The BioMath Package

1 Introduction

The BioMath Package defines a number of functions that are useful throughout MTH 114. To get started, you will need access to Wolfram's computer algebra system *Mathematica*. Most computers on campus should have a version of the software already installed. The icon for *Mathematica* is typically the image below.

Figure 1: Standard *Mathematica* Icon

Wilkes students are also allowed to download a copy of the software for free. Visit the link below for instructions.

Mathematica at Wilkes

A nice general purpose introduction to using *Mathematica* can be found on Wolfram's website.

2 Running the BioMath Package

In order to use the BioMath package, you must download the BioMath Package file, `BioMath.m`. You can find this file here. If you follow this link and

click "*Mathematica:* BioMath Package," this file should be downloaded automatically. **It is important to remember the directory where you save `BioMath.m`!** You will need to tell *Mathematica* where the package is so that it can be loaded successfully.

To make the BioMath Package easier to use, the webpage in the link above also contains a temple *Mathematica* file[1], "`BioMathTemplate.nb`." You can download the template by clicking on the link labeled "*Mathematica:* BioMath Template Notebook." If you open this notebook, you should see the following.

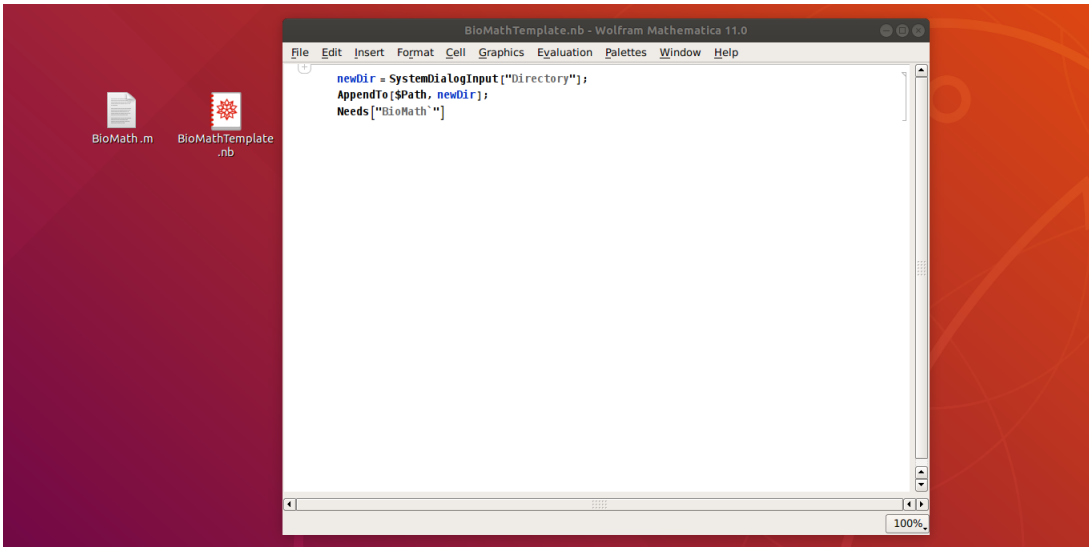

Figure 2: The BioMath Template Notebook

The first three lines of code will allow you to select the directory where you stored `BioMath.m` and then automatically load the package. Once you evaluate this cell, a system dialog box should appear prompting you to select a directory.

[1] *Mathematica* files are typically stored as `file_name.nb` where "`.nb`" stands for "notebook."

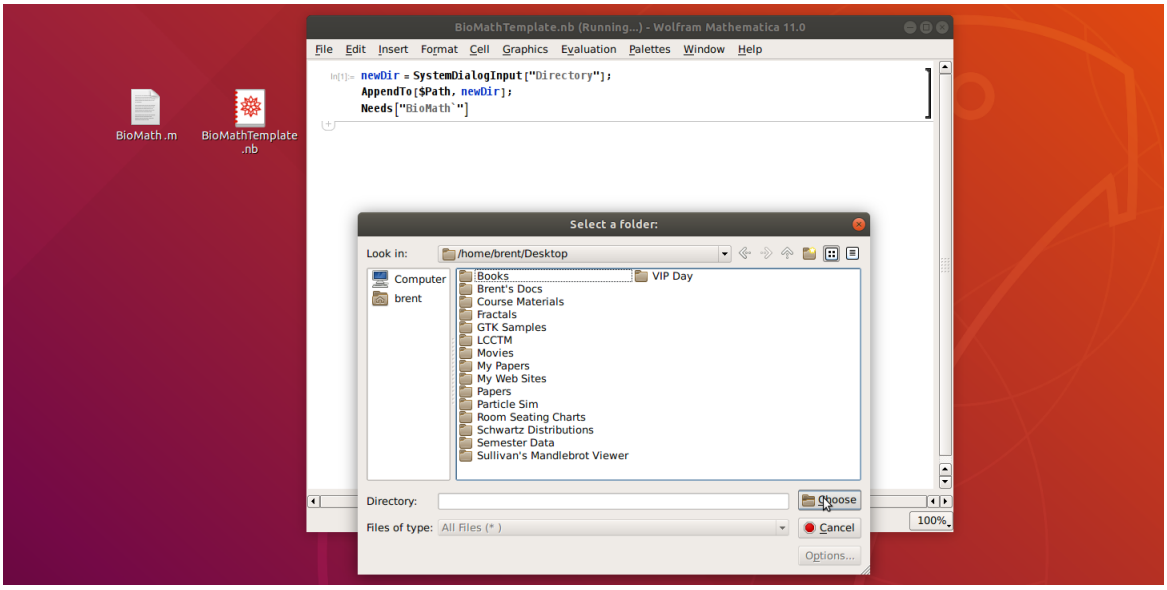

Figure 3: Directory Selection

Note that file names may not appear since you are being asked for a directory! If you have selected the correct directory, then the BioMath Package should load (there will not be any output in this case). Otherwise, you will receive an error message, and you should retry the entire process.

3 Functions Defined by the BioMath Package

3.1 CobwebPlotStatic and CobwebPlotDynamic

Both CobwebPlotStatic and CobwebPlotDynamic produce cobweb plots for a discrete dynamical system with updating function of the form

$$x_{n+1} = f(x_n).$$

CobwebPlotDynamic produces an animated version building the cobweb plot one step at a time. CobwebPlotStatic simply produces the final version of the requested cobweb plot with no animation. Since the two functions have the same set of arguments, we will look at CobwePlotStatic in more detail.

The syntax for CobwebPlotStatic is

$$\text{CobwebPlotStatic}[f, \, x\text{ran}, \, x_0, \, \text{num}]$$

where the arguments are as follows.

1) f is the updating function

2) $x$ran is the range to be plotted

3) x_0 is the specific initial value for the dynamical system

4) num is the total number of steps for the dynamical system

Most of these arguments are self–explanatory, but the $x$ran variable requires some elaboration. The default syntax for specifying a range in *Mathematica* is a list of the form

$$\{\text{variable name, variable minimum, variable maximum}\}.$$

The variable name for $x$ran must match the variable used to define the updating function f.

As a simple example suppose we want to generate a cobweb plot showing 10 steps of the discrete dynamical system

$$\begin{aligned} x_{n+1} &= 2x_n(1 - x_n) \\ x_0 &= 0.2 \end{aligned}$$

over the range $0 \leq x \leq 1$. Since the updating function is $f(x) = 2x(1 - x)$, we can use the command

$$\text{CobwebPlotStatic}[2x(1 - x), \{x, 0, 1\}, 0.2, 10].$$

This gives the complete cobweb plot shown below.

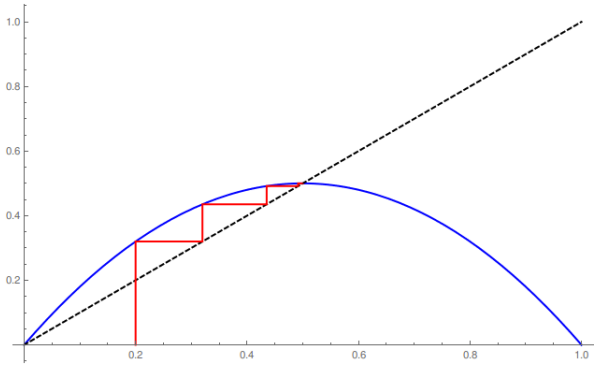

Figure 4: Output from CobwebPlotStatic

Note that the line $y = x$ is shown in dashed black while the graph of the updating function $y = f(x)$ is in solid blue. The steps in the dynamical system are shown in red.

CobwebPlotStatic has a single optional argument, Ran. This can be used to fine tune the viewing window for the cobweb plot, and the default setting is "Full." This option will force *Mathematica* to display the entire graph of $y = x$ and $y = f(x)$. There are two other common choices for this setting.

- Ran→Automatic

  This option will instruct *Mathematica* to find an "optimal" viewing window for the cobweb plot. Optimal in this case is determined by a general plotting algorithm programmed into *Mathematica*.

- Ran→{{horizontal min, horizontal max},{vertical min, vertical max}}

  This option instructs *Mathematica* to produce a viewing window with specified limits.

Typically, Ran can be left on its default setting, but having the freedom to specify the viewing window can occasionally be useful.

As an example, if we give *Mathematica* the instruction

CobwebPlotStatic[$0.2x^3 - x + 1$, $\{x, 0, 3\}$, 2, 10, Ran → Automatic]

we get the following cobweb plot.

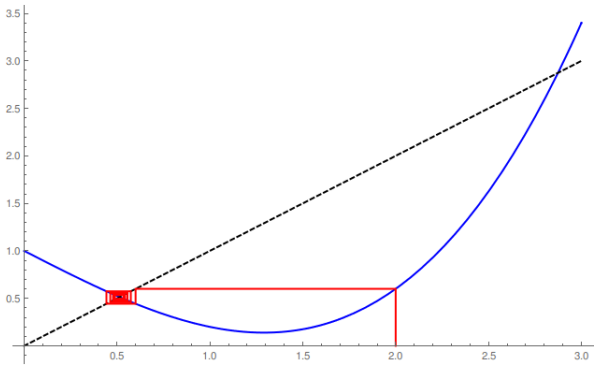

Figure 5: Output from CobwebPlotStatic, Ran → Automatic

In this case, the resulting graph is identical to the one obtained using the default setting for Ran. Since the interesting part of the graph is near the point $(0.5, 0.5)$, we can use the Ran option to zoom in on this section:

CobwebPlotStatic[$0.2x^3 - x + 1$, $\{x, 0, 3\}$, 2, 10,
Ran $\rightarrow$$\{\{0.4,0.6\},\{0.4,0.6\}\}$].

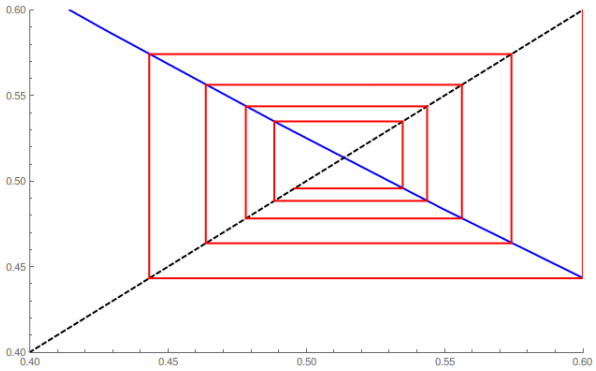

Figure 6: Output from CobwebPlotStatic, Detailed Ran Specification

In this case, we could obtain the nearly the same graph by instead changing $x$ran to $\{x, 0.4, 0.6\}$ and leaving Ran on its default setting.

CobwebPlotDynamic has exactly the same function arguments as CobwebPlotStatic. The figure below shows a typical output for this function.

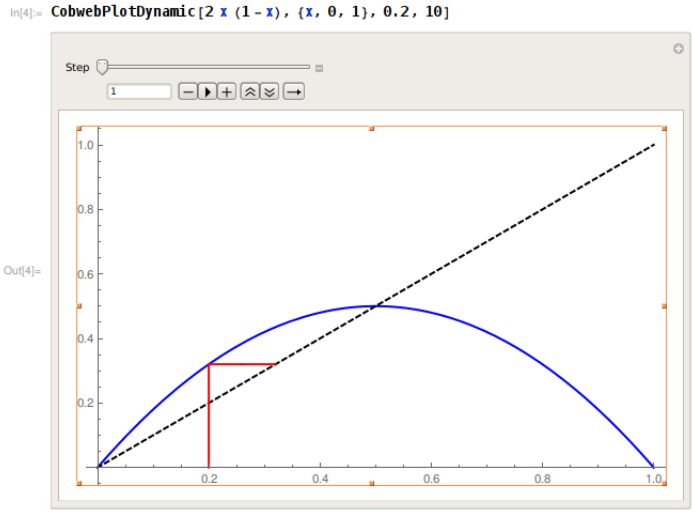

Figure 7: Output from CobwebPlotDynamic

You can either use the slider or the – and + buttons to build up the cobweb plot step–by–step. The ▶ button will animate the cobweb plot. The other

buttons can be used to change the animation rate and determine how the animation loops through the frames.

3.2 VectorField

The function VectorField will produce a vector plot for a 2×2 system of differential equations of the form

$$\frac{dx}{dt} = F(x, y)$$
$$\frac{dy}{dt} = G(x, y).$$

VectorField will also draw the nullclines ($F(x, y) = 0$ will appear in dashed blue and $G(x, y) = 0$ will appear in dashed red) and mark the (real) equilibria. This command is called with the syntax

VectorField[F, G, $x$ran, $y$ran]

where

1) F is the right–hand–side of the first differential equation

2) G is the right–hand–side of the second differential equation

3) $x$ran is the range for the first dependent variable

4) $y$ran is the range for the second dependent variable

Both $x$ran and $y$ran must be in standard *Mathematica* form:

{variable name, minimum value, maximum value}.

The variable names must match the variables used to define F and G.

As an example, to generate a vector plot for the system

$$\frac{dx}{dt} = x(y - x)$$
$$\frac{dy}{dt} = x - y^2$$

over the range $-2 \leq x \leq 2$ and $-2 \leq y \leq 2$, we input the command

VectorField[$x(y-x), x-y^2$, {x, -2, 2}, {y, -2, 2}].

This produces the following plot.

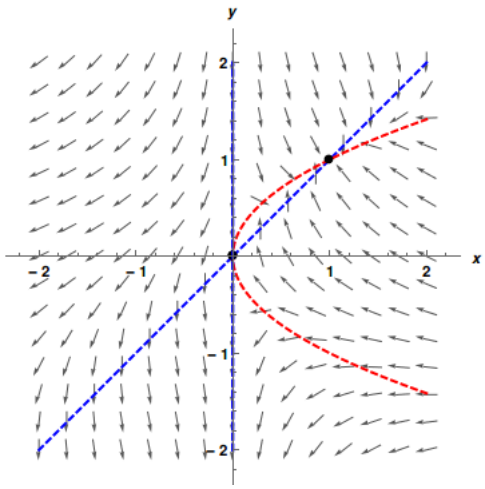

Figure 8: Output from VectorField

VectorField has a large number of optional arguments. The simplest ones to use are the following.

- Vpts→number

  This option instructs *Mathematica* to divide each axis into the given number of intervals and draw a direction vector for each resulting region (i.e. there will be (number)^2 vectors in total). The default setting is Automatic.

- Plotpts→number

  This option is used to refine the rendering of the nullclines. The default setting is 100. If the nullclines seem to be drawn incorrectly, try increasing this value.

- Ptsz→number

  This option controls the relative point size of the disks used to mark the equilibria. The default setting is 0.02.

The remaining options are a bit more complicated, and we will examine them one–by–one.

The **Tcks** option allows you to customize the tick marks shown on the axes. The general form for this command is

$$\text{Tcks}\rightarrow\{\{\text{ticks for horizontal axis}\},\{\text{ticks for vertical axis}\}\}.$$

As an example, if we need a vector plot over the region $0 \leq x, y \leq 6$ but we want *Mathematica* to display the odd integers on the x–axis and the even integers on the y–axis, we would give VectorField the command

$$\text{Tcks}\rightarrow\{\{1,\ 3,\ 5\},\{2,\ 4,\ 6\}\}.$$

Even though 0 is even, *Mathematica* may not display a tick mark for that y–value since $(0, 0)$ is the point where the axes meet in this plot. There is far more functionality in this option than we can discuss in this brief overview. For more details, see the documentation for *Mathematica*'s Ticks option. The syntax is precisely the same for the option Tcks in VectorField.

The **Init** and **Thck** options are used to display sample solution curves to the system of differential equations

$$\begin{aligned}\frac{dx}{dt} &= F(x, y)\\ \frac{dy}{dt} &= G(x, y).\end{aligned}$$

The option Thck→number changes the relative thickness of the solution curves so that they are easier to see on the vector plot. The default thickness setting is 0.005.

The Init option is used to specify the initial conditions for which you want solution curves. The generic form for Init is a *list* of initial conditions:

$$\text{Init}\rightarrow\{\{\text{first initial data}\},\ \{\text{second initial data}\},\ \{\text{third initial data}\},\ \cdots\}.$$

Each individual initial data specification must itself be a list of 4 numbers in the order:

$$\{\text{initial data}\} = \{t_i, x_0, y_0, t_f\}$$

where

- t_i is the initial time (often 0),

- x_0 is the initial value for the first dependent variable,
- y_0 is the initial value for the second dependent variable,
- t_f is the final time (so larger than t_i) for the solution curve to be drawn (necessary since the solution is found numerically).

Any finite number of initial conditions can be specified, however VectorField cycles through five colors when drawing the solutions:

1) Green
2) Magenta
3) Cyan
4) Orange
5) Yellow

After the fifth solution curve, the colors will begin to repeat. VectorField also marks the initial point for each solution curve. If the thickness of the curve is chosen too large, the initial point may be obscured.

As an example, the command

$$\begin{aligned}&\text{VectorField}[x(y-x),\, 2x-y^2,\, \{x,-3,3\},\, \{y,-3,3\},\\ &\qquad \text{Init} \rightarrow \{\{0,-1,2,0.5\},\, \{0,1,1,3\},\, \{0,3,-0.5,1\}\}]\end{aligned}$$

produces a vector plot for the system of differential equations

$$\begin{aligned}\frac{dx}{dt} &= x(y-x)\\ \frac{dy}{dt} &= 2x-y^2\end{aligned}$$

with solution curves having initial data

- $x(0) = -1, y(0) = 2$, and $0 \le t \le 0.5$,
- $x(0) = 1, y(0) = 1$, and $0 \le t \le 3$,
- $x(0) = 3, y(0) = -0.5$, and $0 \le t \le 1$.

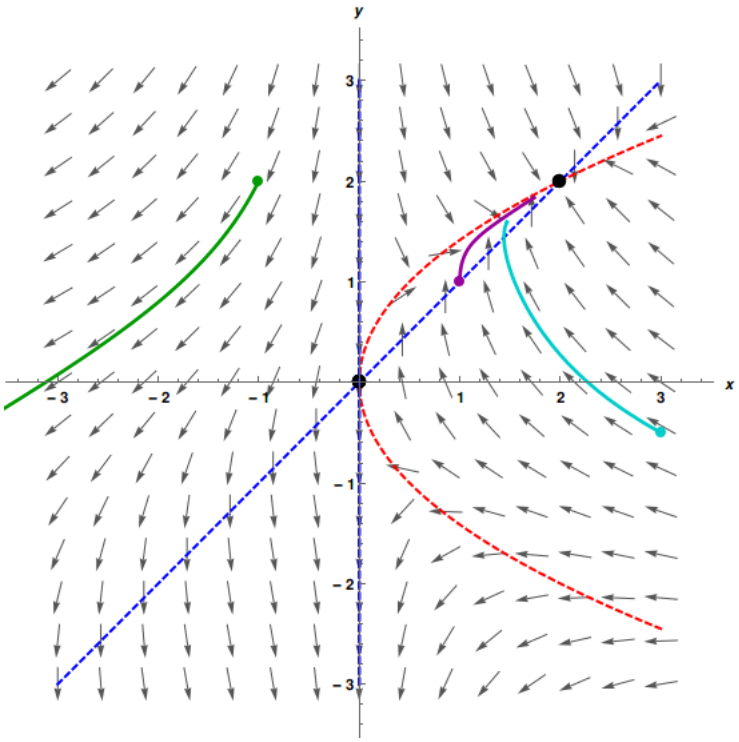

Figure 9: Output from VectorField with Solution Curves

A common error associated with plotting solution curves is that the solution limits to infinity in a finite amount of time. Consider the plot below.

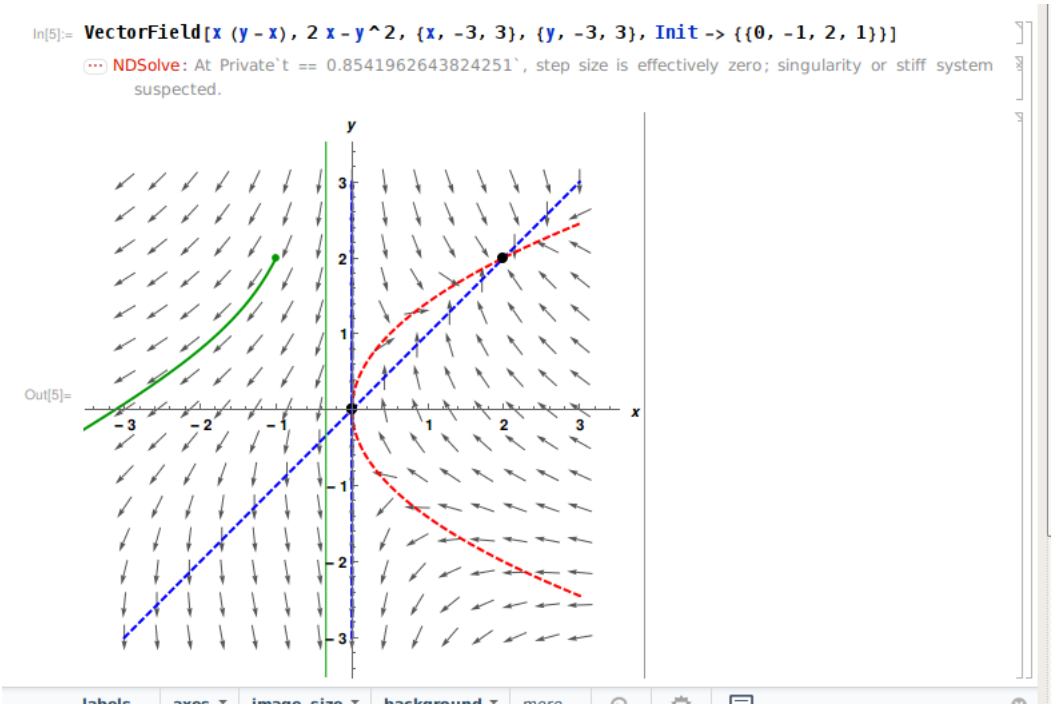

Figure 10: Output from VectorField with Error

What this error is indicating is that the solution curve with initial data $x(0) = -1$ and $y(0) = 2$ seems to stream off to infinity near time $t \approx 0.854$. Notice that *Mathematica* still plots the solution curve, but the time interval $0 \leq t \leq 1$ has been reduced due to the singularity. Also note the spurious vertical line appearing in the plot. If there are multiple initial conditions, it may not be obvious from the error message exactly which initial condition led to the singularity!