1.1 Simple Neural Network: Symmetric/Asymmetric

- The routine will be the same as in homework No.3, Please refer to the Appendix.
- Now here we go to the plotting part, in these plottings, $\beta = 2.0, 0.5$, and $\eta = 0.1$, and $\text{iters}=100$.

- From the above plottings, we can see that there is no qualitative difference in the stability of the anti-symmetric network dynamics for different value of $\beta$. Specifically, the number/position of stable point doesn’t change for this network of two neurons with anti-symmetric connection.
- Now $T_{11} = T_{22} = 0, T_{12} = -0.9, T_{21} = -1.1$, $\beta = 2.0, 0.5$:
• This network doesn’t show qualitatively different behavior than the one with $\alpha = 0$.
• If we want to gain some understanding of asymmetric biological network by studying a simple symmetric network, the condition should be $\alpha << 1$.

1.2 Network Properties and Lyapunov Analysis

• In this problem, we would like to get an informal sense of the difference between networks with symmetric connections and those without. We have seen in lecture and in HKP that the "energy function" given by

$$L = -\frac{1}{2} \sum_{i,j} T_{ij} V_i V_j + \sum_i \int_0^{V_i} g^{-1}(V) dV$$

(1)

is a Lyapunov function for the dynamics

$$\frac{dU_i}{dt} = -u_i + \sum_j T_{ij} V_j$$

(2)

(without losing generality, we assume that the time constant $\tau_i = 1$.) Differentiating $L$ gives

$$\dot{L} = -\frac{1}{2} \sum_{i,j} T_{ij} \frac{dV_i}{dt} V_j - \frac{1}{2} \sum_{i,j} T_{ij} V_i \frac{dV_j}{dt} + \sum_i g^{-1}(V_i) \frac{dV_i}{dt}$$

(3)

By adding and subtracting the first term in this equation and inverting the indices on the second, we get (we use $\dot{\hat{V}} = \frac{dV}{dt}$ for convenience)

$$\dot{L} = -\frac{1}{2} \sum_{i,j} T_{ij} V_i \dot{V}_j - \frac{1}{2} \sum_{i,j} T_{ij} \dot{V}_i V_j + \sum_i g^{-1}(V_i) \dot{V}_i + \frac{1}{2} \sum_{i,j} T_{ij} V_i \dot{V}_j - \frac{1}{2} \sum_{i,j} T_{ij} \dot{V}_i V_j$$

(4)
\[ \dot{L} = \frac{1}{2} \sum_{i,j} (T_{ji} - T_{ij}) \dot{V}_i \dot{V}_j - \sum_i \dot{V}_i (\sum_j T_{ij} \dot{V}_j - u_i) \]

(5)

\[ \dot{L} = K_{\text{sym}} - \sum_i \dot{V}_i \dot{u}_i \]

(6)

and since \( V = g(u) \) we have

\[ \dot{L} = K_{\text{sym}} - \sum_i g'(u_i) (\dot{u}_i)^2 \]

(7)

\[ \dot{L} = K_{\text{sym}} + K_{\text{sym}}. \]

(8)

Clearly \( K_{\text{sym}} \leq 0 \), and if the network is symmetric, \( K_{\text{sym}} = 0 \). But even if the network is not symmetric, \( L \) may still qualify as a Lyapunov function. Suppose the terms \( (T_{ji} - T_{ij}) \) are small and randomly distributed about zero (a reasonable assumption if the asymmetry is interpreted as noise in the connections). But what about the contribution from \( \dot{V}_i \dot{V}_j \)? We could convince ourselves, with a fair amount of handwaving, that if the stored patterns are random and the initial conditions are also random, then the \( \dot{V}_i \dot{V}_j \) terms will be zero-mean random variables. In the large \( N \) limit, we can assume that \( K_{\text{sym}} \) is itself a zero-mean random variable, and that on average, \( L \) still qualifies as a Lyapunov function. Note, however, the similarity to the question of memory stability in the case of associative memories – we estimated the variance of a random walk by a gaussian approximation to the binomial distribution. We leave it to you as an exercise to decide if a similar technique is useful in this setting.

- We can get a little more insight by noting that \( \dot{V}_i = g'(u_i) \dot{u}_i \), and we can write

\[ K_{\text{sym}} = -\sum_i g'(u_i) \dot{u}_i [\dot{u}_i] \]

(9)

\[ K_{\text{sym}} = -\sum_i g'(u_i) \dot{u}_i [\frac{1}{2} \sum_j V_j (T_{ji} - T_{ij})] \]

(10)

- Now let’s summarize three things:
  - \( V_i \in [-1, +1] \) for all \( i \).
  - \( \dot{u} \) is small (i.e. \( \approx 0 \)) only near fixed points (by definition).
  - \( T_{ji} - T_{ij} \) are randomly distributed about zero with variance \( \sigma^2 \).

From these points we conclude that if \( \sigma^2 \) is small, the asymmetric term is significant only near the fixed points (again, compare the bracketed terms in 9 and 10). Away from the fixed points, the network should behave as it would in the perfectly symmetric case.

For each fixed point, then, there will be a neighborhood which attracts nearby trajectories (since the \( \dot{u} \) term dominates), but inside that neighborhood the behavior will be difficult to predict without quantitative knowledge about the weights – the size of that troublesome neighborhood depends on the degree of asymmetry (\( \sigma^2 \)). The growing theory of nonlinear dynamical systems, in particular that of Hopf bifurcations, is necessary if we want to take the analysis farther. A good beginning reference is by M. Hirsch and S. Smale, *Differential Equations and Dynamical Systems.*

- Now we are really just trying to compare the relative magnitudes of the bracketed terms in (9) and (10) above.

So we should compare \( \dot{u}_i \) with \( \frac{1}{2} \sum_j V_j (T_{ji} - T_{ij}) \)
Figure 1: Flow in the neighborhood of a fixed point (shaded region) as weight asymmetry increases. Trajectories flow towards the shaded area, but behavior inside that area is difficult to predict.

1.3 Memory Capacity

This is quite simple since we have roughly $\sim 3 \times 10^{10}$ neurons in the cortex the corresponding number of Hopfield network stable states is

$$\sim 3 \times 10^8$$

Yet given one sensation per 100 ms, we will have

$$10 \times 60 \times 60 \times 24 \sim 9 \times 10^5$$

sensations per day and

$$9 \times 10^5 \times 365 \times 5 \sim 1.5 \times 10^9$$

sensations for five years of our life, which is 5 times more than the theoretical capacticy for uncorrelated patterns which can be stored and recalled in a Hopfield network with $\sim 3 \times 10^{10}$ neurons. For correlated patterns the capacity will be even lower.

So we should NOT conclude that a Hopfield network could account for such huge number of memories.

Appendix

simhop.m:

function [utrace,vtrace] = simhop(T,beta,uinit,eta,itors)
%[utrace,vtrace] = simhop(T,beta,uinit,eta,itors)
%
%SIMHOP
%
%This function simulates a Hopfield network using simple Euler integration.
%
T is the connection matrix (should be symmetric)
% beta is the gain of the activation function (tanh by default)
% uinit is the initial condition vector for the u's
% eta is the timestep for the Euler integration
% iters is the number of iterations to simulate
%
% utrace is the u vector state of the network over time,
% one column per timestep in the simulation
% vtrace is the V vector state of the network over time,
% one column per timestep in the simulation

N = size(T,1);
 utrace = zeros(N, iters);
 vtrace = zeros(N, iters);

 utrace(:,1) = uinit(:);
 vtrace(:,1) = tanh(beta*utrace(:,1));

for i=2:iters
 utrace(:,i)=(1-eta)*utrace(:,i-1)+eta*T*vtrace(:,i-1);
 vtrace(:,i)=tanh(beta*utrace(:,i));
end