2.1 Lyapunov analysis of networks

It was shown in class that the following differential equation is a simple model of a neural network’s dynamics,

$$\tau \dot{u}_i = -u_i + \sum_j T_{ij} V_j + I_i$$

where $V_k = g(u_k)$ for some activation function $g(\cdot)$. The number of neurons is $N$, so $i \in (1, \ldots, N)$ and the sums all run from 1 to $N$. The variable $\tau$ is the characteristic time constant of the system. The form of this equation should be familiar to you from lecture.

It was then claimed that the following Lyapunov function describes the energy of such a system, which always decreases with time if certain conditions are met:

$$L = \frac{1}{2} \sum_{i,j} T_{ij} V_i V_j + \sum_i G(V_i) - \sum_i V_i I_i$$

where $G(V_k) = \int_0^{V_k} g^{-1}(z) dz$ and $g^{-1}(V_i) = u_i$ is the inverse gain function.

What we would like to show in this problem is that by choosing a symmetric $T_{ij}$, we can guarantee that the energy always decreases with time.

- First find the gradient of the energy ($\frac{\partial L}{\partial V_k}$) with respect to every output $V_k$. At this point, insist that $T_{ij}$ is symmetric. What do you notice?

- Next find the derivative of $L$ with respect to time ($\dot{L} = \sum_k \frac{\partial L}{\partial V_k} \dot{V}_k$) and show it always decreases with time, *i.e.*, $\dot{L} \leq 0$ and $\dot{L} = 0 \rightarrow \dot{V} = 0$ for all $i$.

- Finally, show that $L$ is bounded from below, *i.e.*, $L \geq C$, in which $C$ is a constant independent of the state variables $\{u_i\}$ and $\{V_i\}$. Assume that the activation function $g(\cdot)$ is bounded and monotonic and that $g(0) = 0$.

Because the last two conditions hold, the function $L$ is a Lyapunov function of the system (Wiggins 1988). The time evolution of the state in the space of $\{V_i(t)\}$ leads to attractors which are at local minima of $L$.

2.2 Simple Neural Network

- Write a general routine that will simulate this kind of neural network model. You can use the ideas, and maybe even some of the actual code, behind the numerical integration method before. Simulate the same differential equation given above, taking $I_k = 0 \forall k$, and using the activation function $g(x) = \tanh(\beta x)$. The constant $\beta > 0$ is the gain of the neuron amplifiers, and corresponds to the slope of the $g(x)$ curve at the origin. The constant $\tau$ just sets an arbitrary time-scale and doesn’t really affect the behaviour except in its ratio to $\beta$ and the simulation timestep, so for simplicity in all of your code assume $\tau = 1$.

The Euler method that you have seen before will suffice to simulate this model, but feel free to use a more sophisticated integration routine if you know of one. We have provided a skeleton function `simhop.m` which you can use. You need to fill in about 3 lines of code inside the loop.
You will remember from before that excessive step sizes can lead to unstable behavior, so be attentive to that aspect of your program. Note that since $\tau$ sets the time constant of the system, you want to set your time-step size to be smaller than $\tau$: this way simulation time-steps look very tiny from a $\tau$’s-eye viewpoint.

Because you will need to use this program for the next problem, be sure to write it so that it works with any sized square connection matrix $T$ (in other words any $N$) and any general values for the initial conditions in which the network starts.

- We investigated in lecture the two neuron case with phase-plane methods; now you will simulate that case with your program. Use the weights $T_{11} = T_{22} = 0$ and $T_{12} = T_{21} = -1$. As your simulation runs, your program will generate pairs of numbers representing the networks states $(u_1, u_2)$ and $(V_1, V_2)$ at each time step. Plot these pairs of numbers as a trajectory over time in either the $u_1, u_2$ or $V_1, V_2$ phase space for 40 different initial conditions. Use initial conditions evenly distributed around the perimeter of the box defined by $u_i = \pm 1.5$. You should hand in four small graphical plots of the network behaviour. First make one plot of the $u$ phase space showing the trajectories for all 40 initial conditions and a second plot of the same trajectories in the $V$ phase space. Next do the same thing, but for a different setting of the gain $\beta$ that shows a qualitatively different behaviour. LABEL YOUR PLOTS with the values of the gain $\beta$ and the timestep $\eta$ that you used to generate them.

Confused? Here is an example of what two of your four plots should look like (here we are showing the connection matrix $T_{11} = T_{22} = 0; T_{12} = 1; T_{21} = -1$ and $\beta = 1.2$ so don’t think your plots should look exactly like this). The MATLAB command `hold` is important to know about in order to get all 40 traces on the same plot. We have provided a function `plothop2.m` which will plot a single trace so you can get the idea.

- What is the smallest gain $\beta$ for which the system has fixed points away from the origin $V = u = (0, 0)$? Answer this both analytically, by finding the $\beta$ for which the origin becomes unstable given our choice of $g$ and $T$, and empirically, through simulations.