ABE 591S Principles of Systems and Synthetic Biology

MATLAB Crash Course

Kevin Solomon Ph.D. | Purdue | F2017

Objective

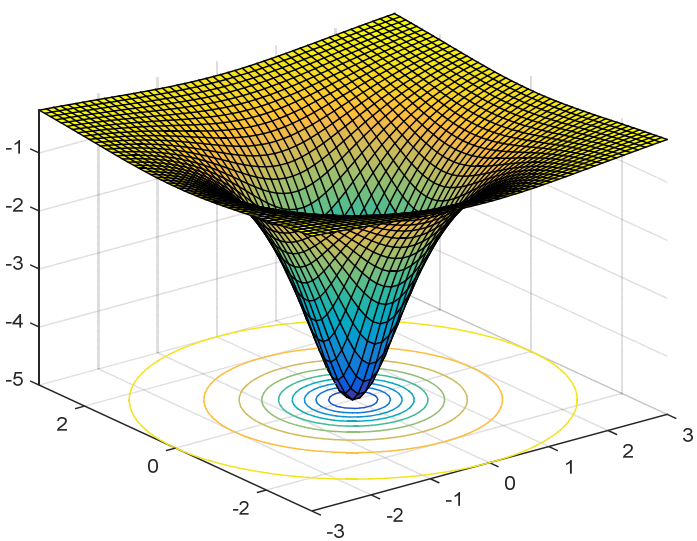
Get comfortable playing with MATLAB & demonstrate how we can apply it to biological problems ….

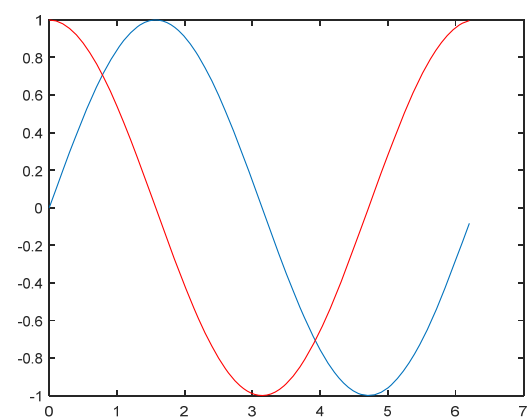
- The MATLAB environment
- Entering & Manipulating Data
- Plotting Data
- m-files and functions
- MATLAB Programming
- Solving ODEs
- Biological oscillators (Elowitz & Leibler, *Nature,* 2000).

What is MATLAB or *mat*rix *lab*oratory?

Powerful platform for mathematical and scientific computation

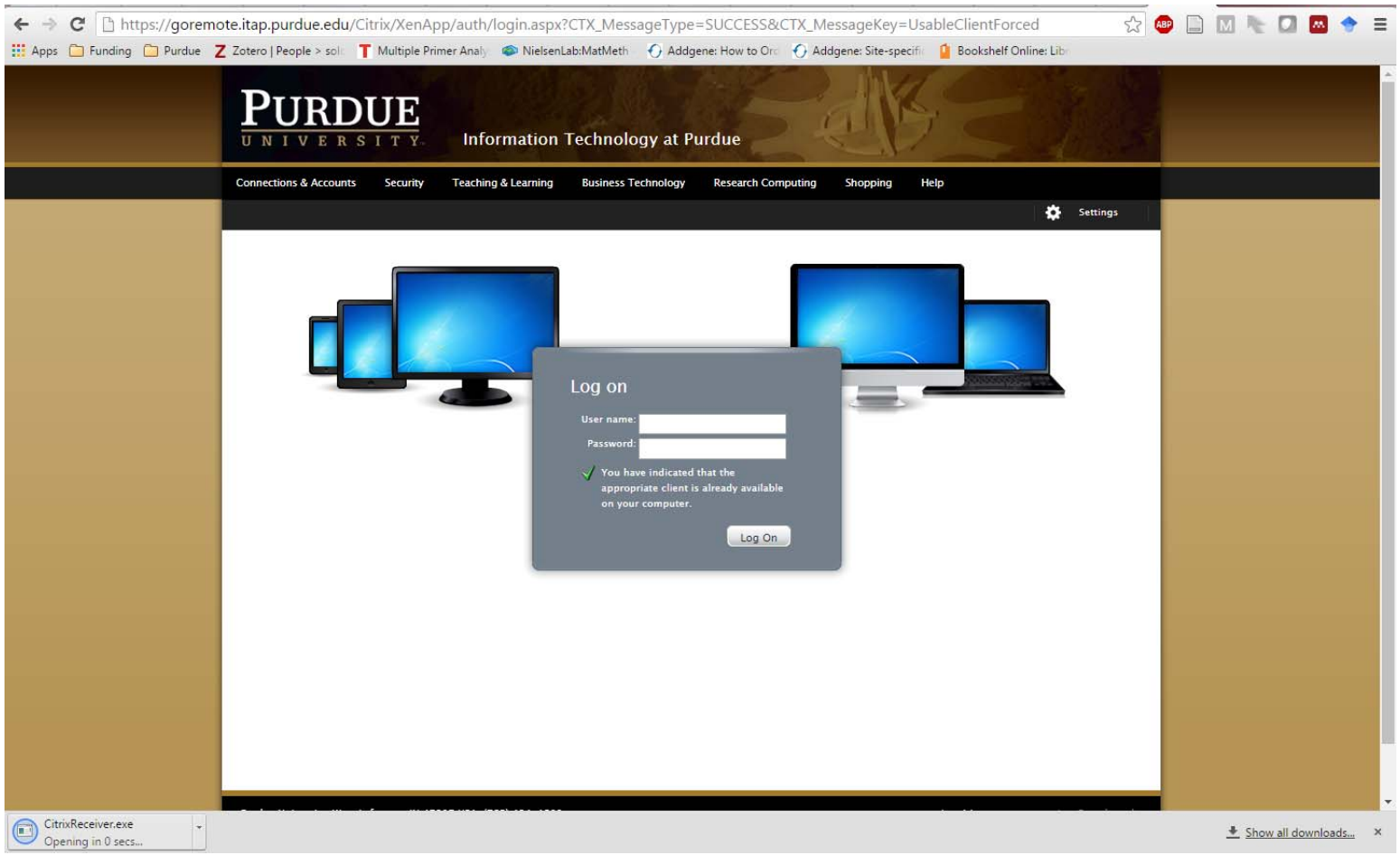
- Easy syntax
- Many modules/functions for most applications
- Solves problems *numerically*
- Standard in education and industry





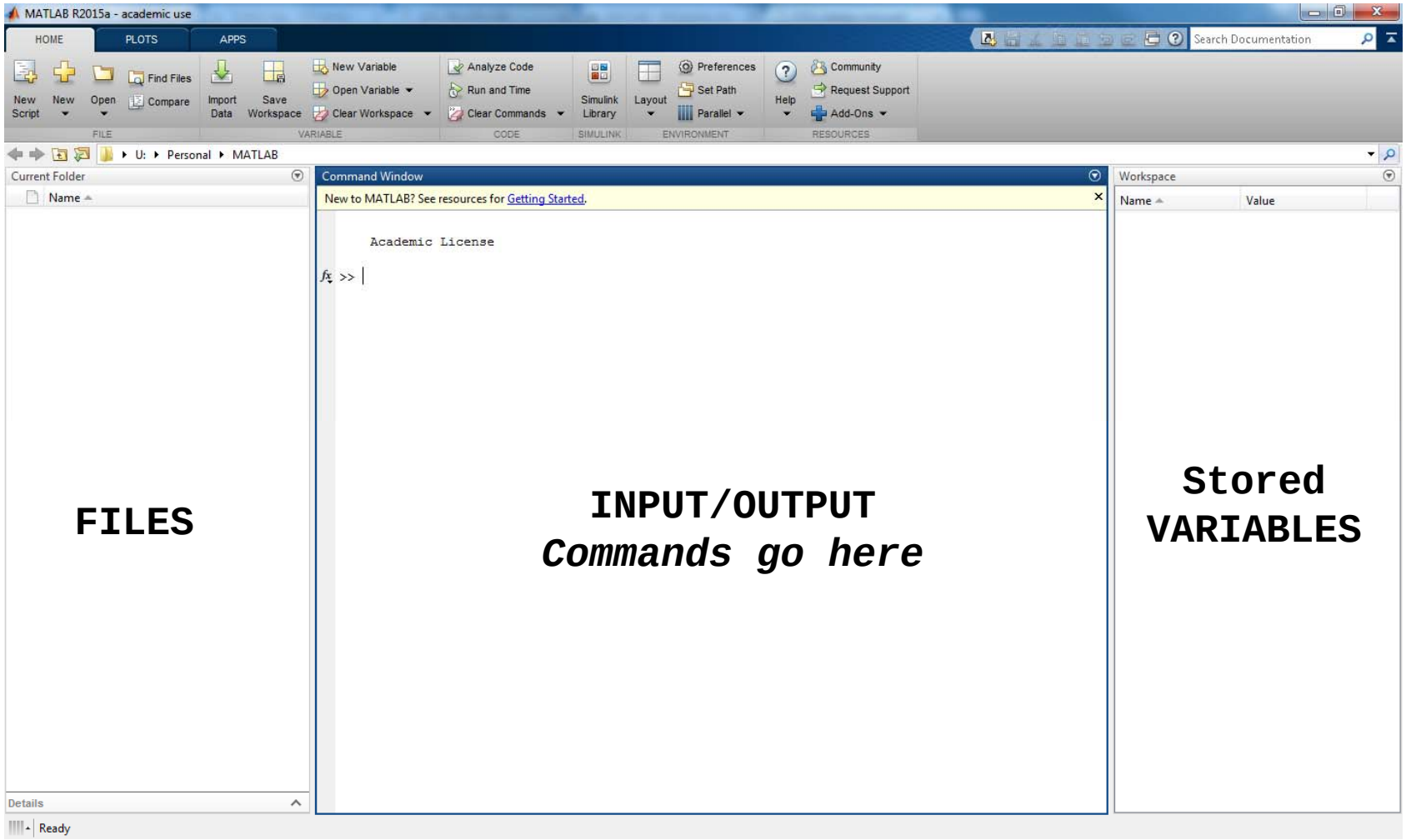
How to get MATLAB?

Any iTAP/ECN computer or at home via iTAP goRemote - https://goremote.itap.purdue.edu/

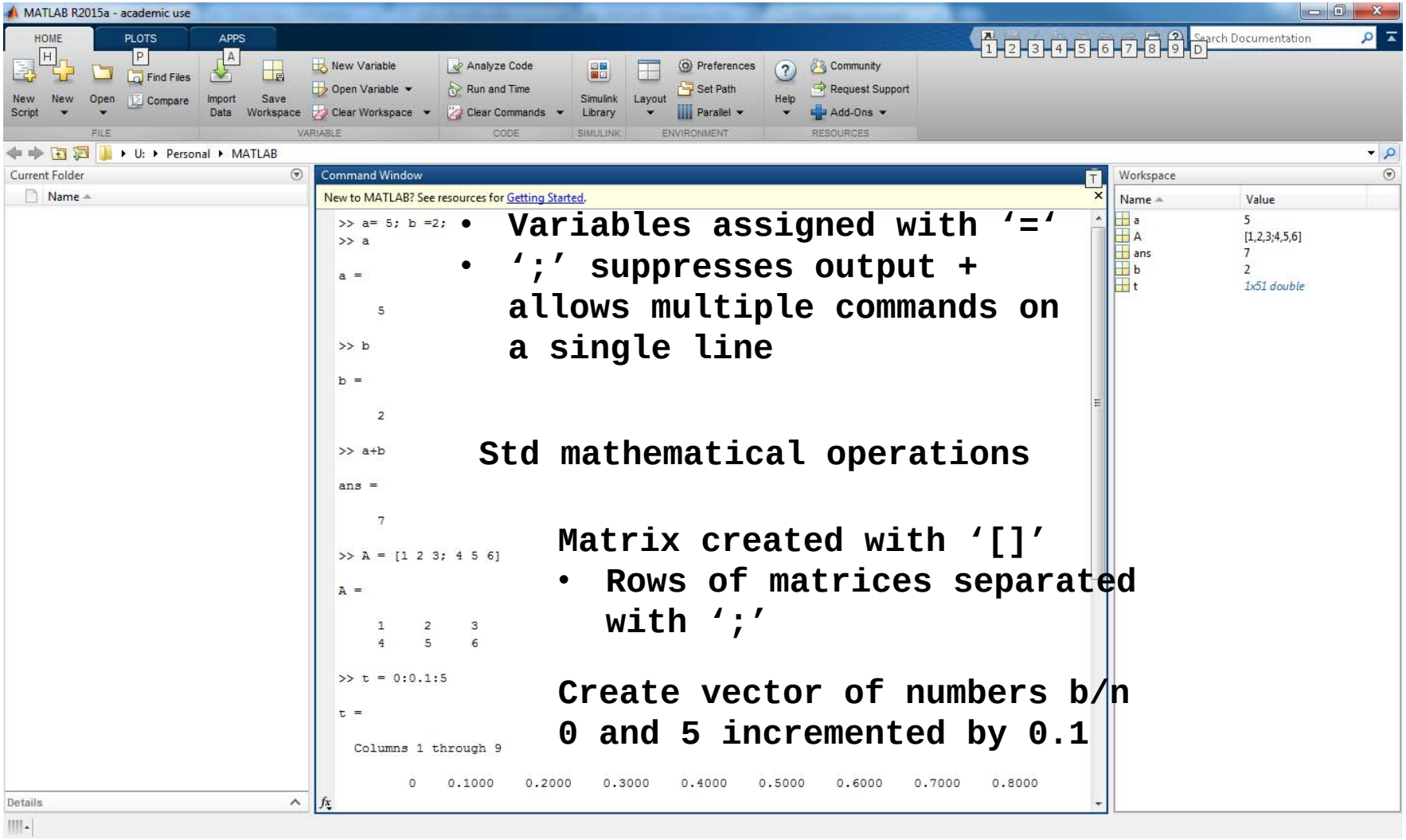


The MATLAB environment

Default IDE below



Entering & Manipulating Data



- Variables assigned with '='
- ';' suppresses output + allows multiple commands on a single line

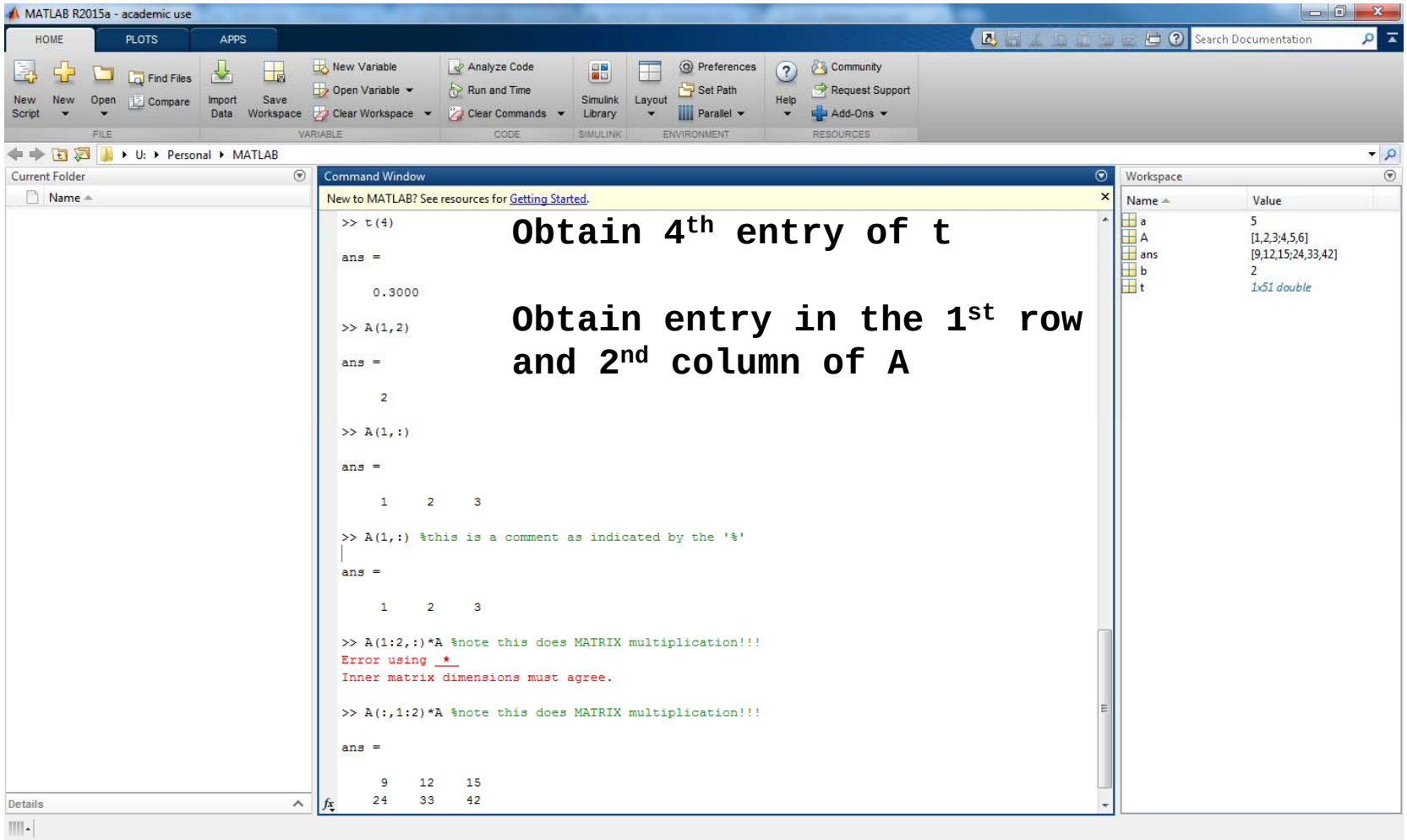
Std mathematical operations

Matrix created with '[]'

- Rows of matrices separated with ';'

Create vector of numbers b/n 0 and 5 incremented by 0.1

Entering & Manipulating Data



Obtain 4th entry of t

Obtain entry in the 1st row and 2nd column of A

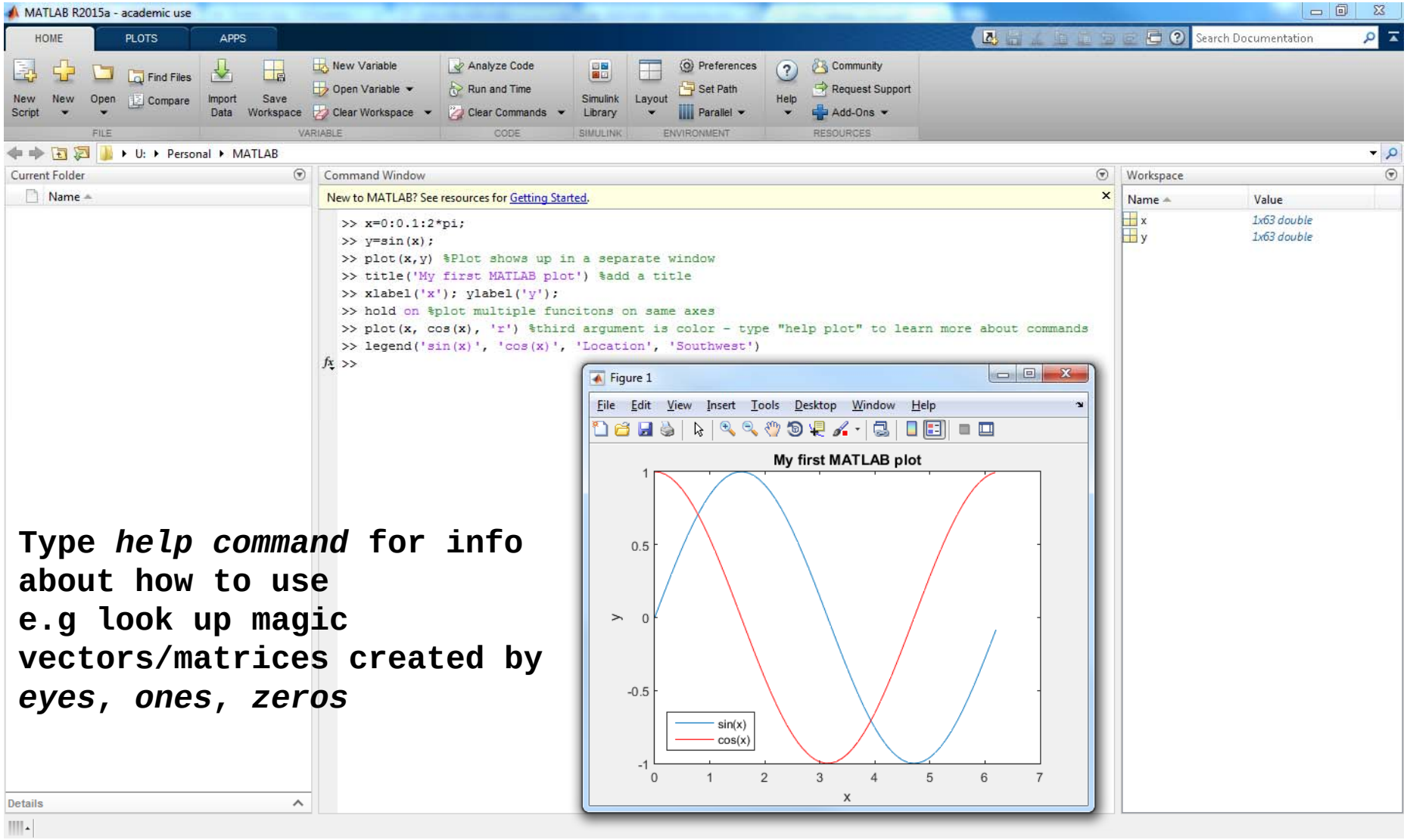
Manipulating data

- The `log` command is natural log. $\log_{10}$ = `log10`
- e^5 is `exp(5)`
- MATLAB recognizes `i` and `j` in complex numbers
- Arithmetic operations are by default matrix operations!
  - Can calculate determinants and eigenvalues

$$\underset{A}{\begin{bmatrix} a & b \\ c & d \end{bmatrix}} \times \underset{B}{\begin{bmatrix} e & f \\ g & h \end{bmatrix}} = \underset{C}{\begin{bmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{bmatrix}}$$

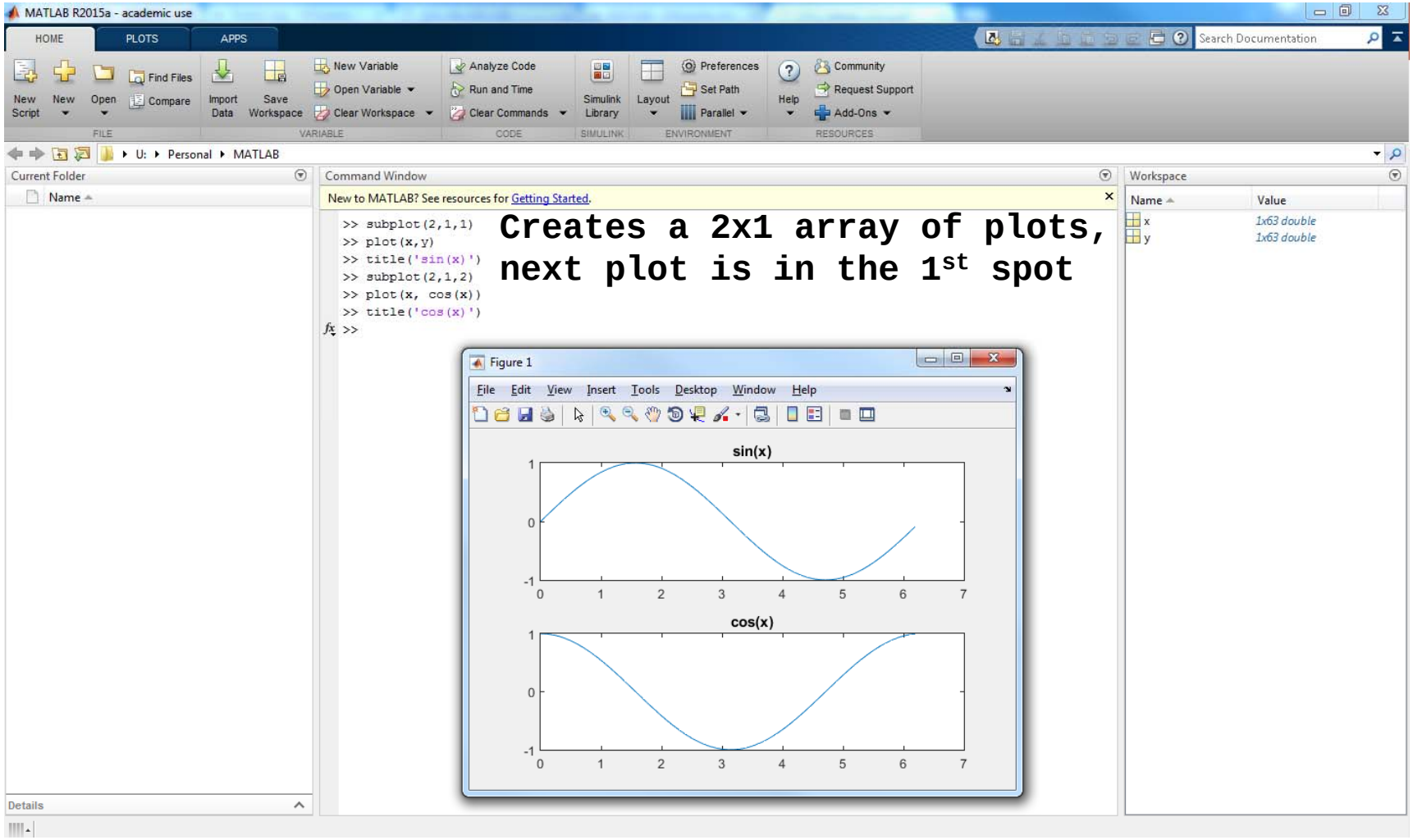
A, B and C are square metrices of size N x N
a, b, c and d are submatrices of A, of size N/2 x N/2
e, f, g and h are submatrices of B, of size N/2 x N/2

Visualizing Data



Type *help command* for info about how to use
e.g look up magic vectors/matrices created by *eyes*, *ones*, *zeros*

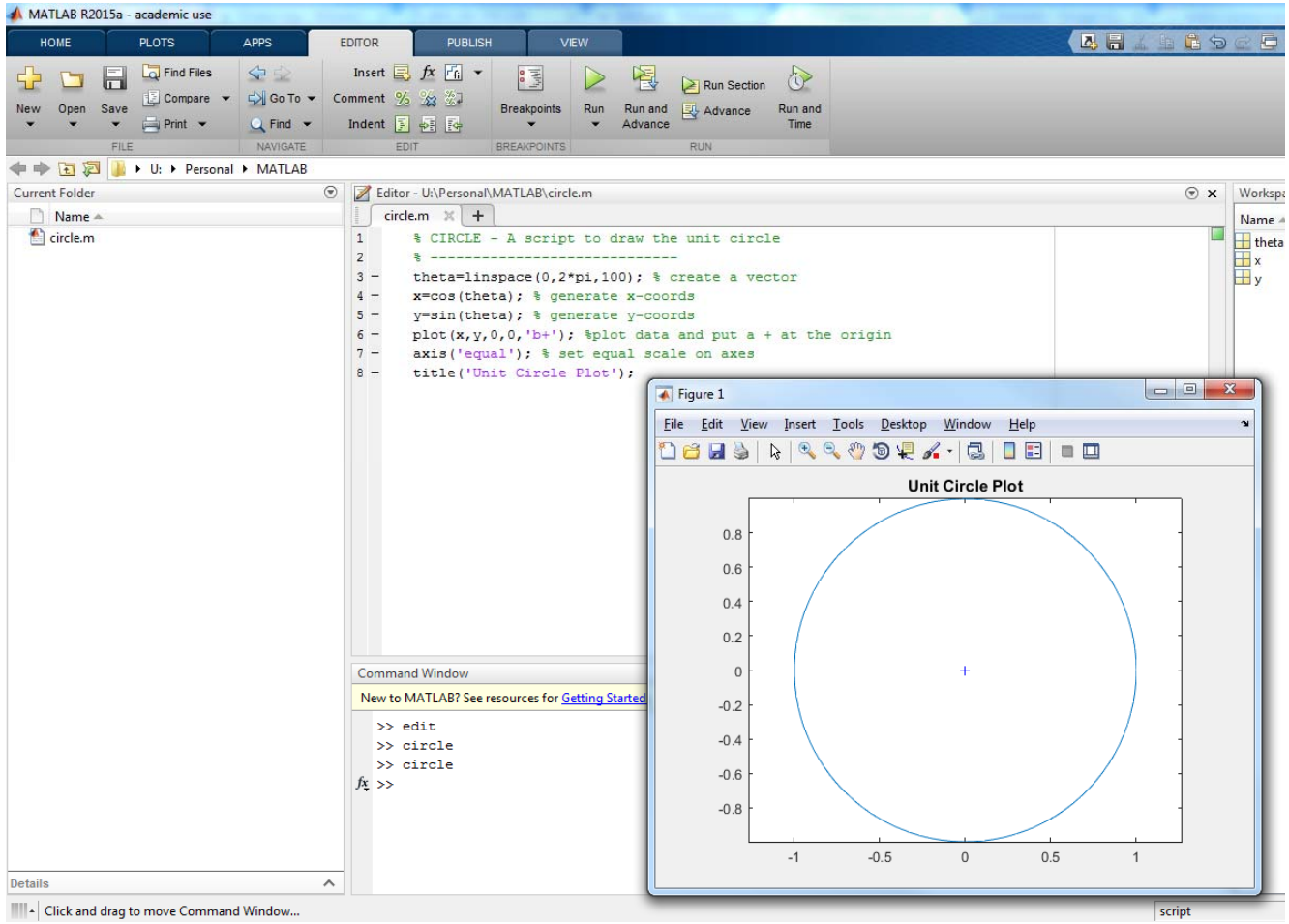
Visualizing Data



m-files and Functions

m-files allow you to write custom functions or <u>scripts of commands</u>

- Call editor with `edit` or write m-file in fave text editor
- Call saved m-file in current directory from command window with $script_name
- Mathematical functions have unique requirements – see e.g.



MATLAB Programming

MATLAB recognizes std. programming techniques

For Loops

```
% print out all values of 5m for integer
values of 1<= m <=100

for m = 1:100

    num = m*5

end
```

While Loops:

```
%find all powers of 2 below 100. will
iterate as long as this is true

k=1; num = 1; v = 1;

while num < 100

    num = 2^k;

    v = [v; num];

    k= k+1;

end
```

If statements

```
k = 6; m = 21

if k>5

    n = k;

elseif (k>1) &(m==20)

    n = 5*k+m;

else

    n=1;

end
```

Comparators

AND	a&b
OR	a|b
Not Equal	a~=b
Equal	a==b

Greater than a>b
Greater than or equal a>=b

Solving ODEs

e.g. $\frac{dy}{dt} = y$; $0 \le t \le 2$; $y(0) = 1$ ← Initial/boundary value – need one for every variable if first order

Define system of equations as a function (in an m-file, as follows)

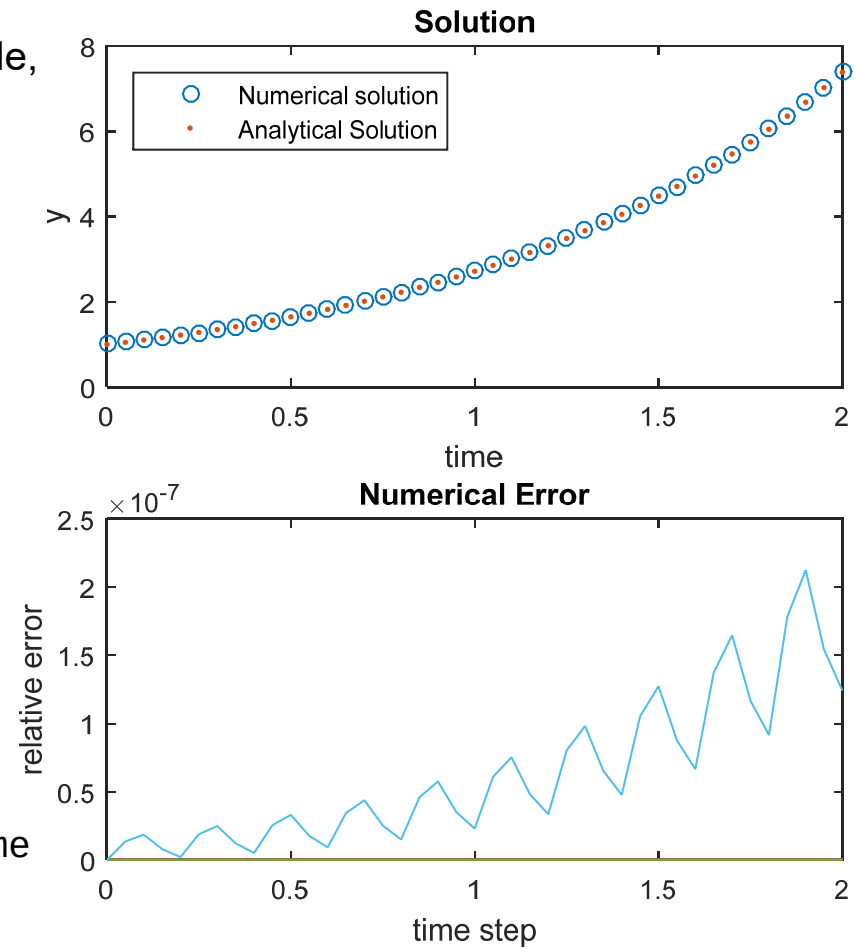
```
function dy =myodes(t, y);
dy = y;
%must save as function name e.g.
myodes.m
```

← **Function name**

List arguments when calling

Solve system in script or command line with

```
[t y] = ode45(@(t,y) myodes(t,y), [0
2], [1]);
subplot(2,1,1);
plot(t, y, 'o', t, exp(t), '.')
title('Solution'); xlabel('time');
ylabel('y')
legend('Numerical solution',
'Analytical Solution',
'Location','Northwest')
subplot(2,1,2);
plot(t, (y-exp(t))/exp(t))
title('Numerical Error'); xlabel('time
step'); ylabel('relative error')
```

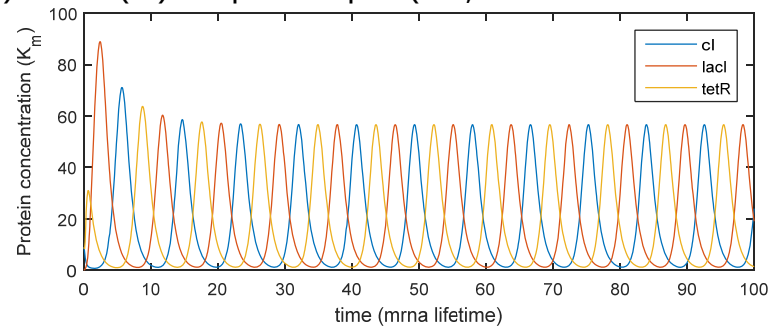


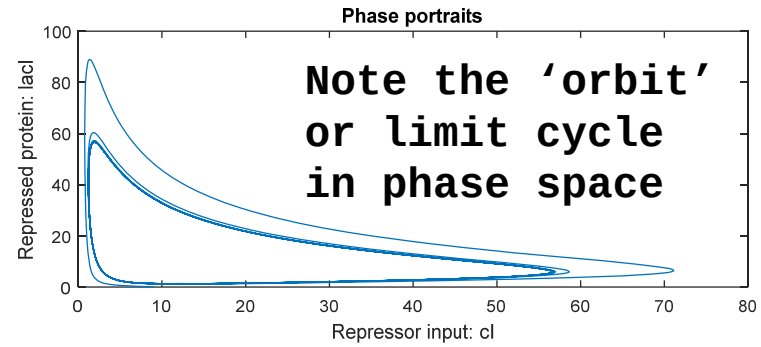
Repressilator

```
function dx = repressilator(t,x, n, b , a, a0)
%x(1) to x(3) = m1 to m3 (lacI, tetR, cI), x(4) to x(6) = p1 to p3 (cI,
%lacI, tetR)

dx(1) = -x(1)+a/(1+x(4)^n)+a0;
dx(2) = -x(2)+a/(1+x(5)^n)+a0;
dx(3) = -x(3)+a/(1+x(6)^n)+a0;
dx(4) =-b*(x(4)-x(3));
dx(5) = -b*(x(5) - x(1));
dx(6) = -b*(x(6)-x(2));

dx=dx';
```





Solve the system with n = 2, β = 5, α = 200, $\alpha_0 = \alpha * 10^{-3}$ and initial values of [1 1 1 9 0 9] to generate the plots above

Linear Stability Analysis

From our analysis of the non-dimensionalized system

$$\frac{dm_i}{dt} = \alpha_0 + \frac{\alpha}{1 + p_j^n} - m_i$$

$$\frac{dp_i}{dt} = -b(p_i - m_i)$$

What conditions were **necessary but not sufficient** for stable oscillations? What must be true of α, α_0, b, n?

Explore the repressilator on your own

Use your code to explore the parameter space of the Hill equation and see its impact on the dynamic behavior. Fig 1B may guide your parameter selections

1. Is the oscillator sensitive to the initial values of the problem? i.e. are the oscillations stable? *Stable systems will ALWAYS return to their stationary values regardless of initial trajectory*

2. If the system is made less cooperative (n~1), does the system oscillate?

3. If you increase b to 1000, will the system oscillate?

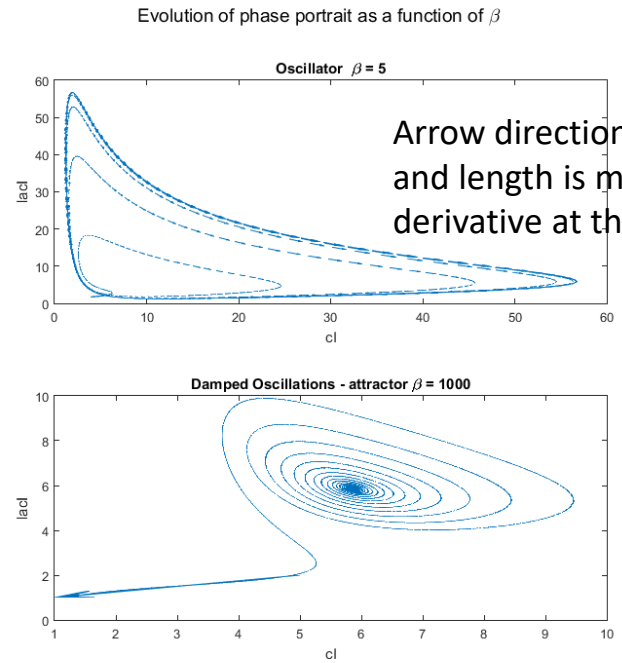
4. What is the impact of the background and full expression strength (α_0 , α)?

5. How does the behavior of this system compare to the simplified one described in class? [Assume mRNA has rapid equilibrium, no leaky expression]

Visualizing trajectories

How does the system evolves as a function of time as parameters or initial conditions vary?

```
n = 2; %hill coefficient
b = [5 1000];
a = 200;
a0 = a*1e-3; %region C of Fig 1B
titles ={Oscillator \beta = 5', 'Damped
Oscillations - attractor \beta = 1000'};
clf
for k = 1:size(b,2)
    subplot(2,1,k)
    clearvars x u t
    %determines the value of all repressors
over time
    [t x] = ode45(@(t, x) repressilator(t, x,
n, b(k), a, a0), [0 100], [1 0 1 5 2 2]);
    %then calculates their rate of change at
those points
    u = ddt(x, n, b(k),a,a0);
    %and plots them
    scale_factor = 0.001; %change relative
scale to make vectors easier to see
    if k == 1
        quiver(x(:,4), x(:,5), u(:,4), u(:,5))
    else
      quiver(x(:,4), x(:,5),
u(:,4)*scale_factor, u(:,5)*scale_factor, 0);
    end
    xlabel('cI'); ylabel('lacI');
title(titles(k))
end
suptitle ('Evolution of phase portrait as a
function of \beta')
```



Arrow direction is direction of change and length is magnitude/size of derivative at that point

```
function dx=ddt(x, n, b , a, a0)
%x(1) to x(3) = m1 to m3 (lacI, tetR, cI), x(4) to x(6)
= p1 to p3 (cI,
%lacI, tetR)

dx(:,1) = -x(:,1)+a./(1+x(:,4).^n)+a0;
dx(:,2) = -x(:,2)+a./(1+x(:,5).^n)+a0;
dx(:,3) = -x(:,3)+a./(1+x(:,6).^n)+a0;
dx(:,4) =-b*(x(:,4)-x(:,3));
dx(:,5) = -b*(x(:,5) - x(:,1));
dx(:,6) = -b*(x(:,6)-x(:,2));
```

Vector fields

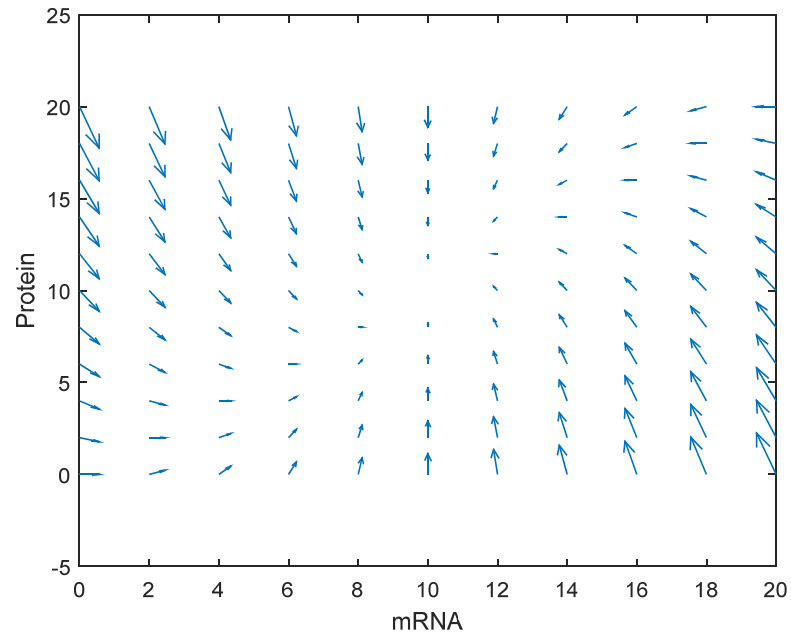
For simpler (low dimensional/non-interacting) systems, may be helpful to look at how these ***derivatives*** vary in phase space

Assume simple constitutive expression of a gene with x = mRNA and y = protein

```
[x y] = meshgrid(0:2:20,
0:2:20);

%constitutive gene expression
dy = 2.*x-2.*y;
dx = 10-x;

quiver(x,y,dx, dy);
```



What are the arrows pointing to? Why do they disappear in the center?