BioMath - Nonlinear dynamics 1- August 28, 2013

1. **Linearizing around fixed points.** Let's take the followin equations:

$$\dot{x} = x^2 + y^2 + x + y \tag{1}$$
$$\dot{y} = -3x \tag{2}$$

• As we did in class, find the fixed points, and use linearization of the dynamics around them to determine whether each is stable or unstable, whether it has an oscillatory component, and if it is unstable, whether it is a saddle point or not.

• Use Matlab's `quiver.m` to visualize the dynamics. Choose the horizontal and vertical limits of your plot, as well as the density of arrows, wisely (i.e., so that all the things you think are relevant are easily visible). Does the plot confirm your findings in the previous bullet point?

• Add lines to your quiver plot in the directions corresponding to the eigenvectors for the fixed points(s) that have no osicllatory component. (The lines should originate at the fixed point.) Can you do the same for the fixed point(s) with an oscillatory component?

• Write a program that, on top of the quiver plot with the eigenvector lines, drops points onto the plot at random (that is, each new point should start at a random position (x, y) and then animates their motion over the flow. Some time Tafter appearing, the point should disappear. Have parameters that determine how many new random points are dropped in per time period T. Use this to animate the flow.

2. **Linearization 2**. Now consider the following equations (which come from a model of glycolysis oscillations in yeast):

$$\dot{x} = -x + ay + x^2 y \tag{3}$$
$$\dot{y} = b - ay - x^2 y \tag{4}$$

• Find the Jacobian matrix for these dynamics

• What is the fixed point for the dynamics? When $a = 0.08$ and $b = 0.6$, is it stable or unstable? Does it have oscillations?

• Using Matlab, animate the trajectory of a particle following these dynamics, that starts very near the fixed point. Have the particle be represented by am easy visible red dot; as it moves, have it leave a blue trail behind, so you can see the history of its trajectory. (For extra cuteness, you could have this all happen on top of a quiver plot that represents the flow). Limits of $[0, 3]$ on the x and y axes will probably be good. Does the particle's trajectory go to infinity? Does it converge to a fixed point? Or does it do something else?

3. **Mutual inhibition** In this question you will examine a 2D circuit similar to the one we did in class. The equations are

$$\frac{dx}{dt} = -x - w\tanh(y) \tag{5}$$

$$\frac{dy}{dt} = -y - w\tanh(x) \tag{6}$$

with $w > 0$.

(a) What is the nullcline for the system when $\dot{x} = 0$? How about $\dot{y} = 0$? Using MATLAB, draw the two nullclines when $w = 0.5$. Now, do it for when $w = 2$. How many fixed points did you find in each case? Can you work out the stability of each point? What do all of the fixed points represent in terms of the steady-state of the network?

(b) There is a fixed point at $x^* = 0, y^* = 0$. Let us find the conditions on w for when this fixed point is linearly stable. We will use a Taylor series expansion again, except now for two variables. The Taylor series expansion of a 2D function about the point x_0, y_0 is given by

$$\begin{aligned} f(x,y) = f(x_0,y_0) + (x-x_0)\left.\frac{\partial f(x,y)}{\partial x}\right|_{x_0,y_0} + (y-y_0)\left.\frac{\partial f(x,y)}{\partial y}\right|_{x_0,y_0} \\ + \frac{(x-x_0)^2}{2!}\left.\frac{\partial^2 f(x,y)}{\partial x^2}\right|_{x_0,y_0} + \frac{(y-y_0)^2}{2!}\left.\frac{\partial^2 f(x,y)}{\partial y^2}\right|_{x_0,y_0} \\ + \frac{(x-x_0)(y-y_0)}{2!}\left.\frac{\partial^2 f(x,y)}{\partial x \partial y}\right|_{x_0,y_0} + \ldots \end{aligned}$$

Linearize equations (5) and (6) about $x^* = 0, y^* = 0$, by performing the Taylor series expansion up to first order derivatives.

(c) Write your equations in matrix form. Now, find the eigenvalues and eigenvectors of your matrix as a function of w. For what values of w is the fixed point at $x^* = 0, y^* = 0$ linearly stable? What type of fixed point does $x^* = 0, y^* = 0$ become when it does not meet this condition?

(d) Use the Euler method to simulate the dynamics of this system. The update equations are given by

$$x_{n+1} = x_n + \delta t(-x_n - w\tanh(y_n)) + \eta_x$$
$$y_{n+1} = y_n + \delta t(-y_n - w\tanh(x_n)) + \eta_y$$

where η_x and η_y are noise values independently sampled from a normal distribution with zero mean and standard deviation $\sigma\sqrt{\delta t}$ (we have set $\tau = 1$). Use $\delta t = 0.01$, $x_0 = 0$ and $y_0 = 0$, $w = 0.5$, $\sigma = 0.1$, and run the simulation up to $t = 10$. Repeat this simulation for ten different trials (don't repeat the noise in each trial but start from the same initial condition). Plot the dynamics of each variable for each trial.

(e) Try the simulation with $w = 2$. First, do one simulation with no noise, i.e. $\sigma = 0$. What happens? Now set $\sigma = 0.1$. What happens after you put in the noise? Again, run the simulation for 10 trials. Do you get the same result on every trial?

(f) Animate your simulations with the phase portrait and nullclines. We will go through this step-by-step.

   i. To plot the phase portrait, we are going to use the MATLAB function quiver. To draw a field of arrows use quiver(x,y,u,v). Where x,y are vectors representing the x- and y-positions of the tails of each arrow, and u,v are the x- and y-components of each arrow respectively. For example, try quiver(0,0,1,0) and quiver([0 0],[0 0],[1 0],[0 1]). quiver will scale the size of the arrows, so it will not always respect the absolute numbers in u and v. To get the vectors x and y, we need to make a grid. We can use the MATLAB function meshgrid. Try [xgrid,ygrid]=meshgrid(-3:0.25:3,-3:0.25:3), then see what it does by typing plot(xgrid,ygrid,'.k'). Evaluate the x- and y- components of each arrow by calculating $\dot{x}$ and $\dot{y}$, from equations (5) and (6), at every point in xgrid and ygrid. Call quiver, now you have the phase portrait.

   ii. On the same figure, we are going to plot the nullclines—lines for which the derivatives in (5) and (6) equal 0. Make an independent vector z=linspace(-3,3,100). To get the nullcline for the y-direction, set $\dot{y} = 0$ to obtain the equation $y = -w\tanh x$. You can draw this line on the same graph with your arrows by typing line(z,-w*tanh(z),'color','k'). For the x-direction, set $\dot{x} = 0$ to obtain the equation $x = -w\tanh y$. Again, draw this line on the same graph using line(-w*tanh(z),z,'color','r') (notice the switch in arguments).

   iii. Draw a single dot representing the initial state of the system, and return its handle. h=line(x0,y0,'color','g','marker','.','markersize',28), where x0,y0 represent your initial x- and y-values. Now, iterate over all of your time steps, and update the position of your green dot using the set function. That is, at every time point call set(h,'xdata',x(t),'ydata',y(t)); drawnow;. Do this for all 10 trials you simulated.

4. **Mutual inhibition with autapses** Now, let's generalize our network to allow for all different kinds of connections. We are going to allow the neurons to have reciprocal connections and autapses, we are not going to make all synaptic weights the same value, and we will allow them to be negative or positive. We can write this system as

$$\frac{dx}{dt} = -x + w_{xx}\tanh(x) + w_{xy}\tanh(y) \tag{7}$$
$$\frac{dy}{dt} = -y + w_{yy}\tanh(y) + w_{yx}\tanh(x) \tag{8}$$

where w_{xy} indicates the synaptic strength that x receives from y, w_{yx} is the opposite, and w_{xx} and w_{yy} represent autapses. Simulate the equations (7) and (8) using the Euler method as detailed in the previous question. Do it for the following set of synaptic weights

(a) $w_{xx} = w_{yy} = -w_{xy} = w_{yx} = 2$.

(b) $w_{xx} = w_{yy} = w_{xy} = -w_{yx} = 2$.

(c) $w_{xx} = w_{yy} = 2, w_{xy} = -1, w_{yx} = 2$.

(d) $w_{xx} = w_{yy} = 2, w_{xy} = -0.5, w_{yx} = 2$.

(e) $w_{xx} = w_{yy} = 2, w_{xy} = -w_{yx} = -0.1$.

For each set of weights, animate your simulations on top of quiver plots, but do not worry about plotting nullclines. (Hint, if you have not done it yet, "functionalizing" your code will help you perform the different parts with minimal extra work.)