CN 510 - Principles and Methods of Cognitive and Neural Modeling

Assignment # 4

John Joseph

Shunting Networks

This assignment asks us to examine equations designed to model networks of n neurons as they approach equilibrium. We are given the following equations:

$$\frac{dx_i}{dt} = -Ax_i + BI_i - \sum_{j \neq i}^{n} I_j \tag{1}$$

$$\frac{dx_i}{dt} = -Ax_i + (B - x_i)I_i - x_i \sum_{j \neq i}^{n} I_j \tag{2}$$

$$\frac{dx_i}{dt} = -Ax_i + (B - x_i)\sum_{k \in D} D_{ki}I_k - (C + x_i)\sum_{l \in E} E_{li}I_l \tag{3}$$

Part one of the assignment asks us to analyze the first two equations, which represent Additive and Shunting Networks (respectively), and part two asks us to numerically simulate equation three, which models a Distant Dependent Shunting network.

Part One: Analysis

The first part of the assignment asks us to take the Additive and Shunting network equations and calculate their equilibirum solution. The assignment presents us with a network of ten neurons, each their own current; in the above equation, $n = 10$, and each component i has it's own current I_i. From this current, as well as two initial conditions A and B, we are able to solve for the equilibrium solution of each neuron.

We are asked to calculate these equilbrium solutions given input parameters A and B for two different inputs currents:

$$A = 0.1, B = 1; \tag{4}$$

$$I = [1, \frac{9}{10}, \frac{8}{10}, \frac{7}{10}, \frac{6}{10}, \frac{5}{10}, \frac{4}{10}, \frac{3}{10}, \frac{2}{10}, \frac{1}{10}] \tag{5}$$

There are two ways to do find the equilibrium solutions: the first is two determine the analytic solution for $x(t)$ and determine its value as $t \to \infty$. The second is two run a numerical simulation of our differential equation and see where it ends up after a long period of time. Both of these methods were carried out, but I will only be outlining the former (the latter was used as a sanity check, and both yielded the same results.)

To begin, let us consider a differential equation of the form

$$\frac{dx}{dt} = -ax + b \tag{6}$$

The solution will contain a homogeneous and particular component; to find the homogenous component, we must solve

$$\frac{dx}{dt} = -ax \tag{7}$$

The solution to which is of the form

$$x_{hi}(t) = Ce^{-at} \tag{8}$$

It should be noted that there is a constant term that should be included; however, this term will be taken care of in the next step, so I choose to leave it out.

The particular solution can be found by solving the equation

$$\frac{dx}{dt} = -ax + b \tag{9}$$

The particular solution must be of the form

$$x(t) = mt + d \tag{10}$$

Plugging this back into the equation, we see that

$$m = -a(mt + d) + b \tag{11}$$

We conclude that m=0, so

$$d = \frac{b}{a} \tag{12}$$

And thus, the solution to the equation proposed above is

$$x(t) = Ce^{-at} + \frac{b}{a} \tag{13}$$

As $t \to \infty$, $x \to \frac{b}{a}$. This is the equilbrium solution. Now let us examine the additive equation; by grouping all terms multiplied by x_i, which in total become analagous to a, and separately grouping the constant terms which are analagous to b, we see that

$$a = A \tag{14}$$

$$b = BI_i - \sum_{j \neq i}^{n} I_j = I_i(B+1) - \sum_{j=1}^{n} I_j \tag{15}$$

Thus, the analytic additive solution for neuron i is

$$x_i(t) = Ce^{-at} + \frac{I_i(B+1) - \sum\limits_{j=1}^{n} I_j}{A} \tag{16}$$

We see that as t grows large, the exponential term will vanish; thus the equilibrium solution is

$$x = [-35, -37, -39, -41, -43, -45, -47, -49, -51, -53] \tag{17}$$

These solutions have been numerically verified for all ten neurons.

The shunting equation appears at first to be more complex but can be easily reduced down to the same basic form. By grouping together all homogeneous and non-homogenous (constant) terms, we see that

$$a = A + I_i - \sum_{j \neq i}^{n} I_j = A + \sum_{j=1}^{n} I_j \tag{18}$$

$$b = BI_i \tag{19}$$

The equlibrium solution is there of the form

$$x_{eqi} = \frac{b}{a} = \frac{BI_i}{A + \sum\limits_{j=1}^{n} I_j} \tag{20}$$

For our parameters and initial current values, this comes out to be

$$x = [0.179, 0.161, 0.143, 0.125, 0.107, 0.089, 0.071, 0.054, 0.035, 0.018] \tag{21}$$

When we change the current values to

$$I = [10, 9, 8, 7, 6, 5, 4, 3, 2, 1] \tag{22}$$

We see that the additive and shunting equations have equilibrium solutions of

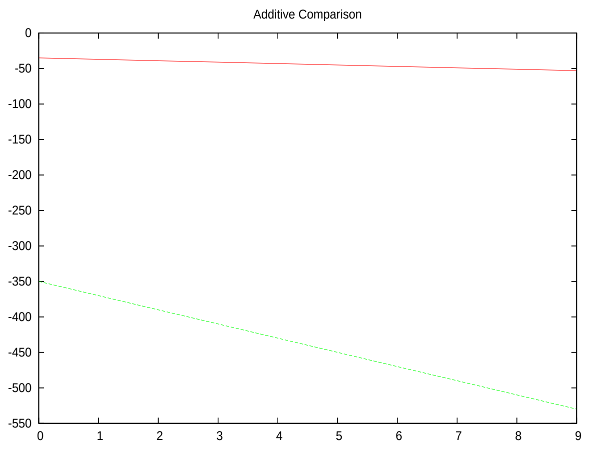
$$x = [-350, -370, -390, -410, -430, -450, -470, -490, -510, -530] \tag{23}$$

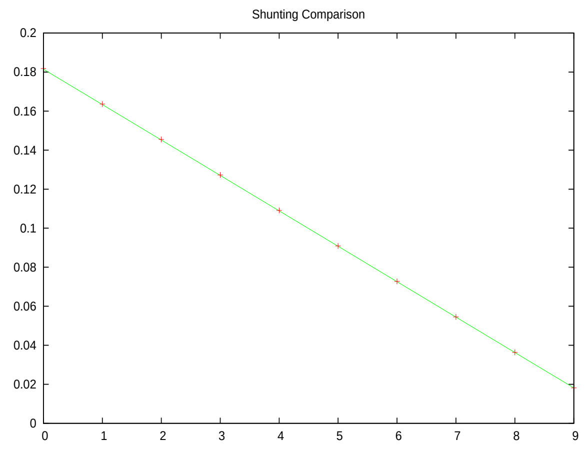
And

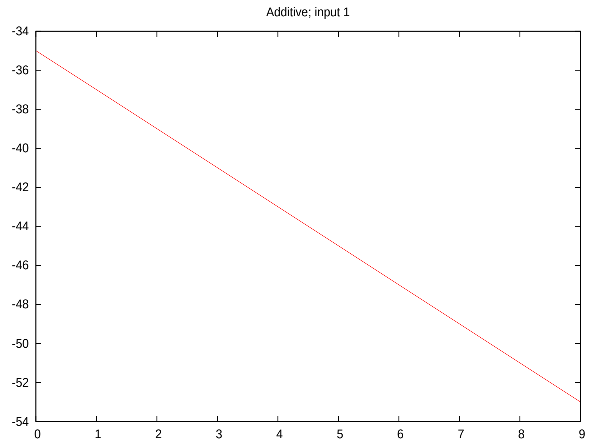
$$x = [0.182, 0.163, 0.145, 0.127, 0.109, 0.091, 0.073, 0.055, 0.036, 0.018] \tag{24}$$

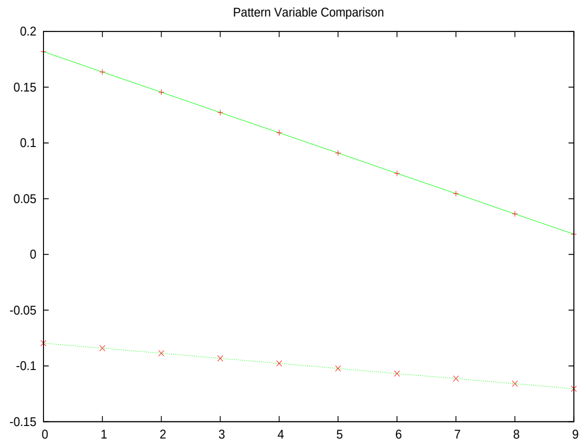
respectively. Note that while the additive network saw a change that was in direct proportion to our change in current, the shunting network remains relatively unchanged. This is due to the competitive nature of the shunting model, in which each neuron inhibits the rest; though the current values were increased tenfold, the uniformity of this increase allowed the equilibrium values to be preserved.

With regards to pattern variables, the Additive network should show the biggest change as it is never "normalized" relative to its neighbor neurons. The Shunting network should remain relatively unchanged.









The top two graphs show current comparisons for the Additive (left) and Shunting (right) neural networks; the proportionality between current and potential in the Additive network directly reflects the tenfold increase.

The bottom left shows the Additive network with the first current configuration (so it could be seen clearly), and the bottom right image shows the comparison of pattern variables for all four tests.

The two plots on the right actually show both input currents overlaid on one another; the negation of the overwhelming difference seen previously in the Additive network shows the impact of the pattern variable, though I had some concern with sign (my values should range [-1:1], not [0:1]).

Part 2 : Computational Simulation of a Distance-Dependent Shunting network

Part two of the assignment asks us to simulate a network of 100 neurons using the Distance-Dependent Shunting model. The model contains both excitatory and inhibitory terms which are weighted by the "distance" from one neuron to the others; in our case, the distance is quantified by the neuron's index within the network, ranging from 0 to 100.

$$\frac{dx_i}{dt} = -Ax_i + (B - x_i)\sum_{k\in D} D_{ki}I_k - (C + x_i)\sum_{l\in E} E_{li}I_l \tag{25}$$

Like our other models, this equation can be analytically solved by combining homogeneous and non-homogeneous terms:

$$a = A + \sum_{k\in D} D_{ki}I_k + \sum_{l\in E} E_{li}I_l \tag{26}$$

$$b = B\sum_{k\in D} D_{ki}I_k - C\sum_{l\in E} E_{li}I_l \tag{27}$$

$$x(t) = Ce^{-at} + b \tag{28}$$

The equilibrium solution is then

$$x_i = \frac{B\sum\limits_{k\in D} D_{ki}I_k - C\sum\limits_{l\in E} E_{li}I_l}{A + \sum\limits_{k\in D} D_{ki}I_k + \sum\limits_{l\in E} E_{li}I_l} \tag{29}$$

I chose to model this network with wrap-around boundary conditions, meaning neurons close to the end of our index are treated as neighbors to those at the beginning. This effects the value of our sum terms, but the means of solving the equation remain unchanged. These sum terms are weighted by the "distance" of a neuron to its neighbors.

$$D_{ki} = e^{-\frac{(k-i)^2}{F^2}} \tag{30}$$

$$E_{li} = 0.5e^{-\frac{(l-i)^2}{G^2}} \tag{31}$$

where F and G are weighting parameters. We throw away any weights less than 0.01, leaving us with a "radius" of neighbor neurons which will impact our neuron of interest;

$$r_D = \sqrt{-F^2 ln(0.01)} \tag{32}$$

$$r_E = \sqrt{-G^2 ln(0.02)} \tag{33}$$

This radius is an integer value, and any neurons outside of it (meaning outside of $i \pm R$ can be ignored in our sum.)

We simulate this equation for four different input current schemes and two different weighting parameters for a total of eight plots. The equilibrium values were calculated by simulating the distance-dependent shunting equation for a reasonably large period of time; with a time step of 0.05 seconds and a maximum time of 100 seconds, there are a total of 2000 integration steps.

The simulation our 1-d network was carried out using the RK4 approximation, which I've simply adapted to a general form and will probably paste around in future assignments.

Changing the number of neurons and maximum time have impacts on the runtime; in serial I measured a 2.39s runtime for a $tMax = 10,000$ with $n = 100$ neurons. The problem at hand is quite parallelizable, however, and most of the computation time is due to the RK4 simulation.

By parallelizing this (using an attached OpenCL kernel) I managed to get a runtime of 0.39s for $tMax = 10,000$ and $n = 1024$ neurons, which is a huge increase. This increase, however, became negligible for our problem wherein something like $tMax = 100$, $n = 100$ at 0.2s proved to be more than ample time to let our system approach equilibrium.

Below you will find my code for running a Runge-Kutta simulation on all neurons given the input current and parameters.

```
void simulate(T A, T B, T C, T F, T G, T *I)
{
  // loop variable and current sums
  int i;
  T Esum, Isum;
  // Our sum-radius (nothing outside of it should be included)
  int rE = sqrt(-F*F*log(0.01));
  int rI = sqrt(-G*G*log(0.02));
  // for all N neurons
  for (i=0;i<N;i++)
    {
      Esum = 0, Isum = 0;
      int k;
      // Calculate the excitatory and inhibitory sums
      for (k=-rE;k<=rE;k++)
        {
          int r = (i+k+N)%N;
          T e = -(k/F)*(k/F);
          Esum += I[r]*exp(e);
        }
      // Note the wrap around (modulus %) condition
      for (k=-rI;k<=rI;k++)
        {
          int r = (i+k+N)%N;
          T e = -(k/G)*(k/G);
          Isum += 0.5*I[r]*exp(e);
        }
      // Find equilibrium values over a long period of time
      T x=0,t=0;
      // I can't explain the /100, other than numerical error, but
      // EQ values should still be the same
      T a = (-A-Esum-Isum)/100;
      T b = (B*Esum-C*Isum)/100;
      do {
         x=solveRK(x,a,b);
         t+=DT;
      } while (t<=T_MAX);
      // write data to file
      fprintf(output,"%d\t%lf\n",i+1,x);
    }
}
```

Here we see my solveRK method and eulerAdvance method, as well an OpenCL kernel that could call them once passed in the a and b values of our general form differential equation.

```
T solveRK (T x0, T a, T b)
{
  T k1,k2,k3,k4,s;
  k1 = a*x0+b;
  k2 = a*eulerAdvance(x0,k1,DT/2)+b;
  k3 = a*eulerAdvance(x0,k2,DT/2)+b;
  k4 = a*eulerAdvance(x0,k3,DT)+b;
  s = (k1+2*k2+2*k3+k4);
  return eulerAdvance(x0,s,DT/6.0);
}

T eulerAdvance(T x0, T v, T dt)
{
  return x0 + dt * v;
}

__kernel void solveEQ
(__global const float* a, __global const float* b, __global float* out, int n)
{
    int i = get_global_id(0);
    if (i >= n)
        return;

    float x = 0.0f;
    float t = 0.0f;
    float ay = a[i];
    float be = b[i];

    while (t<=T_MAX)
    {
       x = solveRK(x,ay,be);
       t+=DT;
     }
     out[i] = x;
}
```

Now we see the plots from this simulation, which vary in input current scheme and parameters F and G. Again, the initial values for A, B, and C are 1, 10, and 1.5, respectively.

The first output plot shows the neuron equilibriums for the following current scheme:

$$I = \begin{cases} 10 & \text{if } x \in [1:24] \text{ or } [76:100] \\ 80 & \text{if } x \in [25:75] \end{cases}$$

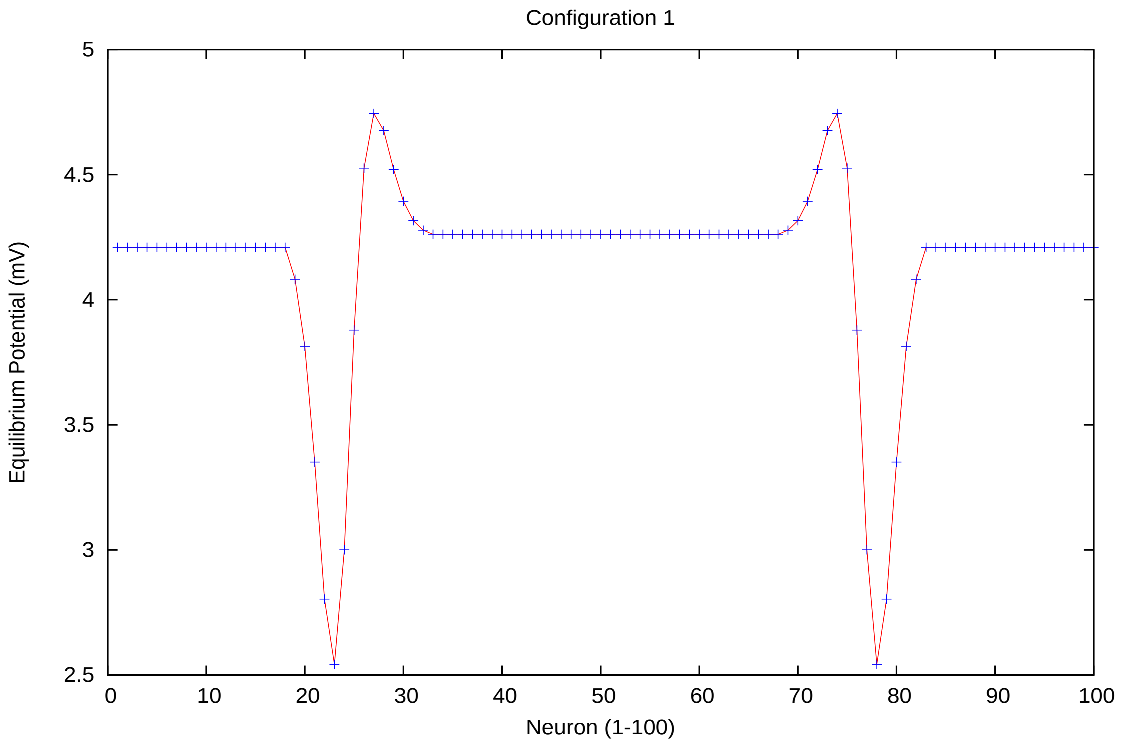


Figure 1: Input Scheme 1, F=2, G=4

The first simulation plots an input current that steps from a value of 10 to a value of 80 between neurons 25 $\rightarrow$ 74. Note the effect of neighboring neurons as the current changes from a low to high value.

$$I = \begin{cases} 10 & \text{if } x \in [1:24] \text{ or } [76:100] \\ 80 & \text{if } x \in [25:75] \end{cases}$$

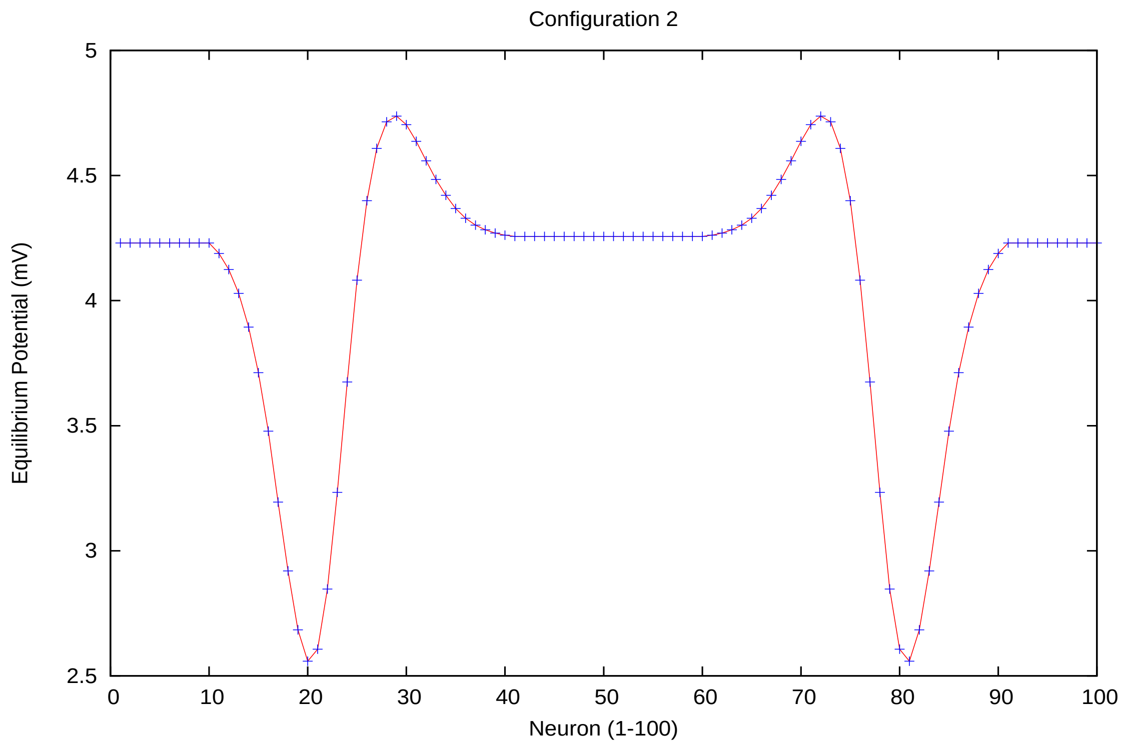


Figure 2: Input Scheme 1, F=4, G=8

I noticed that, in general, the increase in F and G (or the "radius" of our Gaussian) seemed to smoth out values; I interpreted this smoothness as a supression of inputs whose differences were relatively small, who could have been approximated as being uniform with one another. I think this is what was meant by uniform suppression, though I am not sure.

$$I = \begin{cases} 1 & \text{if } x \in [1:24] \text{ or } [76:100] \\ 8 & \text{if } x \in [25:75] \end{cases}$$

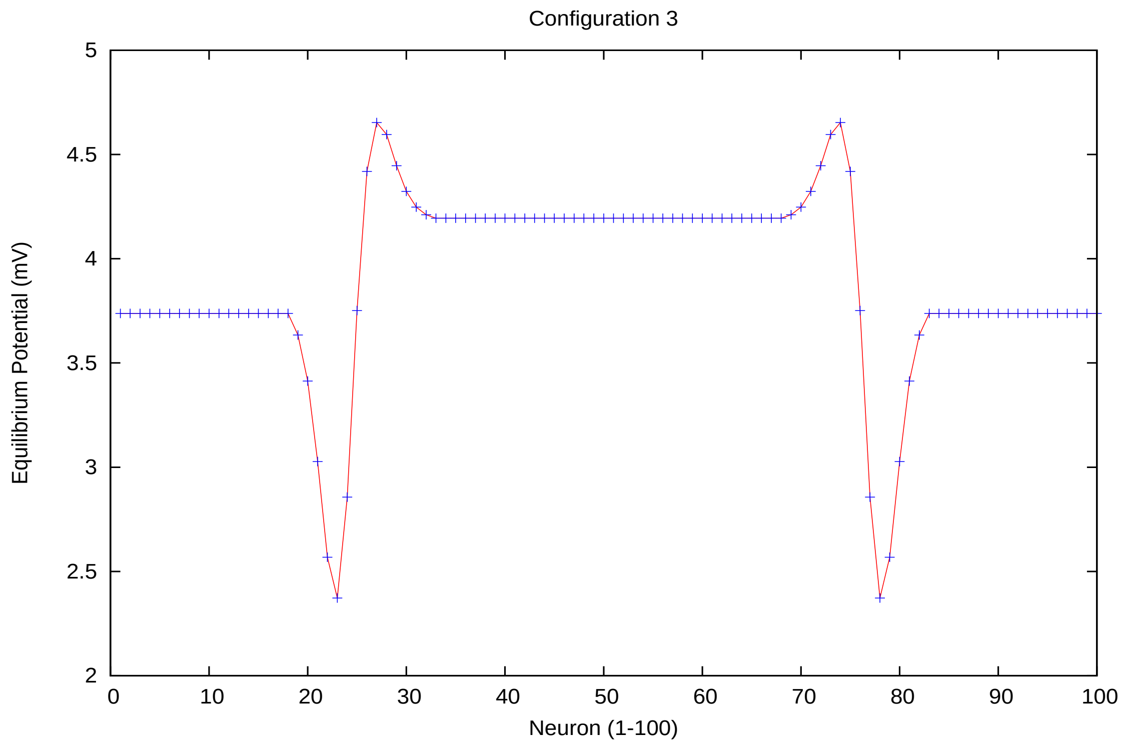


Figure 3: Input Scheme 2, F=2, G=4

Since we are using a Distant-Dependent Shunting model, the neurons inhibit one another and generate the same plot as before despite a significantly lower input current.

$$I = \begin{cases} 1 & \text{if } x \in [1:24] \text{ or } [76:100] \\ 8 & \text{if } x \in [25:75] \end{cases}$$

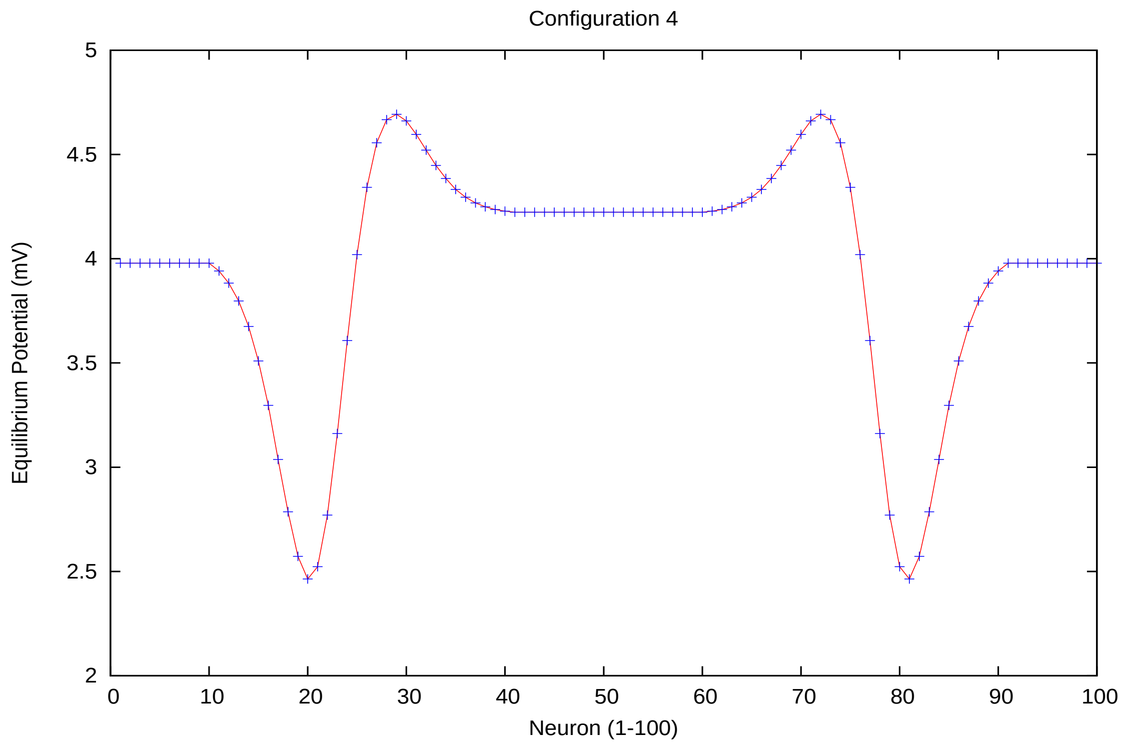


Figure 4: Input Scheme 2, F=4, G=8

Similar to before, where an increase in kernel width lead to smoother results. Note the areas of activity occur around changes in input current, and are quick to level off in areas of uniform intensity.

$$I = \begin{cases} 10 & \text{if } x \in [1:24] \text{ or } [76:100] \\ 80 & \text{if } x \in [25:75] \end{cases}$$

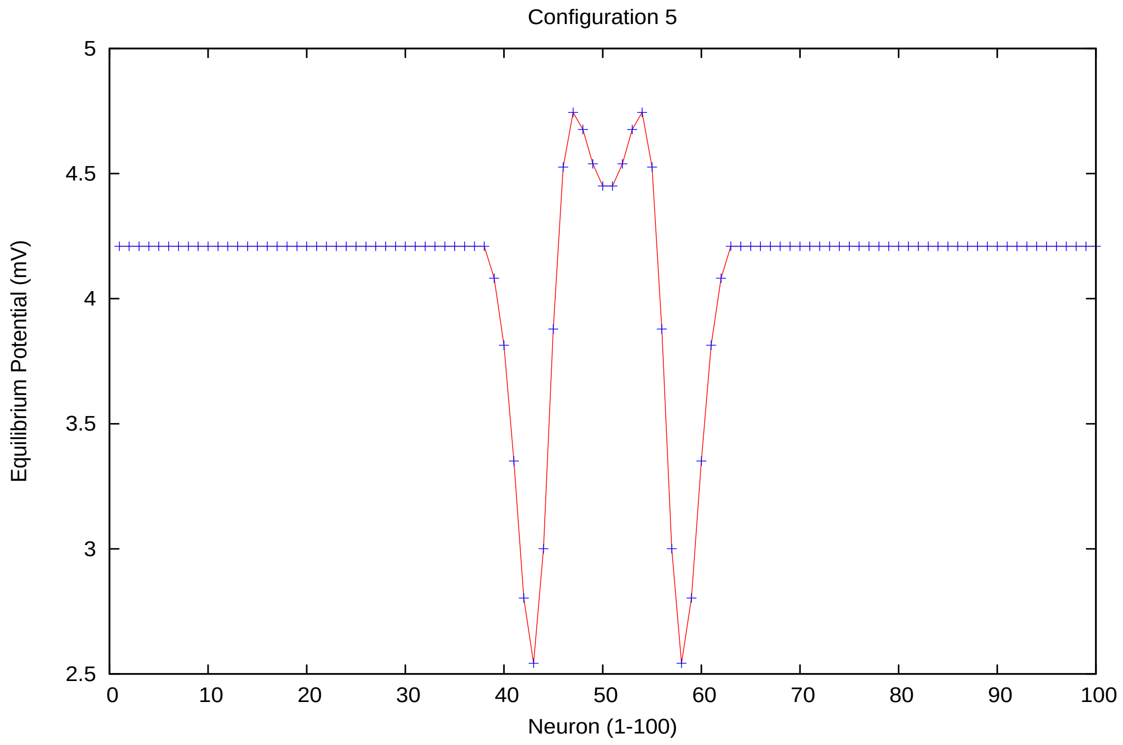


Figure 5: Input Scheme 3, F=2, G=4

This was the first of my plots that I think look a bit strange; I'd assume that given the short nature of our current-step we do not allow the neighbors enough space to level off their equilibrium differences. The longer step widths mean that there are fewer areas with high levels of comptetition, and by minimizing this width we can see our radius variable will now have sharper changes to take into account.

$$I = \begin{cases} 10 & \text{if } x \in [1:44] \text{ or } [56:100] \\ 80 & \text{if } x \in [45:55] \end{cases}$$

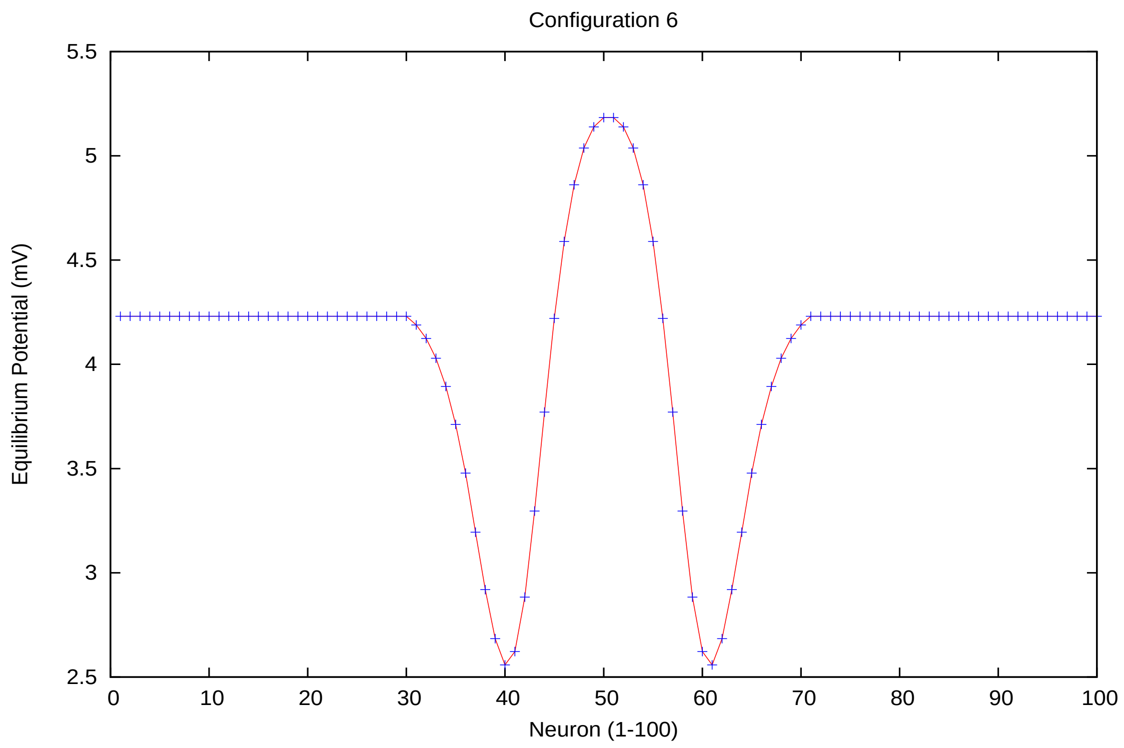


Figure 6: Input Scheme 3, F=4, G=8

We see similar behavior here, although the increased kernel width has allowed the middle area to remain smooth. This is an effect of uniform suppression, in which intensities one could see as relatively uniform were approximated as such.

$$I = \begin{cases} 1 & \text{if } x \in [1:10] \\ i - 9 \text{ or } (2 \rightarrow 81) & \text{if } x \in [11:89] \\ 82 & \text{if } x \in [90:100] \end{cases}$$

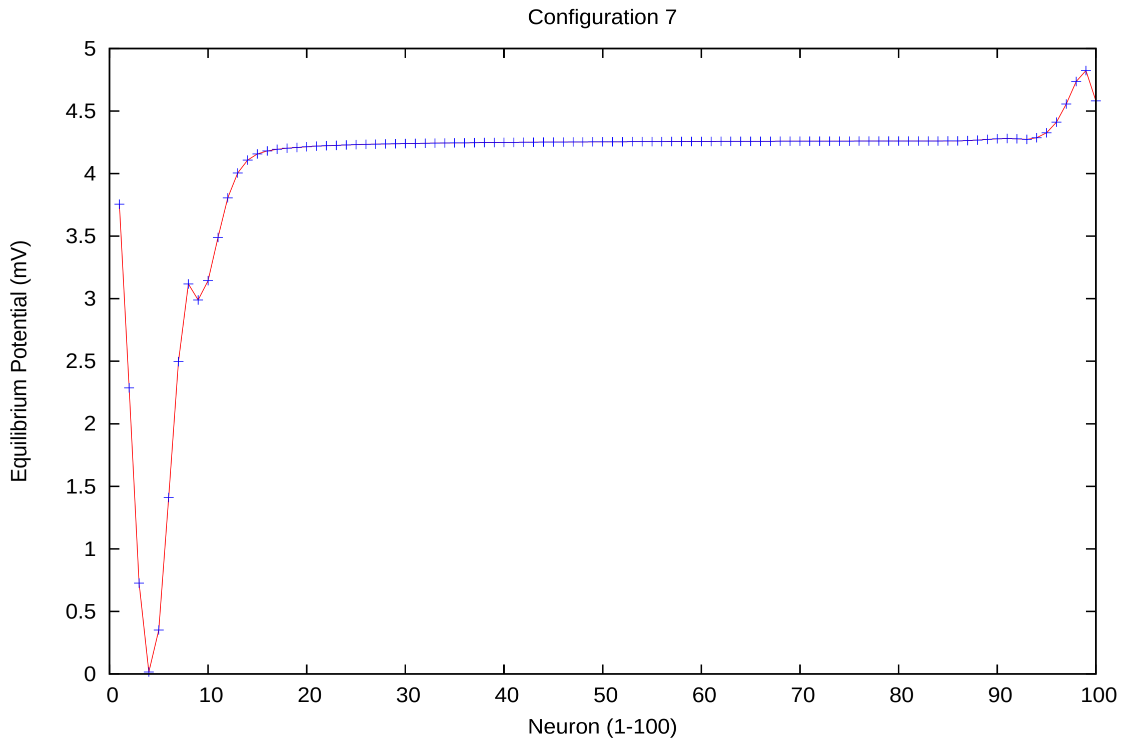


Figure 7: Input Scheme 4, F=2, G=4

These may the plots containing “artifacts” due to the wrap-around condition. I’m actually not sure what constitutes an artifact, although I assume the sharp changes near the beginning and end of the plot are due to the sharp difference in current intensity between the beginning and end of the network.

$$I = \begin{cases} 1 & \text{if } x \in [1:10] \\ i - 9 \text{ or } (2 \rightarrow 81) & \text{if } x \in [11:89] \\ 82 & \text{if } x \in [90:100] \end{cases}$$

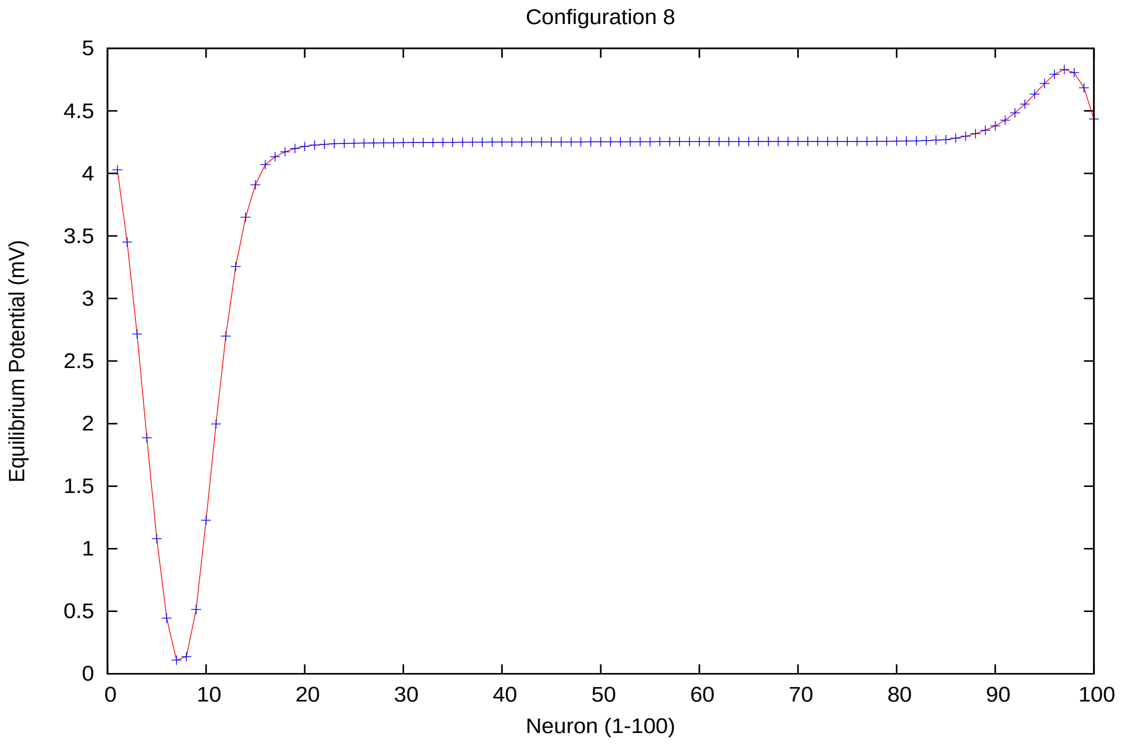


Figure 8: Input Scheme 4, F=4, G=8

The smoothing behavior is expected here, and the sharp changes should again be attributed to the wrap-around condition. I suppose if I had padded the edges with their border-value (i.e plenty of 1's toward -x and plenty of 82's toward +x). Still, I would be hesitant to compare this to our initial ramped current, and the difference between the two should be adapted to the distance-dependent nature of the network, in which only local neurons inhibit the neuron at hand.