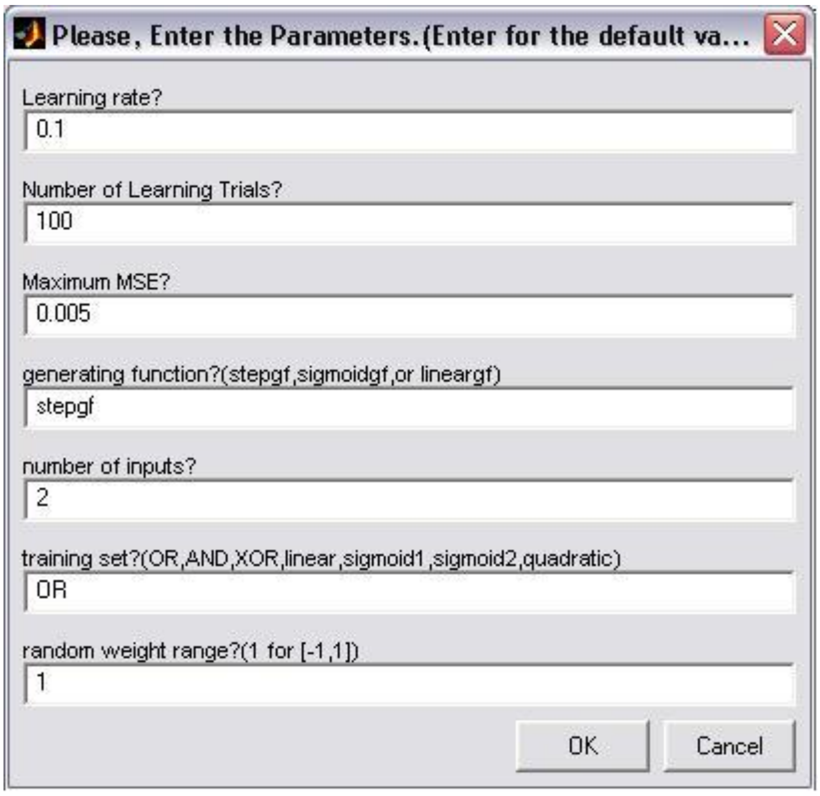
BME285J Introduction to Theoretical/Computational Neuroscience
Yul Young Park 02/05/04

**1a. Program in *matlab* a preceptron with a perceptron learning rule and solve the OR, AND and XOR problems.**

➔ I programmed a perceptron and solve OR, AND problem. However, XOR problem cannot be solved with only one perceptron because XOR is not linearly separable
The files are:
-'adaline.m' => This is main function and documentation is included, and just type 'adalin' to run. Then, you can type the parameter value in the 'prompt window' and the results will be return as the figure of MSE convergence and the command line output. It train the perceptron with training data set and after that, test the perceptron with test data set
-'stepgf.m' => implement step function
-'testset.m'=> test data sets are defined
-'trainingset.m' => training data sets are defined.


Please, Enter the Parameters.(Enter for the default va...

Learning rate?
0.1

Number of Learning Trials?
100

Maximum MSE?
0.005

generating function?(stepgf,sigmoidgf,or lineargf)
stepgf

number of inputs?
2

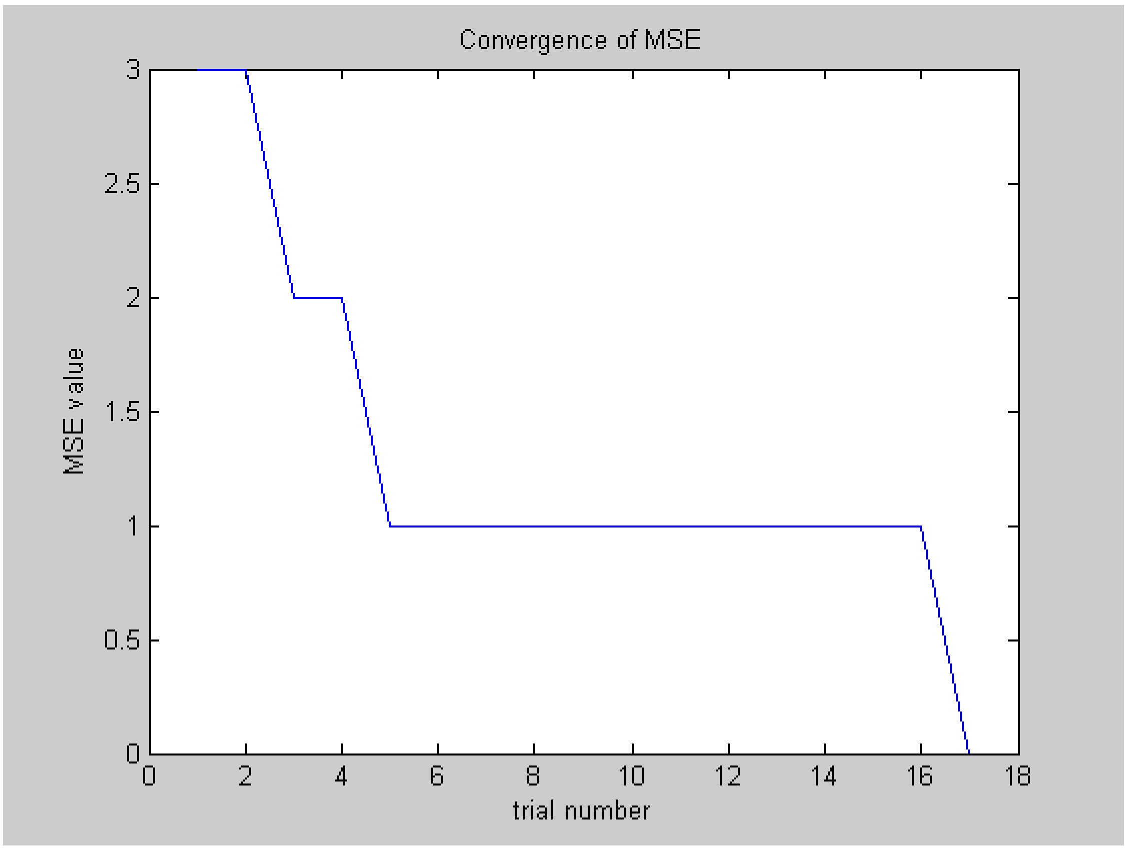
training set?(OR,AND,XOR,linear,sigmoid1,sigmoid2,quadratic)
OR

random weight range?(1 for [-1,1])
1

From the above ‘prompt window’, you can choose a training set(OR,AND,XOR) and see the program solve that problem. For example, when you choose ‘**OR**’ and use the default value of the other parameter, you can see the result on the terminal

```
>> adaline
Trial #1:    Stimulus #2,    MSE = 3,    error signal = 4,    # of misclassification = 1
Trial #2:    Stimulus #4,    MSE = 3,    error signal = 0,    # of misclassification = 0
Trial #3:    Stimulus #3,    MSE = 2,    error signal = 4,    # of misclassification = 1
Trial #4:    Stimulus #3,    MSE = 2,    error signal = 0,    # of misclassification = 0
Trial #5:    Stimulus #2,    MSE = 1,    error signal = 4,    # of misclassification = 1
Trial #6:    Stimulus #3,    MSE = 1,    error signal = 0,    # of misclassification = 0
Trial #7:    Stimulus #3,    MSE = 1,    error signal = 0,    # of misclassification = 0
Trial #8:    Stimulus #2,    MSE = 1,    error signal = 4,    # of misclassification = 1
Trial #9:    Stimulus #3,    MSE = 1,    error signal = 0,    # of misclassification = 0
Trial #10:    Stimulus #2,    MSE = 1,    error signal = 4,    # of misclassification = 1
Trial #11:    Stimulus #2,    MSE = 1,    error signal = 0,    # of misclassification = 0
Trial #12:    Stimulus #1,    MSE = 1,    error signal = 0,    # of misclassification = 0
Trial #13:    Stimulus #1,    MSE = 1,    error signal = 0,    # of misclassification = 0
Trial #14:    Stimulus #3,    MSE = 1,    error signal = 4,    # of misclassification = 1
Trial #15:    Stimulus #4,    MSE = 1,    error signal = 0,    # of misclassification = 0
Trial #16:    Stimulus #1,    MSE = 1,    error signal = 0,    # of misclassification = 0
Trial #17:    Stimulus #2,    MSE = 0,    error signal = 4,    # of misclassification = 1
Training perceptron is completed. Now, the perceptron will be tested using test data set

test_pattern =
  1    1
  1 -1
-1    1
-1 -1

test_output =
  1
  1
  1
-1
```

Also, you can see the MSE convergence plot as following


Convergence of MSE
MSE value
trial number
0
0.5
1
1.5
2
2.5
3
0
2
4
6
8
10
12
14
16
18

**1b. Implement a one layer linear and sigmoidal network, fit a 1D a linear, a sigmoid and a quadratic function, for both networks. Linear y=-0.5+2x where x is in the -1,1 range.**
**Sigmoidal: y=sig(x),y=sig(x-0.5) in the -1,1 range, quadratic: y=x(x-0.5), in the range [0..1].**
→ Here, the necessary files are the same as hw1a with more function implementation file.
-'adaline.m' => This is main function and documentation is included, and just type 'adalin' to run. Then, you can type the parameter value and the results in terms of MSE plot and the command line output. It train the perceptron with training data set and after that, test the perceptron with test data set
-'stepgf.m' => implement step function
-'sigmoidgf.m' => implement sigmoid function
- 'lineargf.m' => implement linear function
-'testset.m'=> test data sets are defined
-'trainingset.m' => training data sets are defined.

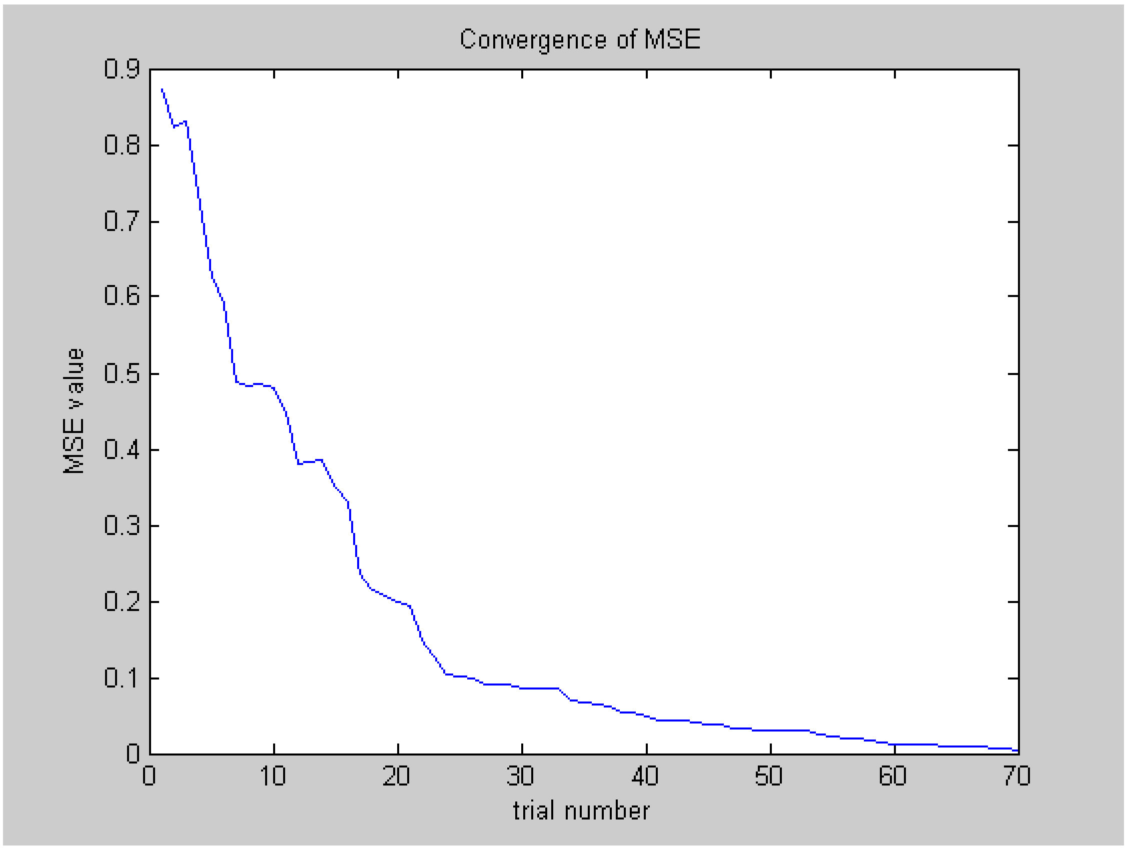
From the 'prompt window', you have to set value of 'generating function' as '**lineargf**' for the one layer linear network, '**sigmoidgf**' for the one layer sigmoid network. The linear function, sigmoid function, and quadratic function which are fitted by both networks are set by making the value of the parameter 'train set'as '**linear**' for the linear function y=-0.5+2x, '**sigmoid1**' for y=sig(x), '**sigmoid2**' for y=sig(x-0.5) in [-1,1], and '**quadratic**' for y=x(x-0.5) in [0,1]
For example, if you set the 'generating function' as '**lineargf**' and 'train set'as '**linear**', you will have

```
Trial #1:   Stimulus #7,   MSE = 0.87121,   error signal = 0.00040488,   # of misclassification = 0
Trial #2:   Stimulus #11,   MSE = 0.82332,   error signal = 0.25202,   # of misclassification = 0
Trial #3:   Stimulus #1,   MSE = 0.82896,   error signal = 0.71858,   # of misclassification = 0
Trial #4:   Stimulus #13,   MSE = 0.73758,   error signal = 0.60766,   # of misclassification = 0
Trial #5:   Stimulus #15,   MSE = 0.62783,   error signal = 0.88038,   # of misclassification = 1
Trial #6:   Stimulus #12,   MSE = 0.59097,   error signal = 0.23132,   # of misclassification = 0
Trial #7:   Stimulus #17,   MSE = 0.48706,   error signal = 1.0217,   # of misclassification = 1
Trial #8:   Stimulus #4,   MSE = 0.4834,   error signal = 0.30435,   # of misclassification = 0
Trial #9:   Stimulus #8,   MSE = 0.48486,   error signal = 0.0021592,   # of misclassification = 0
Trial #10:    Stimulus #2,    MSE = 0.4809,    error signal = 0.45574,    # of misclassification = 0
Trial #11:    Stimulus #12,    MSE = 0.44929,    error signal = 0.19579,    # of misclassification = 0
Trial #12:    Stimulus #16,    MSE = 0.37987,    error signal = 0.63072,    # of misclassification = 1
Trial #13:    Stimulus #6,    MSE = 0.38289,    error signal = 0.065711,    # of misclassification = 0
Trial #14:    Stimulus #6,    MSE = 0.38671,    error signal = 0.05031,    # of misclassification = 0
Trial #15:    Stimulus #14,    MSE = 0.34949,    error signal = 0.29758,    # of misclassification = 1
Trial #16:    Stimulus #13,    MSE = 0.32898,    error signal = 0.156,    # of misclassification = 0
Trial #17:    Stimulus #21,    MSE = 0.23625,    error signal = 1.1583,    # of misclassification = 0
Trial #18:    Stimulus #17,    MSE = 0.21345,    error signal = 0.29471,    # of misclassification = 0
Trial #19:    Stimulus #6,    MSE = 0.20544,    error signal = 0.13689,    # of misclassification = 0
Trial #20:    Stimulus #5,    MSE = 0.19805,    error signal = 0.15854,    # of misclassification = 0
Trial #21:    Stimulus #5,    MSE = 0.19456,    error signal = 0.11835,    # of misclassification = 0
Trial #22:    Stimulus #20,    MSE = 0.1487,    error signal = 0.56604,    # of misclassification = 0
Trial #23:    Stimulus #19,    MSE = 0.12674,    error signal = 0.30578,    # of misclassification = 0
Trial #24:    Stimulus #1,    MSE = 0.10497,    error signal = 0.35486,    # of misclassification = 0
Trial #25:    Stimulus #14,    MSE = 0.10081,    error signal = 0.044143,    # of misclassification = 1
Trial #26:    Stimulus #14,    MSE = 0.097877,    error signal = 0.035044,    # of misclassification = 1
Trial #27:    Stimulus #4,    MSE = 0.090888,    error signal = 0.12214,    # of misclassification = 0
Trial #28:    Stimulus #9,    MSE = 0.091026,    error signal = 0.0026527,    # of misclassification = 0
Trial #29:    Stimulus #12,    MSE = 0.089713,    error signal = 0.010219,    # of misclassification = 0
Trial #30:    Stimulus #15,    MSE = 0.084902,    error signal = 0.056555,    # of misclassification = 0
Trial #31:    Stimulus #10,    MSE = 0.084914,    error signal = 0.00089442,    # of misclassification = 0
Trial #32:    Stimulus #10,    MSE = 0.08494,    error signal = 0.00072287,    # of misclassification = 0
Trial #33:    Stimulus #11,    MSE = 0.084833,    error signal = 0.0005664,    # of misclassification = 0
Trial #34:    Stimulus #21,    MSE = 0.06849,    error signal = 0.25112,    # of misclassification = 0
Trial #35:    Stimulus #8,    MSE = 0.066353,    error signal = 0.024828,    # of misclassification = 0
Trial #36:    Stimulus #15,    MSE = 0.065063,    error signal = 0.024652,    # of misclassification = 0
Trial #37:    Stimulus #6,    MSE = 0.060667,    error signal = 0.056609,    # of misclassification = 0
Trial #38:    Stimulus #19,    MSE = 0.054599,    error signal = 0.10272,    # of misclassification = 0
Trial #39:    Stimulus #15,    MSE = 0.054301,    error signal = 0.013346,    # of misclassification = 0
Trial #40:    Stimulus #5,    MSE = 0.047952,    error signal = 0.07524,    # of misclassification = 0
Trial #41:    Stimulus #19,    MSE = 0.044317,    error signal = 0.071265,    # of misclassification = 0
Trial #42:    Stimulus #10,    MSE = 0.043403,    error signal = 0.0066541,    # of misclassification = 0
Trial #43:    Stimulus #11,    MSE = 0.043105,    error signal = 0.0015646,    # of misclassification = 0
Trial #44:    Stimulus #18,    MSE = 0.041528,    error signal = 0.040343,    # of misclassification = 0
Trial #45:    Stimulus #8,    MSE = 0.03898,    error signal = 0.023349,    # of misclassification = 0
Trial #46:    Stimulus #14,    MSE = 0.03907,    error signal = 0.0030625,    # of misclassification = 0
Trial #47:    Stimulus #4,    MSE = 0.033084,    error signal = 0.07192,    # of misclassification = 0
Trial #48:    Stimulus #10,    MSE = 0.032813,    error signal = 0.0024002,    # of misclassification = 0
Trial #49:    Stimulus #18,    MSE = 0.030835,    error signal = 0.037844,    # of misclassification = 0
Trial #50:    Stimulus #10,    MSE = 0.030325,    error signal = 0.0038608,    # of misclassification = 0
Trial #51:    Stimulus #11,    MSE = 0.030182,    error signal = 0.00075412,    # of misclassification = 0
Trial #52:    Stimulus #16,    MSE = 0.029691,    error signal = 0.013754,    # of misclassification = 0
Trial #53:    Stimulus #15,    MSE = 0.029685,    error signal = 0.0055959,    # of misclassification = 0
Trial #54:    Stimulus #3,    MSE = 0.023964,    error signal = 0.069711,    # of misclassification = 0
Trial #55:    Stimulus #4,    MSE = 0.021183,    error signal = 0.038152,    # of misclassification = 0
Trial #56:    Stimulus #16,    MSE = 0.02022,    error signal = 0.014928,    # of misclassification = 0
Trial #57:    Stimulus #7,    MSE = 0.019409,    error signal = 0.010794,    # of misclassification = 0
Trial #58:    Stimulus #4,    MSE = 0.01782,    error signal = 0.025878,    # of misclassification = 0
Trial #59:    Stimulus #21,    MSE = 0.014038,    error signal = 0.055263,    # of misclassification = 0
Trial #60:    Stimulus #3,    MSE = 0.012258,    error signal = 0.026725,    # of misclassification = 0
Trial #61:    Stimulus #12,    MSE = 0.012179,    error signal = 0.00074706,    # of misclassification = 0
Trial #62:    Stimulus #12,    MSE = 0.012121,    error signal = 0.00060378,    # of misclassification = 0
Trial #63:    Stimulus #18,    MSE = 0.010998,    error signal = 0.017191,    # of misclassification = 0
Trial #64:    Stimulus #5,    MSE = 0.010079,    error signal = 0.012703,    # of misclassification = 0
Trial #65:    Stimulus #18,    MSE = 0.0091899,    error signal = 0.013951,    # of misclassification = 0
Trial #66:    Stimulus #15,    MSE = 0.0090775,    error signal = 0.0028373,    # of misclassification = 0
Trial #67:    Stimulus #20,    MSE = 0.0083758,    error signal = 0.015568,    # of misclassification = 0
Trial #68:    Stimulus #2,    MSE = 0.0063345,    error signal = 0.02473,    # of misclassification = 0
Trial #69:    Stimulus #4,    MSE = 0.0055304,    error signal = 0.010567,    # of misclassification = 0
Trial #70:    Stimulus #21,    MSE = 0.0046767,    error signal = 0.014703,    # of misclassification = 0
Training perceptron is completed. Now, the perceptron will be tested using test data set

test_pattern =


   -1
-0.9
-0.8
-0.7
-0.6
-0.5
-0.4
```

```
-0.3
-0.2
-0.1
      0
  0.1
  0.2
  0.3
  0.4
  0.5
  0.6
  0.7
  0.8
  0.9
      1


test_output =


  -2.3757
  -2.1867
  -1.9978
  -1.8089
  -1.6199
   -1.431
  -1.2421
  -1.0531
-0.86421
-0.67528
-0.48634
-0.29741
-0.10848
0.080458
  0.26939
  0.45833
  0.64726
  0.83619
```

```
1.0251
1.2141
 1.403
```

and the MSE convergence plot will be

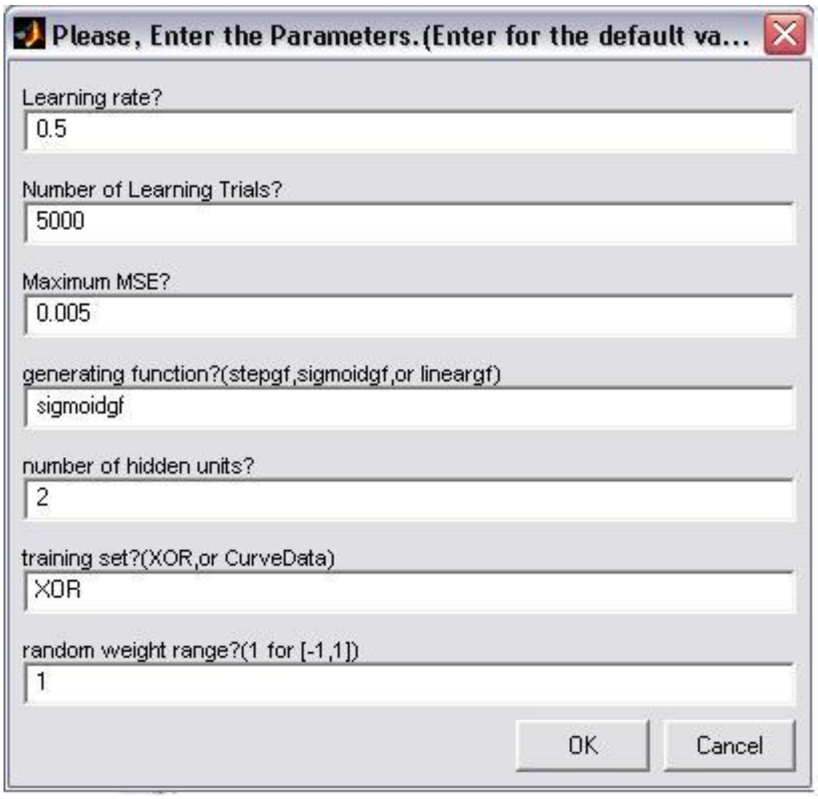


**1c. Program a 2 layer backprop network in matlab, solve the XOR problem. Fit the curve: y=x(x-0.5) between 0 and 1, how many hidden units did you need?**

→ Here are the necessary filenames.

-'backpropagation.m' => this is the main program of the 2-layer backpropagation network. If you type 'backpropagation', then you can see the 'prompt window' which was shown above hw1a, hw1b. After you type the parameter value, it is executed and shows the results through a figure showing MSE convergence and the terminal output.
-'sigmoidgf.m' => implement sigmoid function
- 'lineargf.m' => implement linear function
-'testset.m'=> test data sets are defined
-'trainingset.m' => training data sets are defined.

The 'prompt window' is basically same as the above one, but it looks slightly different because 2 layer backpropagation network needs more additional parameters such as the number of hidden unit. The 'prompt window' looks like


Please, Enter the Parameters.(Enter for the default va...

Learning rate?
0.5

Number of Learning Trials?
5000

Maximum MSE?
0.005

generating function?(stepgf,sigmoidgf,or lineargf)
sigmoidgf

number of hidden units?
2

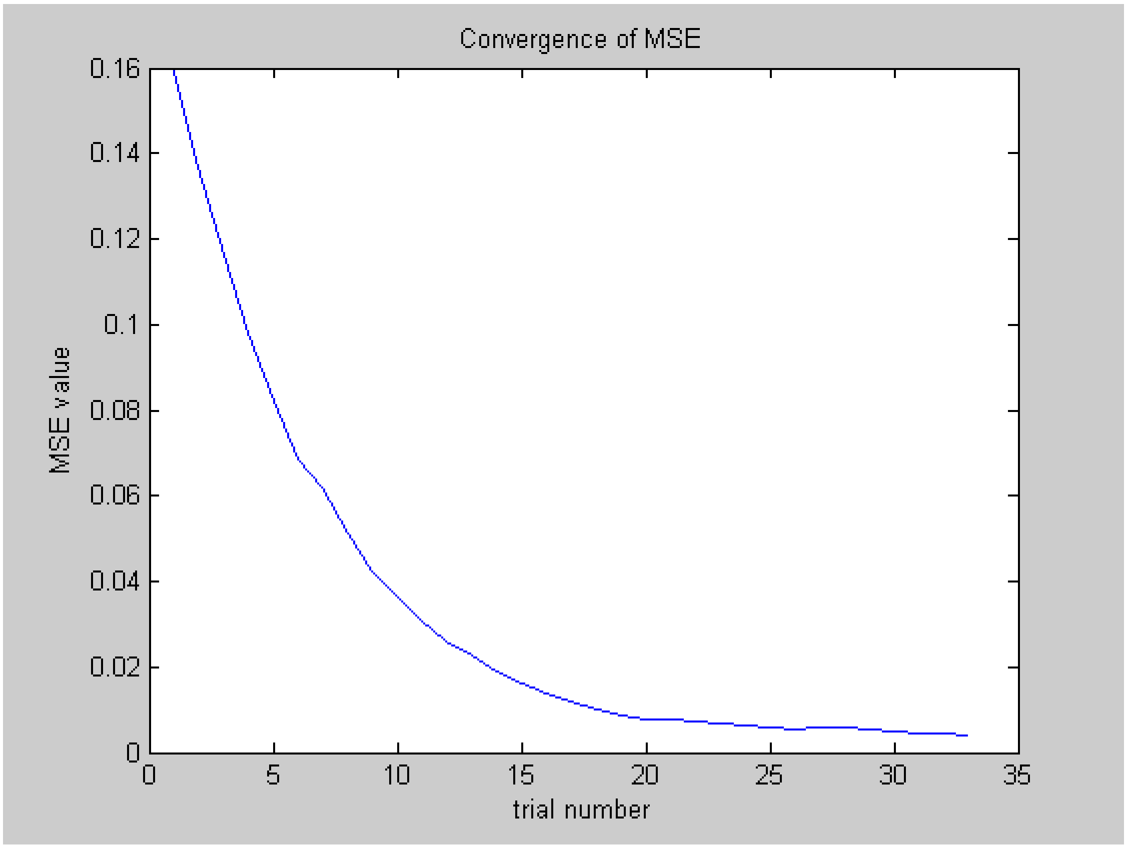
training set?(XOR,or CurveData)
XOR

random weight range?(1 for [-1,1])
1

And, I think 'backpropagation.m' takes too much time in training. Even though I used large learning rate for quick convergence, it hardly converged. However, it could fit the curve y=x(x-0.5) well and fast. Only two hidden units were enough to fit the curve fast. For example, if you set ‘train set’ parameter as ‘**CurveData**’ instead of XOR, you will have

```
>> backpropagation
Trial #1:    Stimulus #3,    MSE = 0.15916,    error signal = 0.53169
Trial #2:    Stimulus #4,    MSE = 0.13674,    error signal = 0.49766
Trial #3:    Stimulus #3,    MSE = 0.1162,    error signal = 0.45969
Trial #4:    Stimulus #2,    MSE = 0.09779,    error signal = 0.39681
Trial #5:    Stimulus #5,    MSE = 0.08211,    error signal = 0.3707
Trial #6:    Stimulus #9,    MSE = 0.068492,    error signal = 0.095485
Trial #7:    Stimulus #3,    MSE = 0.062165,    error signal = 0.34438
Trial #8:    Stimulus #3,    MSE = 0.05128,    error signal = 0.31512
Trial #9:    Stimulus #7,    MSE = 0.042245,    error signal = 0.17892
Trial #10:    Stimulus #6,    MSE = 0.036296,    error signal = 0.21439
Trial #11:    Stimulus #6,    MSE = 0.030589,    error signal = 0.1962
Trial #12:    Stimulus #7,    MSE = 0.025865,    error signal = 0.1335
Trial #13:    Stimulus #2,    MSE = 0.022465,    error signal = 0.19605
Trial #14:    Stimulus #1,    MSE = 0.018912,    error signal = 0.14683
Trial #15:    Stimulus #5,    MSE = 0.016279,    error signal = 0.17137
Trial #16:    Stimulus #5,    MSE = 0.013822,    error signal = 0.15834
Trial #17:    Stimulus #4,    MSE = 0.011813,    error signal = 0.16133
Trial #18:    Stimulus #5,    MSE = 0.010094,    error signal = 0.13582
Trial #19:    Stimulus #1,    MSE = 0.0087507,    error signal = 0.096338
Trial #20:    Stimulus #9,    MSE = 0.0077838,    error signal = 0.0049173
Trial #21:    Stimulus #9,    MSE = 0.0075802,    error signal = 0.0045926
Trial #22:    Stimulus #7,    MSE = 0.0073894,    error signal = 0.059204
Trial #23:    Stimulus #8,    MSE = 0.0067495,    error signal = 0.024502
Trial #24:    Stimulus #2,    MSE = 0.0063733,    error signal = 0.10532
Trial #25:    Stimulus #9,    MSE = 0.0056804,    error signal = 0.0018274
Trial #26:    Stimulus #11,    MSE = 0.0055958,    error signal = 0.04696
Trial #27:    Stimulus #9,    MSE = 0.0060412,    error signal = 0.0023162
Trial #28:    Stimulus #5,    MSE = 0.005938,    error signal = 0.10384
Trial #29:    Stimulus #2,    MSE = 0.0053043,    error signal = 0.095528
Trial #30:    Stimulus #3,    MSE = 0.004784,    error signal = 0.10344
Trial #31:    Stimulus #10,    MSE = 0.004316,    error signal = 0.0099547
Trial #32:    Stimulus #4,    MSE = 0.0044543,    error signal = 0.10034
Trial #33:    Stimulus #10,    MSE = 0.0040385,    error signal = 0.011137
Training perceptron is completed. Now, the perceptron will be tested using test data set

test_pattern =


    0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
    1



test_output =


0.56158
0.56188
0.56217
0.56246
0.56274
0.56301
0.56327
0.56352
0.56376
0.56398
  0.5642
```

and the MSE convergence plot will be



Thank you for your time.

# Title : the spike train analysis of a fly H1 neuron

Term Project

BME385J Introduction to Theoretical/Computational Neuroscience

Yul Young Park

05/12/04

# Title: the spike train analysis of a fly H1 neuron

## Abstract

A fly H1 neuron spikes are statistically analyzed, and modeled by poisson distribution spikes and gamma distribution spikes. Through comparison of the coefficient of variation and Fanor factor between the H1 neuron response data and models, H1 neuron spikes show more variability in their inter-spike interval distribution and spike count distribution that that of the models, and which will be better explained by the stochastic features of the ion channel state transition and neurotransmitter release in the synaptic region.

## Introduction

Action potentials which are simplified as 0 or 1 digital signals in time domain are called spikes, and they contain the information of the stimulus in the form of their mean value and/or their timing. The first one is called rate coding which argues the stimuli parameters are included in the mean spike number of certain interval, and it is effective on the description of the time-independent stimuli. The latter one is known as temporal coding which says the spike timing reflects the various parameters of the stimuli. This temporal coding can be considered as discrete stochastic process and analyzed by probability and statistical methods. Thus, the spikes of a fly H1 neuron can be analyzed mainly based on the temporal coding concept with the help of statistical methods.

The spikes of a fly H1 neuron are modeled by two mathematical models: poission spike train and gamma spike train. Poisson spike train can be generated by the simple integrate-firing neuron with constant current input and random membrane threshold potential which has exponential distribution. The leaky intergrate-and-fire neuron is

$$\tau_m \frac{dV}{dt} = E_L - V + R_m I_e \qquad (1)$$

, and the exponential probability distribution of the random membrane threshold potential is

$$p(V) = (1/V_{th}) e^{-V/V_{th}} \qquad (2)$$

where Vth is the mean value of the distribution p(V). The assumptions of the above are the independency among the spikes, randomness of the each spike generation, and a uniform probability of occurrence in time. Equivalently, non leaky integrator with fixed threshold and poisson-distributed synaptic input generate the spike train of same distribution. However, real spikes are not only related to the other spikes, but also affected by the biophysical properties such as refractory period and spike rate adaptation.

The inter-spike intervals from the poisson spike train will show exponential distribution, and it is called 'inter-spike interval(ISI)' histogram. The variability of ISI distribution is described by the coefficient of variation(Cv) which is dimensionless number defined as the standard deviation( $\sigma_t$ )of ISI distribution divided by the mean( $\mu_t$ ) of ISI:

$$C_V = \frac{\sigma_t}{\mu_t} \qquad (3)$$

where $\mu_t = \int_0^\infty t \cdot p(t)dt$ and $\sigma_t^2 = \int_0^\infty (t-\mu_t)^2 \cdot p(t)dt$. The Cv can give a measure of short term variability based on the one of the above assumption that a spike time is only dependent on the very previous one, not further on the past history of the spikes.

The long-term variability beyond the previous spike interval can be obtained from the observation and count of the spikes over a time period, T. Since the spikes follow poisson distribution, the probability of having n spikes in the observation window, T is

$$p(n) = \frac{(T/\mu_t)^n}{n!} e^{-\frac{T}{\mu_t}} \qquad (4)$$

Upon this spike count probability distribution, the ratio of variance, V(T) to the mean, M(T) gives the long-term variability, called Fanor factor,F(T):

$$F(T) = \frac{V(T)}{M(T)} \qquad (5)$$

where $M(T) = \int_0^\infty n \cdot p(n)dn$ and $V(T) = \int_0^\infty (n - M(T))^2 \cdot p(n)dn$ with the window, T.

It is also known that $F(T) \cong C_V^2$ for large T.

On the other hand, gamma spike train can be generated by the simple integrate-firing neuron with constant current input and random membrane threshold potential which has gamma distribution:

$$p_n(V) = \frac{c_n V^{n-1}}{V_{th}^{n-1}} e^{-\frac{nV}{V_{th}}} \qquad (6)$$

where $c_n = \frac{n^n}{(n-1)!} \frac{1}{V_{th}}$. The Cv and Fanor factor of gamma spike train are defined in the same manner as that of poisson spike train.

Therefore, the modeling of H1 neuron spikes as the poisson spike train and gamma spike train can be refuted by the comparison of ISI distribution, Cv, and Fanor factor among these three different data set.

## Methods

Data of a fly H1 neuron response was given by Dayan/Abott which was accompanied by the problem 8 on the chapter8 of the textbook, "Theoretical neuroscience". The stimulus is white-noise and the 0/1 spike responses are included. The total recording data is 20min * 60sec * 500Hz = 600000 samples.

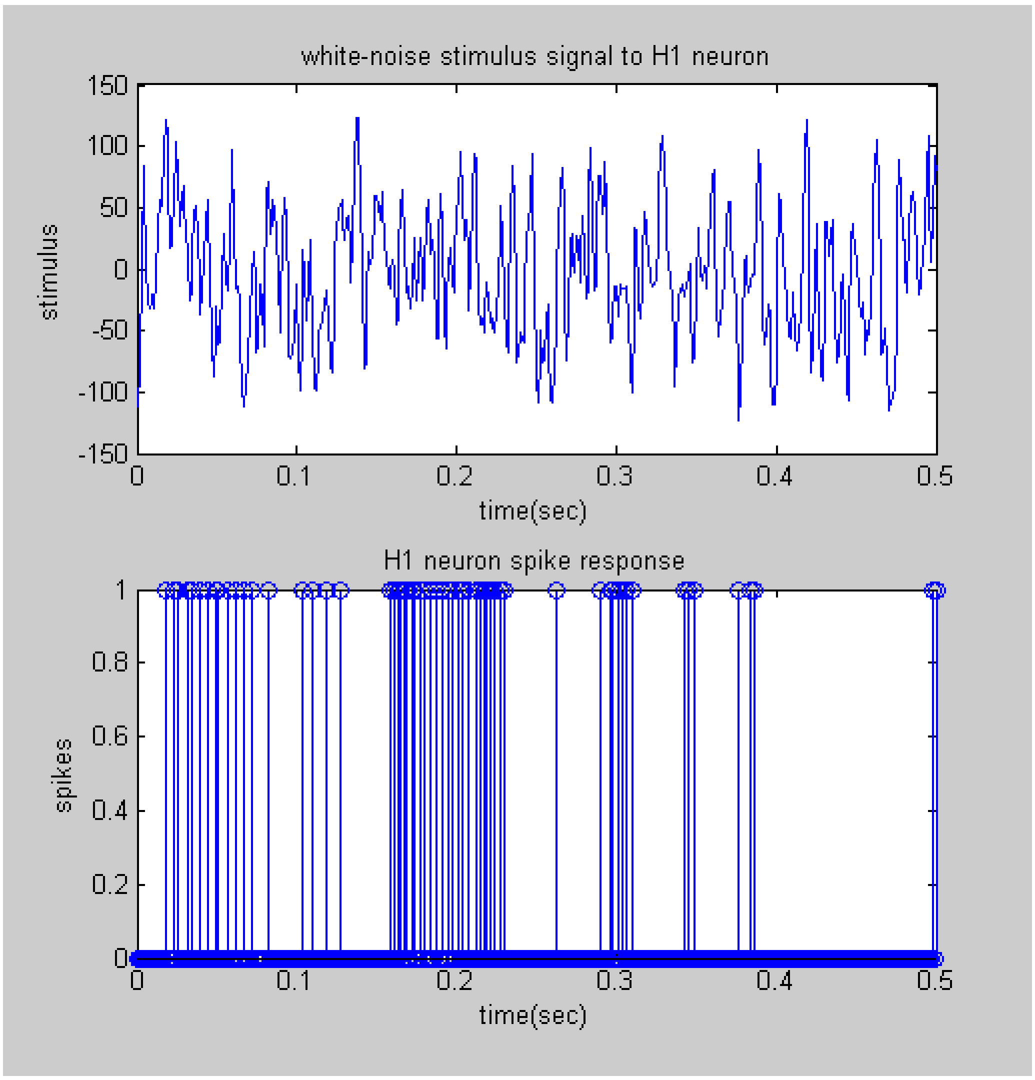


Figure1. A fly H1 neuron spike response and stimulus

The leaky integrate-fire neuron was used for the model neuron, and the membrane potential was calculated according to the numerical integration of equation (1) with the time step of 2msec in order to match the sampling period of H1 neuron data. Poisson spike train of 60sec duration was generated by the above model neuron and constant current input with the random threshold( refer to ‘poisson_spk_gen.m’ for the detail ). Thus, the resulting membrane potential and poisson spike train have been seen below

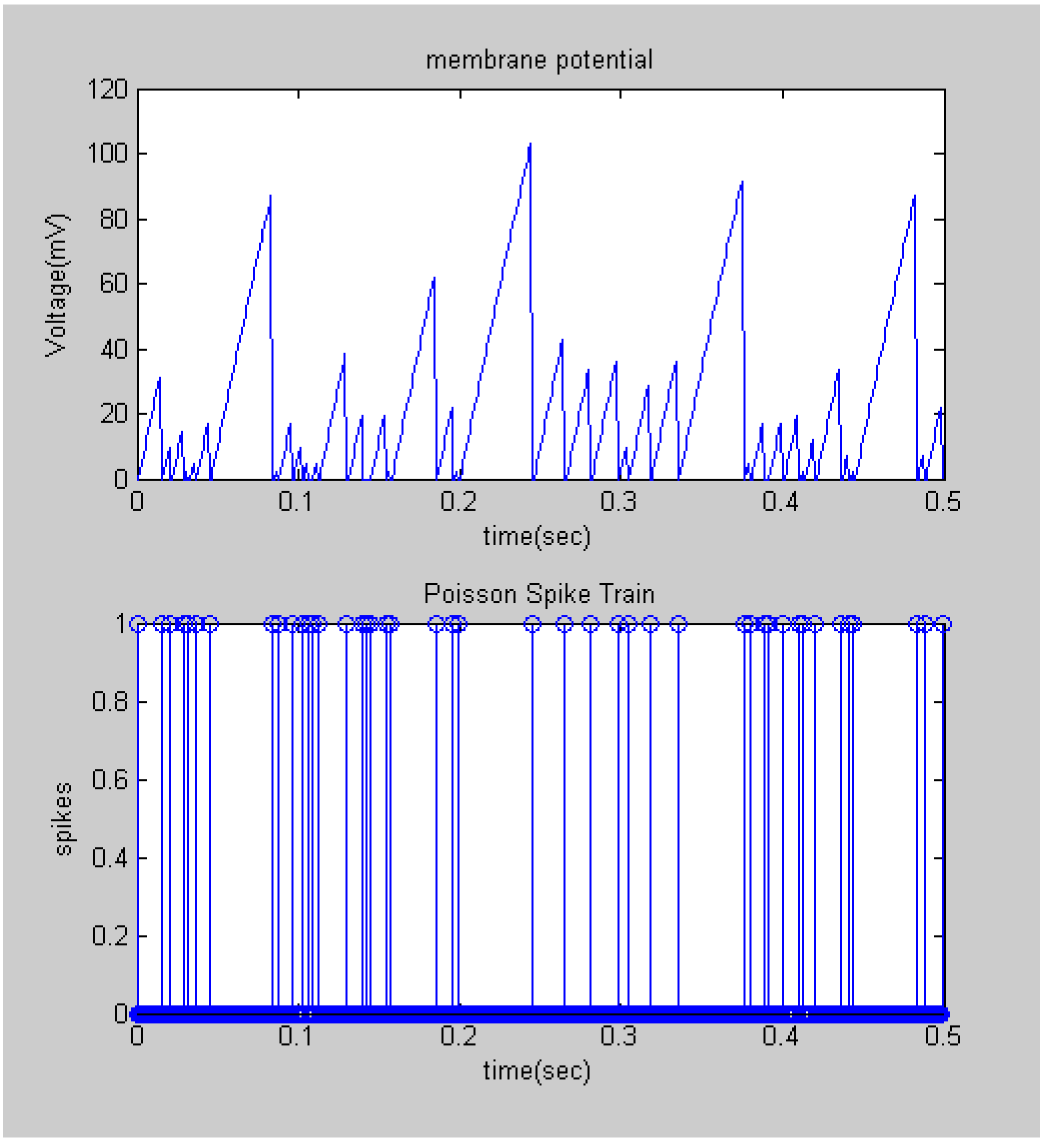


Figure2. Poisson spike train from simulation

Gamma spike train of 60sec duration was generated in the same way as that of poisson spike train except that the gamma distribution of order = 5 was used instead of exponential distribution for the random threshold generation. Actually, with gamma distribution of order =1, poisson spike train was acquired (refer to ‘gamma_spk_gen.m’ for the detail). The gamma distributed random number was generated by using ‘gamrnd’ function in the statistics toolbox in MATLAB. The resulting membrane potential and gamma spike train have been seen below

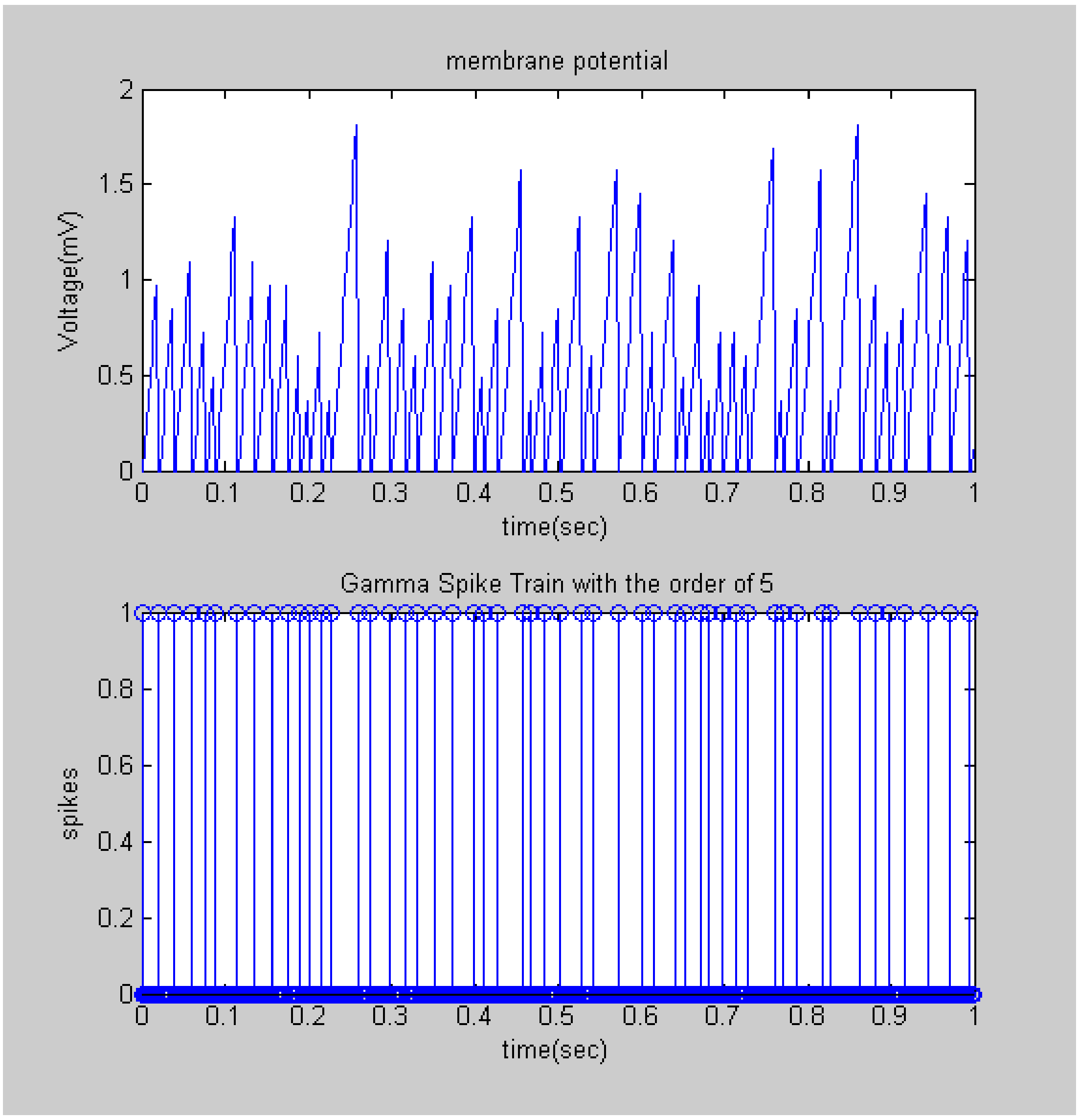


Figure3. gamma spike train from simulation

For the above 3 types of data, H1 neuron response, poisson spike train, and gamma spike train, the ISI histogram, Cv, and Fanor factor were processed and compared to show the similarity among them(refer to 'distribution.m' for the detail).

## Results

From the H1 neuron response, the following ISI distribution and spike count distribution were obtained. The total spike number in the response data set was 53601/600000, and total spike count number was 3000 in the observation window T =200 msec. The mean ISI was 22.385 msec, and Cv_x = 2.0086. The mean spike count was 17.867 spikes/T, and the Fanor factor of the spike count was 5.873.

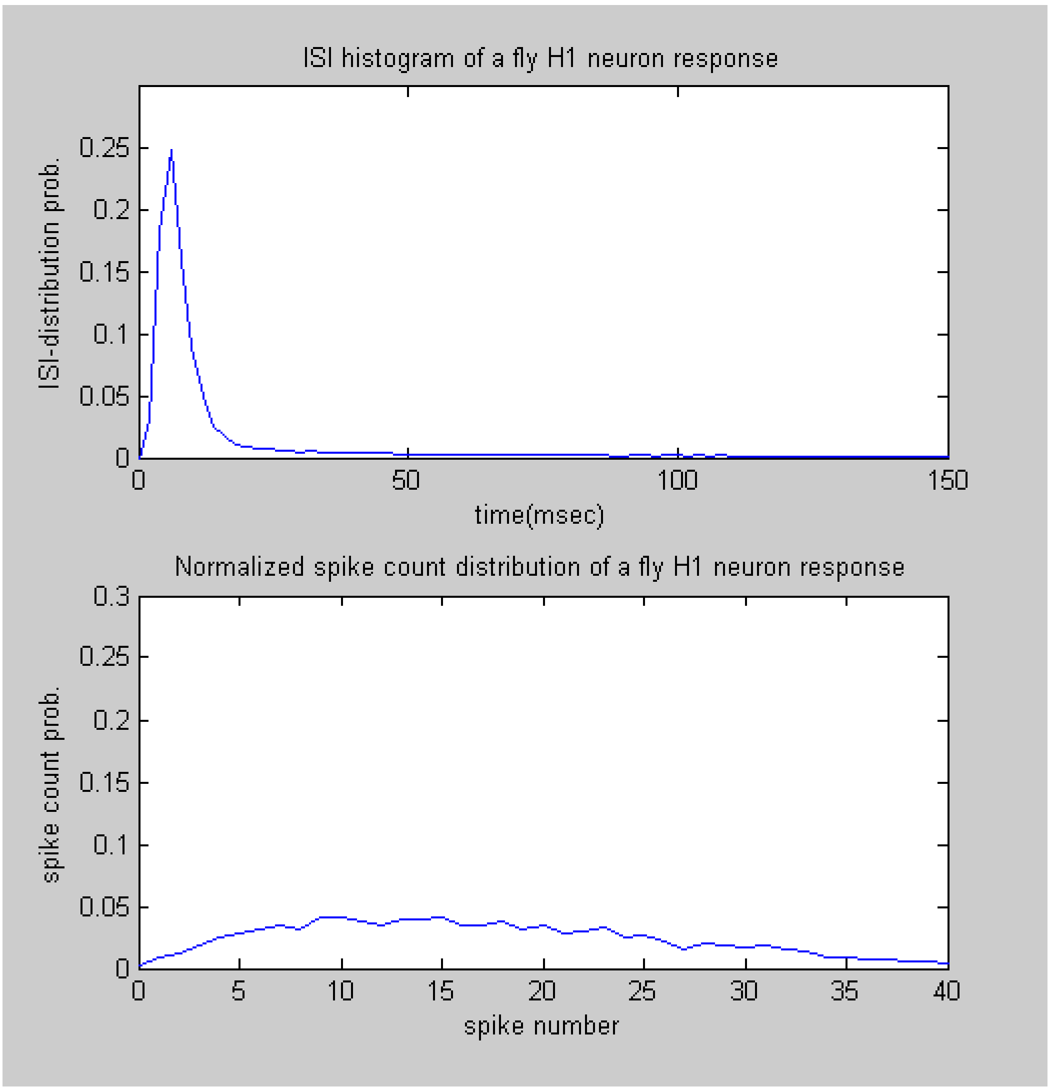


Figure4. ISI distribution and spike count distribution of H1 neuron response

From the poisson spike train of 60sec duration, the following ISI distribution and spike count distribution were obtained. The total spike number in the response data set was 2738/30000, and total spike count number was 150 in the observation window T =200 msec. The mean ISI was 21.908 msec, and Cv_x = 0.91301. The mean spike count was 18.2533 spikes/T, and the Fanor factor of the spike count was 0.77223.

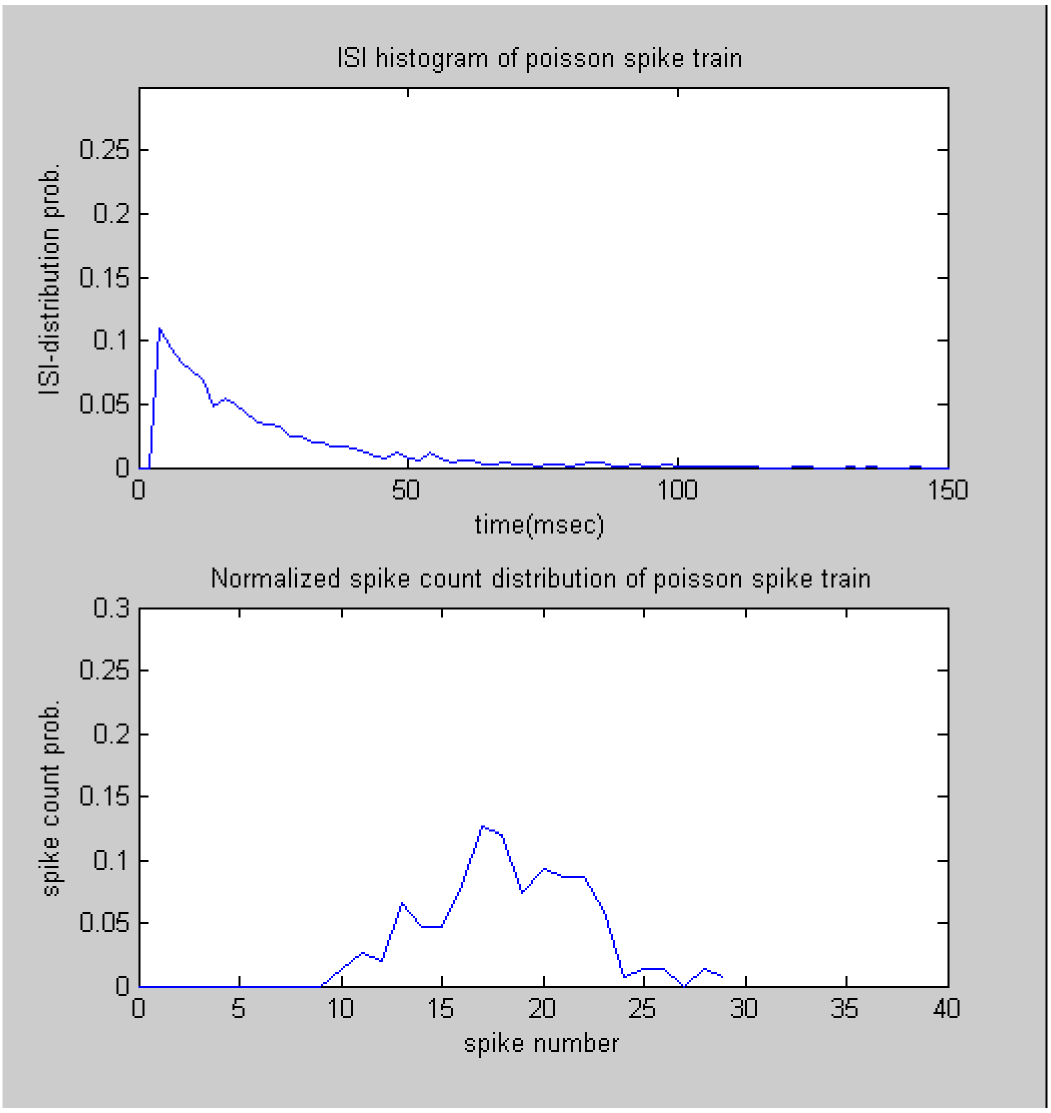


Figure5. ISI distribution and spike count distribution of poisson spike train

From the gamma spike train of 60sec duration, the following ISI distribution and spike count distribution were obtained. The total spike number in the response data set was 3085/30000, and total spike count number was 150 in the observation window T =200 msec. The mean ISI was 19.436 msec, and Cv_x = 0.37472. The mean spike count was 20.5667 spikes/T, and the Fanor factor of the spike count was 0.1429.

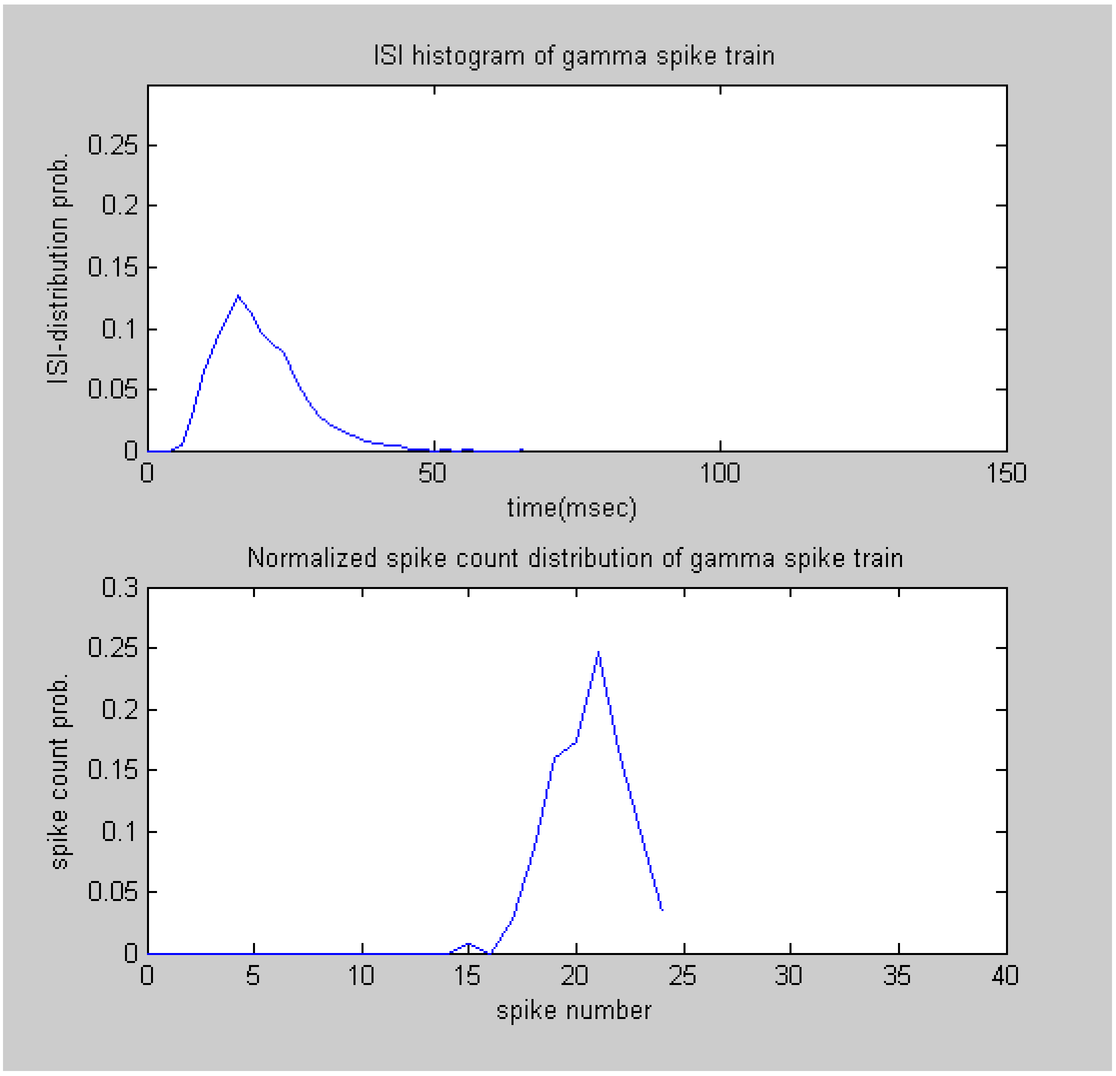


Figure6. ISI distribution and spike count distribution of gamma spike train

## Discussion

From the ISI distribution of H1 response(Figure4), we can observe the dispersive ISI by the existence of large ISI and bursting ISI by the large number of small ISI at the same time.

The poisson spike ISI distribution better matched the dispersion of H1 response ISI distribution than the gamma spike ISI, and this is apparent due to the large value of Cv and Fanor factor of poisson train than those of gamma train. Gamma train seemed to better model the collective components of short ISI of H1 response than poisson train based on the comparison of the ISI distributions. Also, poisson and gamma train include the refractory period consideration, they can model the behavior of very short ISI in the distribution.

However, the similarity of ISI distribution among the above 3 results could be only found in certain degree, and neither of both spike train could model the H1 response spike correctly. H1 response gave more overall variance than the other two models, and high values of Cv and Fanor factor reflect this point.

There can be many sources which are responsible for this mismatch. Among them, the feature of stimulus can be one source. In other word, the white-noise stimulus was used for the H1 neuron, so the randomness included in the stimulus may increase the variability in the spikes. This can be the situation that random input and random threshold work at the same time and make the spike timing ‘doubly stochastic’ process. The Neyman type A distribution caused by ‘doubly stochastic process’ actually show the larger value of Cv and Fanor factor, and may give better description of H1 neuron spike response.

The other source of the more variability can be the ion channels in real neuron. Here, since the simple integrate-fire neuron model was used, this might over-simplify the real situation. If the ion channel status is considered by the gating variables of analytic method or the discrete states of stochastic method, the variability of H1 response spike may be better modeled and described.

On the other hand, the model parameters of the neuron model such as membrane resistance and capacitance need to be chosen carefully. Membrane resistance and capacitance determine the time constant of ODE, and they dramatically affect the inter-spike interval. Too short time constants caused many number of short ISI and vice versa. Also, enough large value of steady state membrane potential was crucial to get the right spike train.

## References

Dayan, P. & Abbott, L.F. (2001) *Theoretical Neuroscience*, Ch1&Ch5. Cambridge, MA: MIT Press.

Gabbiani, F, & Koch, C (1998) Principles of spike train analysis. In C Koch, & Segev, eds., *Methods of Neuronal Modeling,* 313-360. Cambridge, MA: MIT Press.

Rieke, FM, Warland, D, de Ruyter van Steveninck, R, & Bialek, W (1997) *Spikes: Exploring the Neural Code*. Cambridge, MA: MIT Press.

Stark, H. & Woods, J.W. (2002) *Probability and Random Processes with applications to Signal Processing,* Ch.2. Upper Saddle River, NJ: Prentice Hall