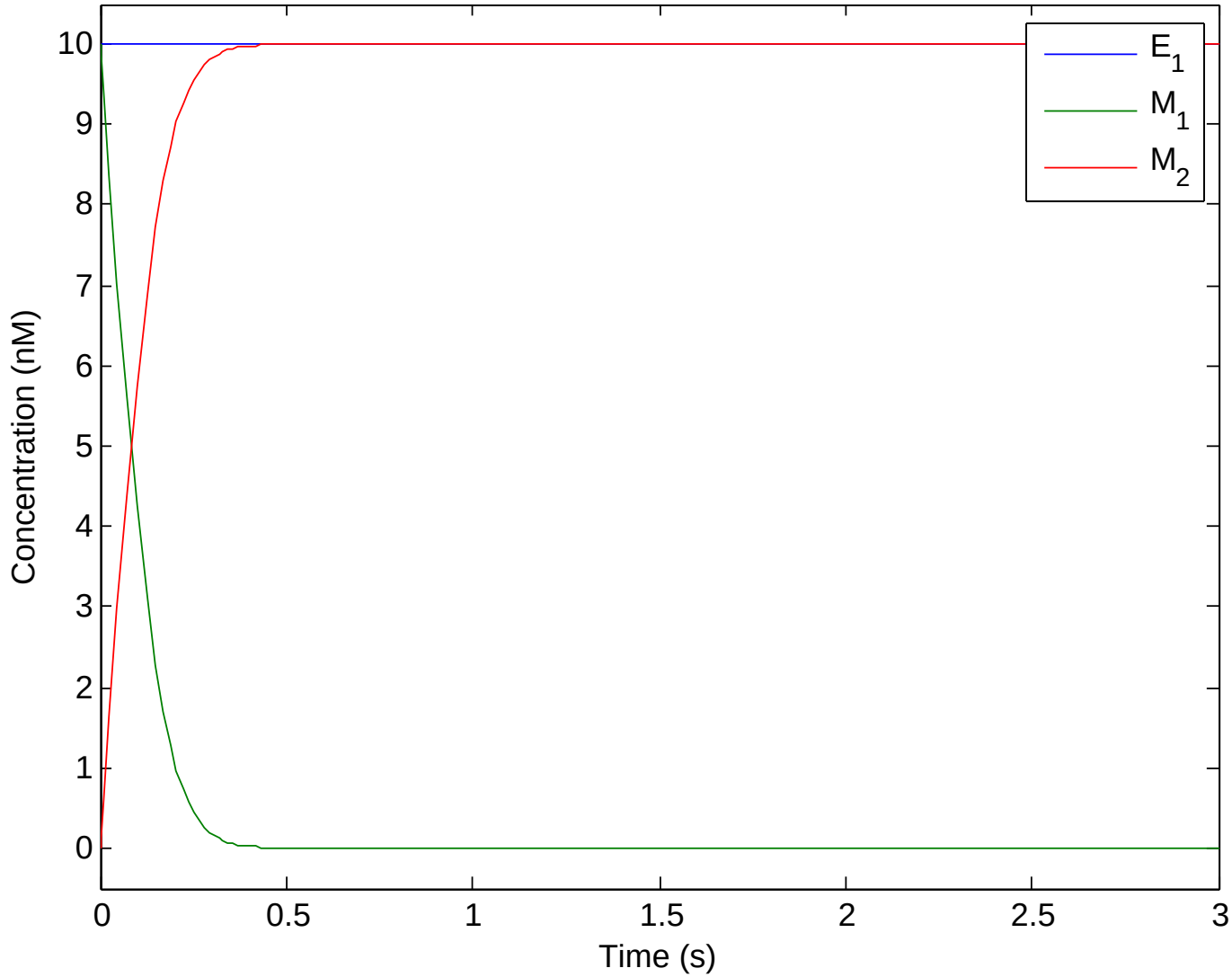
```
0001 sys1 = BioSystem();
0002
0003 % create compositors, set initial state
0004 E1 = Compositor('E1', 10); % units implied
0005 M1 = Compositor('M1', 10);
0006 M2 = Compositor('M2', 0);
0007
0008 sys1.AddCompositor(E1);
0009 sys1.AddCompositor(M1);
0010 sys1.AddCompositor(M2);
0011
0012 sys1.AddConstant('k_cat', 10); % units implied
0013 sys1.AddConstant('K_m', 5);
0014
0015 P1 = Part('M1-(E1)->M2', [M1 E1 M2], ...
0016            [ Rate('- k_cat * E1 * (M1 / (K_m + M1))') ... % M1
0017              Rate('0'), ... % E1
0018              Rate('  k_cat * E1 * (M1 / (K_m + M1))') ... % M2
0019            ]);
0020
0021 sys1.AddPart(P1);
0022
0023 [T, Y] = sys1.run([0 3]) % units implied
0024 plot(T, Y)
```

How can *I* run it?

Go to the folder where you've downloaded the framework as well as `sys1_simple_enzyme.m`

Then in your prompt:

```
>>  sys1_simple_enzyme
```


Concentration (nM)
Time (s)
E1
M1
M2
0
1
2
3
4
5
6
7
8
9
10
0
0.5
1
1.5
2
2.5
3

A mystery part in a new system

Part: a mystery process involving a single state variable, x. The behavior of x is governed by the equation

$$\frac{d^2x}{dt^2} - \mu(1 - x^2)\frac{dx}{dt} + x = 0$$

This state variable starts off with $x(0) = 1, \dot{x} = 0.5$

Compositors: ?!? how do I define a compositor when we take second derivatives ?!?
Answer: Substitution! use $y = \dot{x}$
Now we have a state variable that's not a physical thing, but that's OK, it's all make-believe anyway!

$$v_x^1 = \frac{dx}{dt} = y$$

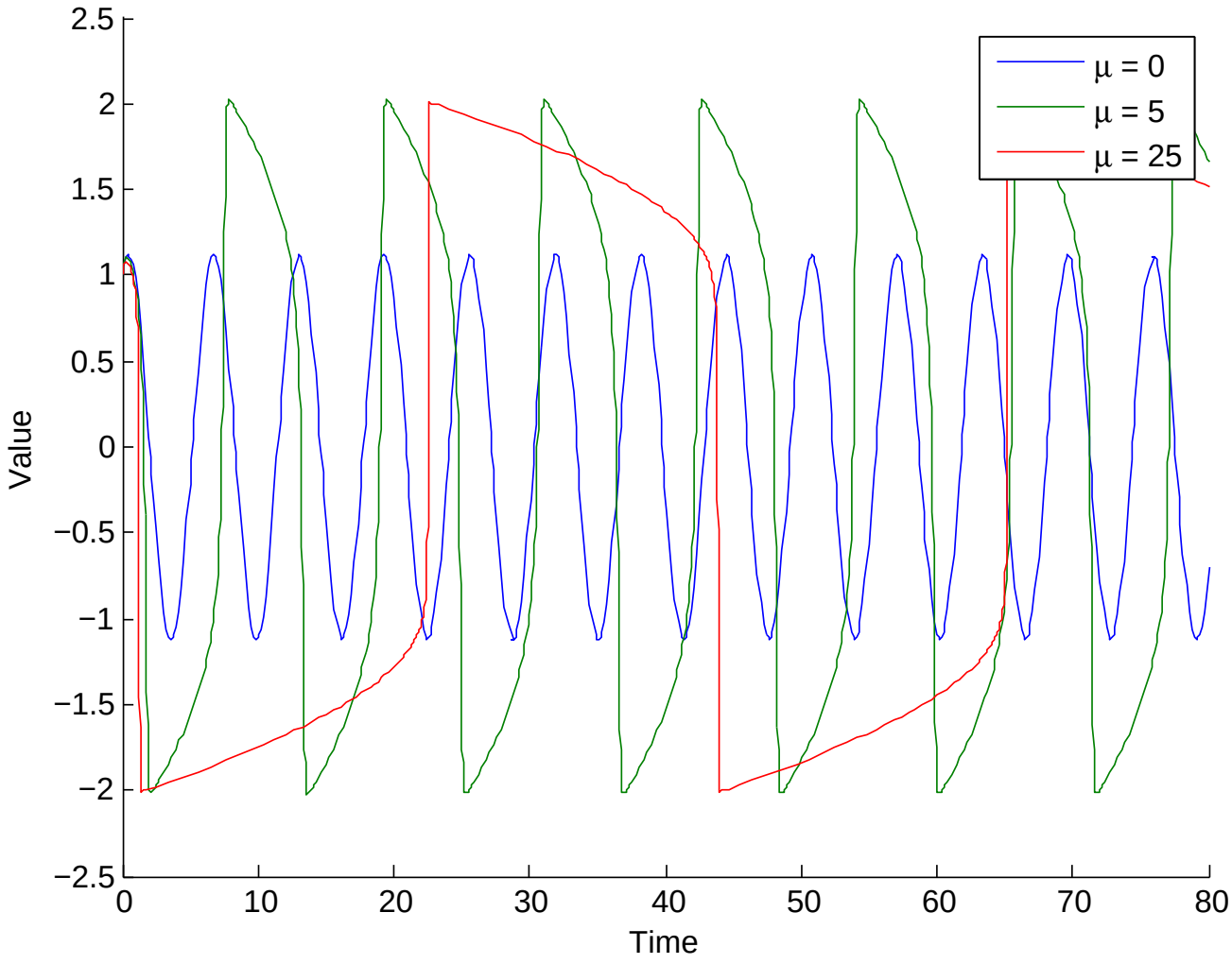
$$\frac{dy}{dt} - \mu(1 - x^2)y + x = 0 \Rightarrow v_y^1 = \frac{dy}{dt} = \mu(1 - x^2)y - x$$

Simulating our mystery part

$$v_x^1 = y$$
$$v_y^1 = \mu(1 - x^2)y - x$$

```
0001 sys2 = BioSystem();
0002
0003 x = sys2.AddCompositor('x', 1);
0004 y = sys2.AddCompositor('y', 0.5);
0005
0006 sys2.AddConstant('mu', 5);
0007
0008 sys2.AddPart(Part('Van der Pol wibbly-wobbly-timey-wimey', [x y], ...
0009     [ Rate('y') ... % x
0010       Rate('mu * (1 - x^2) * y - x'), ... % y
0011     ]));
0012
0013 figure();
0014 hold all; % plot() will use same figure, different colors
0015 time_interval = [0 80];
0016 for mu = [0 5 25]
0017     sys2.ChangeConstant('mu', mu)
0018     [T, Y] = sys2.run(time_interval);
0019     plot(T, Y(:, 1))
0020 end
0021 legend('\mu = 0', '\mu = 5', '\mu = 25')
0022 xlabel('Time')
0023 ylabel('Value')
```

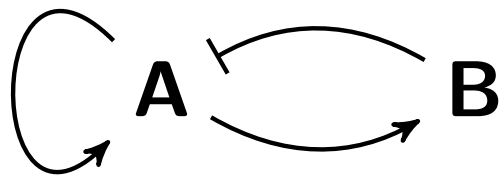
What does this wibbly-wobbly-timey-wimey part do then?!



Ah, I want a wibbly-wobbly-timey-wimey thing!

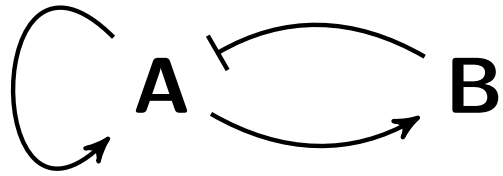
You're in luck! There are many ways to implement oscillators using gene-regulatory elements.

Here's one way ("activator-inhibitor" topology):



A circuit with this abstract topology was implemented e.g. by Stricker *et al.* (Nature 2008), Prindle *et al.* (Nature 2012, http://go.nature.com/lrOtKQ)

Decomposing the activator-inhibitor topology

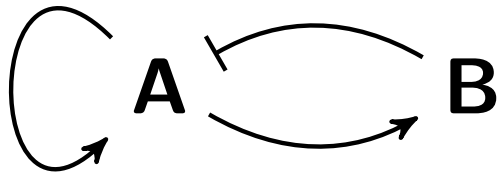


Parts: transcription of A, B, translation of A, B, degradation of A, m_A, B, m_B

Compositors: $\frac{dm_A}{dt}, \frac{dm_B}{dt}, \frac{dA}{dt}, \frac{dB}{dt}$

General model: fraction of promoter bound by X is given by $\frac{X^n}{X^n+K_X^n}$ where K_X is the concentration of X where the fraction bound is 0.5 and n is a characteristic constant, roughly how many molecules of X come together before binding.

Writing out the rates for parts in the activator-inhibitor topology



Parts: transcription of A, B (1-2), translation of A, B (3-4), degradation of A, m_A, B, m_B (5-8)

Rates:

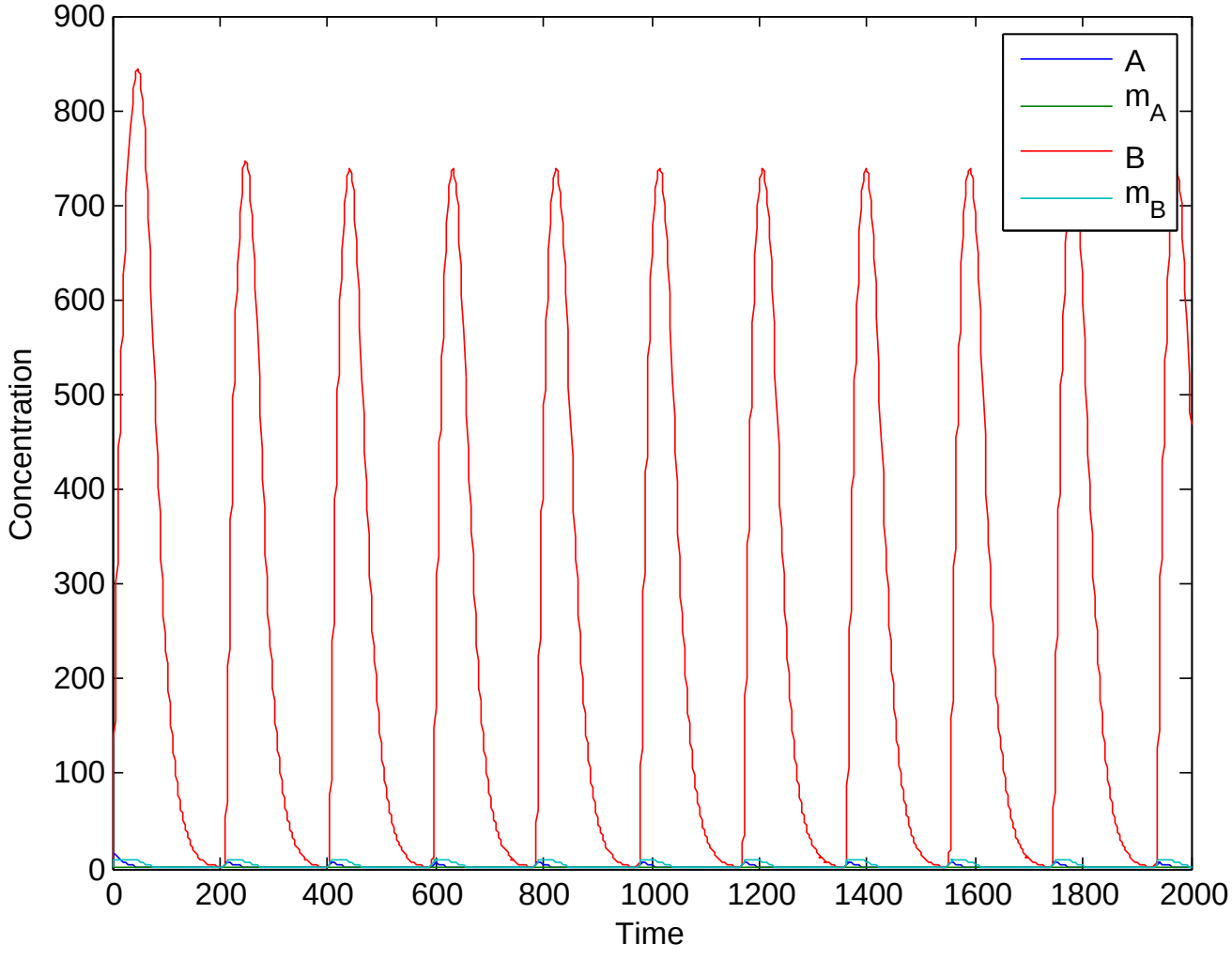
$$v^1_{m_A} = k_{tx} \cdot \frac{A^n}{K_A^n + A^n} \cdot \frac{K_B^n}{K_B^n + B^n} \qquad v^2_{m_B} = k_{tx} \cdot \frac{A^n}{K_A^n + A^n}$$

$$v^3_A = k_{tln} \cdot m_A \qquad v^4_B = k_{tln} \cdot m_B$$

$$v^5_A = -k_{pdeg} \cdot A \qquad v^6_{m_A} = -k_{mdeg} \cdot m_A$$

$$v^7_B = -k_{pdeg} \cdot B \qquad v^8_{m_B} = -k_{mdeg} \cdot m_B$$

```
0001 sys3 = BioSystem();
0002
0003 A = sys3.AddCompositor('A', 10); mA = sys3.AddCompositor('mA', 1);
0004 B = sys3.AddCompositor('B', 10); mB = sys3.AddCompositor('mB', 1);
0005
0006 sys3.AddConstant('k_tx', 5); sys3.AddConstant('k_tl', 5);
0007 sys3.AddConstant('k_mdeg', 0.5); sys3.AddConstant('k_pdeg', 0.05);
0008 sys3.AddConstant('K_A', 1); sys3.AddConstant('K_B', 2);
0009 sys3.AddConstant('n', 2);
0010
0011 sys3.AddPart(Part('Transcription of A', [mA], ...
0012     [ Rate('k_tx * A^n / (K_A^n + A^n) * K_B^n / (K_B^n + B^n)') ]));
0013
0014 sys3.AddPart(Part('Transcription of B', [mB], ...
0015     [ Rate('k_tx * A^n / (K_A^n + A^n)') ]));
0016
0017 sys3.AddPart(Part('Translation of A', [A], ...
0018     [ Rate('k_tl * mA') ]));
0019 sys3.AddPart(Part('Translation of B', [B], ...
0020     [ Rate('k_tl * mB') ]));
0021
0022 sys3.AddPart(Part('Degradation of A', [A], ...
0023     [ Rate('- k_pdeg * A') ]));
0024 sys3.AddPart(Part('Degradation of mA', [mA], ...
0025     [ Rate('- k_mdeg * mA') ]));
0026 sys3.AddPart(Part('Degradation of B', [B], ...
0027     [ Rate('- k_pdeg * B') ]));
0028 sys3.AddPart(Part('Degradation of mB', [mB], ...
0029     [ Rate('- k_mdeg * mB') ]));
0030
0031 figure();
0032 [T, Y] = sys3.run([0 2000]);
0033 plot(T, Y);
0034 legend('A', 'm_A', 'B', 'm_B');
0035 xlabel('Time');
0036 ylabel('Concentration');
```


900
800
700
600
500
400
300
200
100
0
Concentration
0
200
400
600
800
1000
1200
1400
1600
1800
2000
Time
A
m_A
B
m_B

A tasty treat for suffering through these slides

http://vimeo.com/37189162
A microfluidic device is filled with bacteria that express a synchronized oscillator circuit. The cells are constantly dividing and are washed away at the bottom to make room. Synchronization is achieved using a small diffusible molecule.

http://vimeo.com/33812959
A set of microfluidic devices share the same air and are furthermore coupled using H_2O_2.
Prindle *et al.* A sensing array of radically coupled genetic 'biopixels'. Nature 2012, http://go.nature.com/IrOtKQ

Modeling helped figure out the right conditions for getting these oscillations!