EEB 324: Theoretical Ecology (TA: Marissa Baskett)
Lab 1 notes: Basic Matlab functions, plotting, and numerical integration

To be able to access your files from any cluster computer, you need to save them on your H: drive. To keep organized, it might be a good idea to create a folder on your H: drive for the class, and one within that for each problem set. Each time you open Matlab, make sure you change the directory to the correct folder (by clicking on the ... button with in the upper right hand side of the Matlab window).

When you open Matlab, you should see a prompt that looks like this:

```
>>
```

You can do basic math at this prompt, for example

```
>> 3+4
```

gives you

```
ans =
7
```

and

```
>> exp(1)
```

gives you

```
ans =
2.7183
```

The `help` function gives you a quick idea of what functions do, for example

```
>> help exp
```

gives

```
EXP Exponential.
EXP(X) is the exponential of the elements of X, e to the X.
For complex Z=X+i*Y, EXP(Z) = EXP(X)*(COS(Y)+i*SIN(Y)).
See also LOG, LOG10, EXPM, EXPINT.
```

For more detailed help, use the Help menu.

You can also assign values to variables, such as

```
>> x = 3

x =
3
```

Now every time you use `x`, Matlab will substitute the value 3, for example

```
>> y = x+1

y =
4
```

If you don't want Matlab to report back at you every time, you can use the semicolon

```
>> y = x+1;
```

Vectors are a list of values. The colon gives a sequential vector, for example

```
>> v = 1:5

v =
1 2 3 4 5
```

If you want increments in values other than one, put the increment step in the middle:

```
>> w = 1:2:9

w =
1 3 5 7 9
```

You can see or change a particular entry with parentheses

```
>> w(3)

ans =
5
```

You can perform basic math on vectors, for example

```
>> log(3*w)

ans =
1.0986 2.1972 2.7081 3.0445 3.2958
```

To do element-by-element math with two vectors, use a period. For example,

```
>> v.*w

ans =
1 6 15 28 45
```

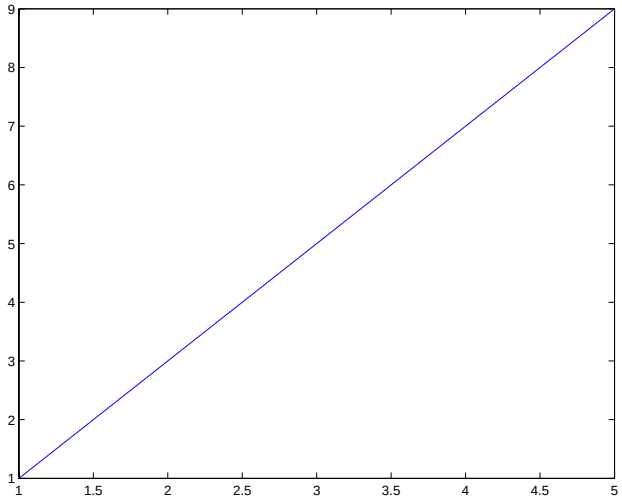
You can plot two vectors of equal length. First pull up a new figure with

```
>> figure
```

Then use the `plot` command

```
>> plot(v,w)
```

and you should get



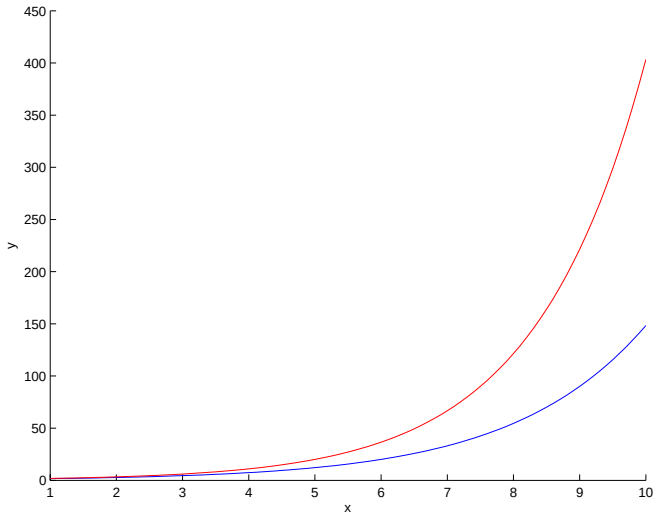

A script lets you run commands you've saved in a file. To create a script, go to the File menu, and select New > M-file. Now write a few commands, such as

```
% test.m is an example script that plots two functions
% These are comments - everything after the % is ignored by Matlab

% Create some vectors
x = 1:0.1:10;
y1 = exp(0.5*x);
y2 = exp(0.6*x);

% Plot the vectors
figure % Pull up a new figure
hold on % Allow multiple plots on the same figure
plot(x,y1)
plot(x,y2,'r') % Use a red line for the second plot
xlabel('x'), ylabel('y') % Label the axes
print -dpdf test.pdf % Save to a file
```

Save this file as test.m in the same folder you're currently working in. Now you can go back to the Matlab prompt and type

```
>> test
```

and it should do all of the commands in your script, giving you your figure




and saving that figure to test.pdf in your folder.

You can also create functions with M-files. To create a function that gives the population change over time for logistic growth, open a new M-file and type

```
function dNdt = logGrowth(t,N)
% logGrowth gives the growth rate dNdt of a population of size N at time t
% based on the logistic model
% Usage:  dNdt = logGrowth(t,N)
r = 0.5;
K = 100;
dNdt = r*N.*(1-N/K);
```

and save this as logGrowth.m (notice that's the same name as the function). Technically, we don't need to know the time `t` for this function, but it will be clear later why that's there. The first lines of comments will come up when you type help:

```
>> help logGrowth

logGrowth gives the growth rate dNdt of a population of size N at time t
based on the logistic model
Usage:  dNdt = logGrowth(t,N)
```

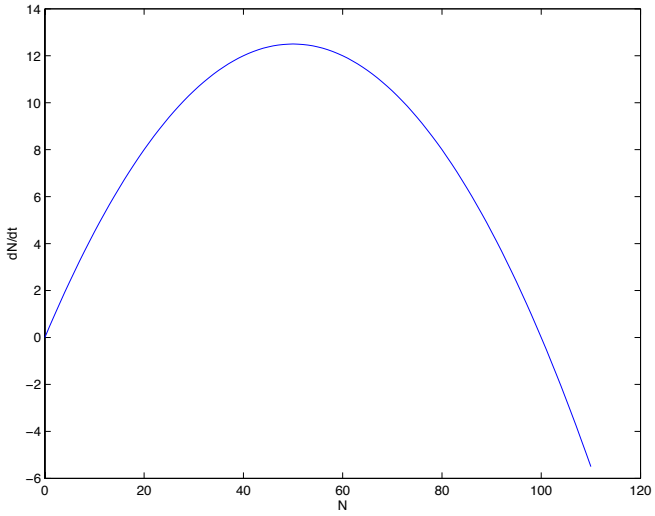
Now you can use your function like one of Matlab's, for example,

```
>> logGrowth(1,100)

ans =
0
```

Having the `.*` in the function lets you use vectors in the function, for example

```
>> vN = 0:110;
>> figure
>> plot(vN, logGrowth(0,vN))
>> xlabel('N'),ylabel('dN/dt')
```

gives a plot



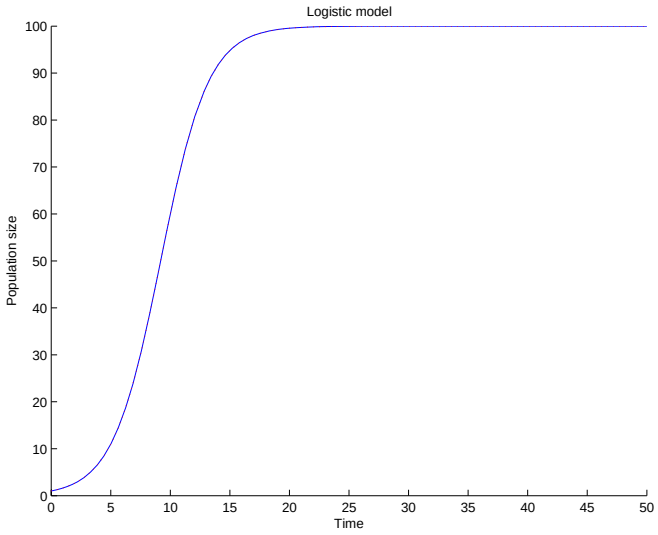

Matlab can numerically integrate a differential equation like the logistic growth function with `ode45`. First you need to create a function for your differential equation, which we just did. To use `ode45`, that function must have the form `dxdt = odefun(t,x)`, which is why we have the `t` even though it isn't used. Here's a script that numerically integrates our logGrowth function:

```
% intLog.m integrates the logistic growth model

% Numerical solution
t0 = 0; % Initial time
tf = 50; % Final time
N0 = 1; % Initial population size
[T, vNint] = ode45(@logGrowth, [t0 tf], N0); % Numerically integrate

% Actual solution
r = 0.5;
K = 100;
vNact = (N0*exp(r*T))./(1+N0*(exp(r*T)-1)/K); % Actual solution

% Plot results
figure, hold on
plot(T,vNint) % Plot numerically integrated solution
plot(T,vNact,'r:')  % Plot actual solution in red dotted line
xlabel('Time'), ylabel('Population size'), title('Logistic model')
print -dpdf logPlot.pdf % Save
```

Save this script as intLog.m and run it with

```
>> intLog
```

which gives the plot


Logistic model
Population size
Time
0
10
20
30
40
50
60
70
80
90
100
0
5
10
15
20
25
30
35
40
45
50