Spike Train Analysis & The Jitter Method

Simon Locke & Ernst Niebur

August 2022

Abstract

In this note, we present a tutorial on basic analyses of spike train data including an introduction to point processes, cross-correlation and the jitter method. The tutorial will include additional material on generating Monte Carlo data and hypothesis testing. There will be an opportunity for students to work with synthetic data and recorded data from Macaque motor cortex.

For Students: Students, in this note you will learn the basics of spike train analysis and implement a rigorous statistical method for detecting the presence of synchrony in neural data. These tasks will involve learning a couple basic algorithms that you will implement in Matlab. These exercises may require the use of Matlab functions and techniques that you are not familiar with. If this is the case, we encourage you to review the documentation for each function or technique and to review the Matlab Get Started Primer to make sure everything is fully understood. If you have any questions, please do not hesitate to ask!

Introduction:

An action potential occurs when the membrane potential of a specific cell location rapidly rises and falls (figure 1). For simplifying purposes, each spike may be considered to occur at a single point in time and thus a spike train is a sequence of times at which a neuron spikes. The total duration of a recorded spike train can range from less than a second to many minutes or even the entire length of the recording session. Spike train analysis constitutes a major area of research in neuroscience as spikes are considered to be the primary mode of information transmission across the brain and nervous system.

Spikes can carry information about internal signals of the body, external signals from the environment and signals indicating future actions. The study of how these signals are transformed into neural activity patterns that the brain can understand is known as neural coding. Deciphering the neural code is an extremely complicated task because of the sheer complexity and size of the brain, but also the complexity of the environment and our internal physiological system. Deciphering the neural code will likely yield major insights into the treatment of all diseases and it will require substantial progress in almost all fields related to neuroscience, such as biology, chemistry, physics, applied mathematics, statistics, computer science etc...

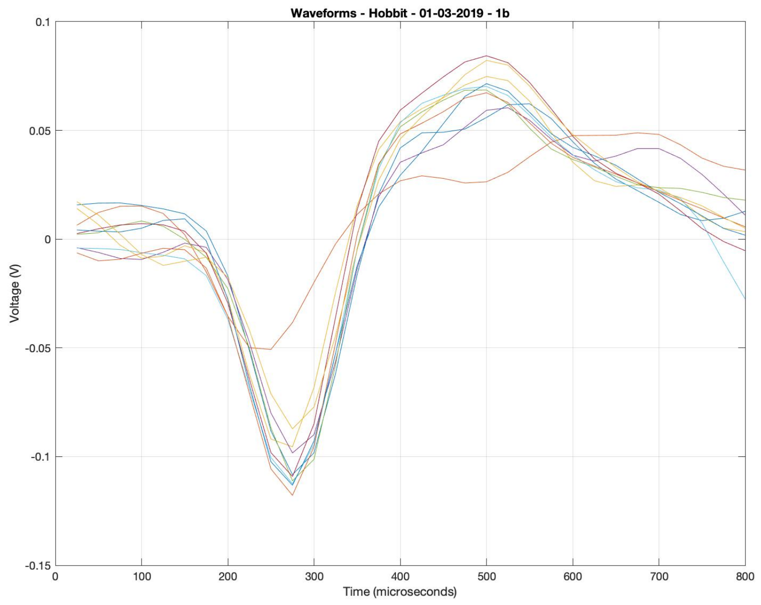

Figure 1. A few action potential traces recorded from a neuron in the Macaque motor cortex

There are many different types of neural codes that have been reported in the literature, but the main two we will focus on are rate coding and temporal coding. The rate coding perspective assumes that information about some signal (internal or external) is contained in the rate at which a neuron fires. The firing rate is defined in terms of the number of spikes that occur over a time interval of length Δt, such as $\Delta t = 300\text{ms}$,

$$\text{Firing Rate} = \frac{\text{number of spikes}}{\Delta t}$$

Firing rate codes have been extremely popular and useful for decades, but they often represent simplifications of the neural code. On the other hand, temporal coding is less explored, and works under the assumption that precise timing of spikes and interspike intervals can carry important information. One type of temporal code is based on synchrony, which is a phenomenon occurring when two neurons fire at the same time or when their fire is consistently offset by some small Δt. Synchrony between neurons has been found in numerous brain areas, though it's functional role has yet to be completely elucidated. In this tutorial, we on how to measure synchrony between two neurons. The first step is to determine mathematically how we will represent our spike trains.

Modeling Spike Trains with Point Processes:

The most common framework for modeling spike trains is the theory of point processes. Suppose we observe k spikes during a particular trial in an experiment and let $s_1, s_2, ..., s_k$ denote the spike times where s_0 denotes the beginning of the trial. Thus $x_i = s_i - s_{i-1}$ is the ith interspike interval for $i = 1, ..., k$. Given a set of positive interspike interval values $x_1, x_2, ..., x_n$ the spike train can be described by a sum of spike times $s_j = \sum_{i=1}^{j} x_i$ where s_j is the jth spike time for $j = 1, ..., k$ (figure 2). Spike trains recorded with an accuracy of at least 1ms can be converted to a sequence of 0s and 1s, where a 1 denotes the occurence of a spike and 0 the absence of one. This procedure is called binning and we will use it in this tutorial.

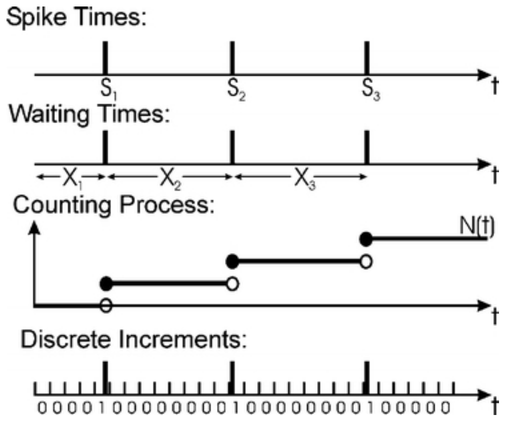

Figure 2. Mathematical Descriptions of a Spike Train (Eden, 2015).

Another way of representing both spike trains is in terms of spike counts. Let $N(t)$ denote the total number of spikes occurring for all $t' \leq t$. Thus, for $s < t$, the spike count between s and t is $N(t) - N(s)$. The expected number of spikes per unit time is called the intensity or rate of the point process. If the point process is time-invariant, it is called stationary or homogeneous, and the intensity is constant, say $\lambda \geq 0$. Otherwise, the point process is non-stationary or inhomogeneous, and the intensity can be a function of time, say $\lambda(t)$ or other known or unknown variables. The intensity can be conceptualized as the instantaneous probability of observing a spike at time t, written as

$$P(\text{spike in } (t, t + dt] = \lambda(t)dt$$

where dt denotes an infinitesimal length of time. Note the similarities to previous definition of the firing rate of a neuron.

The simplest of all point processes is the Poisson process. **It has the following properties, (i) For every pair of time values s and t, with $s < t$, the random variable $N(t) - N(s)$ is Poisson distributed and (ii) For all non-overlapping time intervals the corresponding spike count random variables are independent**. Poisson processes also have the memoryless property, thus their intensity does not depend on the history, H_t, of the process, but is the same as the unconditional intensity: $\lambda(t|H_t) = \lambda(t)$. They cannot exhibit refractory periods, bursting, or any other spike history effects that do not come from the intensity. Homogeneous Poisson processes are time-invariant in the sense that the probability distribution of a count $N(t) - N(s)$ depends only on the length of the time interval $t - s$. Its intensity is constant, say, $\lambda \geq 0$, and its entire distribution is summarized by this single parameter λ. Another key property of the Poisson process that we will make use of later is that **the spikes located in the time interval $(t, t + dt]$ are uniformly distributed within the interval**. Thus there is no opportunity for precise timing of spikes, since their distribution is completely random. While there are deviations from Poisson statistics in real neural data, Poisson processes play a prominent role in theoretical neuroscience and in the statistical analysis of spiking data.

To summarize, we give a procedure for simulating the Poisson process (Cardanobile & Rotter, 2010):

Poisson process:

(1) Draw a random number N_T distributed according to the Poisson distribution with mean λT
(2) For $k = 1, ..., N_T$, draw an independent spike time uniformly distributed on $[0,\ T]$.

Note, the choice of T is up to the user.

Exercise: Open the file **Generating_synthetic_data.m**. In this file, implement the previous algorithm for a homogeneous Poisson process in Matlab and use it to generate 100ms long synthetic spike trains from two neurons for 100 trials. One neuron with a firing rate of 10 Hz and the second with a firing rate of 5 Hz. For generating Poisson and uniformly distributed random numbers, use the Matlab functions **unifrnd** and **poissrnd**. Once you are finished, check the average firing rate across your synthetic spike train dataset to make sure it aligns with the rate parameter you specified in your Poisson process.

Exercise: Open **raster_plot.m**. This program will create raster plots of your synthetic spikes trains. See the file for further instructions.

Measures of Synchrony: the Cross-Correlation Function (CCF):

Now that we have a clear mathematical description of spike trains and you have some experience with Poisson processes, we need to decide how to measure synchrony. Measures of neuronal synchrony are estimators of the synchrony between two (or sometimes more) time series of brain activity which yield low values for independent time series and high values for correlated time series. There are many different techniques used to measure neuronal synchrony, and they can be divided into two categories; **linear** and **nonlinear** measures. The simplest and most commonly used measure of synchrony is linear **cross-correlation**, which is the focus of the remainder of this tutorial. Cross-correlation measures the similarity between a vector x and shifted (lagged) copies of a vector y as a function of the lag values, which are set beforehand. In our case, the vectors are spike trains from each neuron. The cross-correlation function of two spike trains S_j and S_k at lag τ, from neurons j and k recorded on the ith trial is given by

$$S_j^i \odot S_k^i(\tau) = \sum_{\mu=t_0}^{t_{\text{end}}} S_j^i(\mu+\tau) S_k^i(\mu)$$

where S_k is the reference spike train, S_j is the shifted spike train, and $\odot$ is the cross-correlation operator. One detail that needs to be considered when dealing with cross-correlation is how to deal with edge effects. Edge effects refer to the fact that, for a fixed and finite observation window, the amount of the observation window that can contain a synchronous event varies systematically with the lag of the CCF. For example, if you calculate a CCF for two spike trains that are each 100ms long, you will have less data at lag = 50ms than at lag = 0. Eventually, one will completely run out of data. This creates the annoying feature that a raw CCH will typically taper to zero for very long lags, not because of any drop in correlation, but rather because of a lack of data. There are methods for adjusting these effects, but we will not go into them as the statistical method we will learn will correct for them entirely.

Exercise: First, open the file titled **Cross_correlation_spiketrains.m**. The MATLAB function **Xcorr** computes the cross-correlation function between two discrete time series. Using Xcorr, calculate the cross-correlation function between spike trains from each Poisson neuron for each trial and then sum the CCFs across trials. This will require you to bin the spike trains, using 1ms bins. Once this is finished, using the Matlab function **plot**, visualize your cross-correlation function with the lags on the x-axis. This plot is called a cross-correlation histogram (CCH). Write a brief description of your results. Note: you will be required to set the limits for the lag range of the CCF so be sure to look over the documentation for Xcorr. A maximum lag of 100 is a good choice.

Statistical Estimation of Synchrony: the Jitter Method & Monte Carlo data:

With a method in hand for measuring synchrony, we can now apply it to data, but how will we know whether our results are statistically significant? This endeavor can be surprising subtle. Let's say we want to record from two neurons and compute a CCH to see if there is evidence of synchrony between these cells. One potential issue, is that there are countless events going on elsewhere in the brain that may influence the firing properties of our two neurons. Some examples include slow-temporal gain modulations, metabolic changes and adaptation over trials, internally generated slow-wave oscillations, or drifts in the recording signal. These events could influence observed spike timing behavior and induce spurious correlations, which we want to avoid. How might one account for these broader temporal influences on spike timing?

The answer, is to separate the fine temporal influences on spike timing from the many coarse temporal influences on spike timing (Amarasingham et al., 2012). This is achieved by choosing a model that will allow for coarse temporal influences on spike timing, which we will then use for hypothesis testing. For the hypothesis testing, we begin with a model, the null hypothesis or null model, that allows only coarse temporal spiking and ask if the observed data are consistent with the model. A negative conclusion, i.e., a rejection of the null hypothesis, is interpreted as evidence in favor of fine temporal spiking dynamics that create excess observed synchrony, though this is a simplification.

For the model, we need to rely on the theory of point processes discussed earlier. Since firing rates are known to vary randomly during and across trials, we will start with an inhomogenous Poisson process, which if you recall, is a Poisson process with the a time-varying intensity $\lambda(t)$. Suppose we take two spike trains, A and B, and assume that the intensities $\lambda^A(t)$ and $\lambda^B(t)$ are piecewise constant over the disjoint Δ-length intervals $[(n-1)\Delta, n\Delta]$ for integers $n \in \mathbb{Z}$. This model is a sequence of small independent, homogeneous Poisson process models, one for each Δ-length interval. The piecewise constant assumption is made to approximate an intensity function that is slowly varying on time scales larger than Δ. If we let $\Delta = 20$ms, this means that we allow our time-varying firing rates to vary on timescales larger than 20ms which will mimic the coarse, temporal influences on spiking. Now we apply the **jitter method**! **Jittering refers to the procedure of uniformly re-distributing the spikes in each Δ-length window**. Jittering the spikes in each $\Delta-$length interval preserves features of the spike trains on timescales larger than Δ (i.e firing rate) while destroying correlations on timescales smaller than Δ. This is called the Δ-uniform model and it will serve as our null model. For more information see (Harrison et al., 2013).

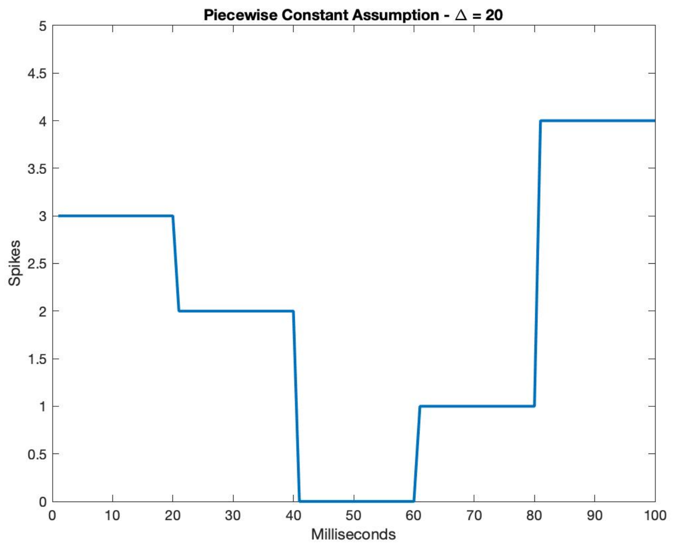

Figure 3. A piecewise constant function to illustrate a slowly varying firing rate. Each constant represents the number of spikes in the Δ-length bin.

Since we have specified our null model, it is straightforward to describe the null variability in the CCH. This requires a brief discussion of **Monte Carlo simulation**. Monte Carlo methods are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. They are very useful for generating draws from a probability distribution, which is exactly how we will use them. To proceed, we generate a sample of many (usually 1000) independent Monte Carlo pseudo-datasets from the null distribution over pairs of spike trains and compute a CCH for each one.

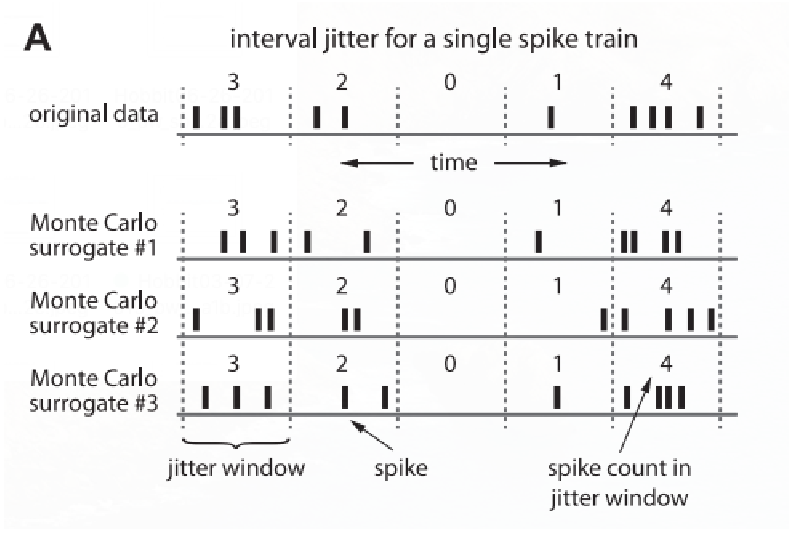

Figure 4. A few examples of jittering across Monte Carlo surrogate (Amarasingham et al., 2012)

This collection of pseudo-CCHs can be used to approximate the typical variability under the null distribution, which gives us an idea of how much synchrony is present based on coarse, temporal influences on the firing rates alone. Now that we have explained everything, let's take a look at our statistical modeling framework:

(1) Pick a model that allows for coarse temporal influences on spike timing

(2) Once we choose a model, use it generate Monte Carlo data

(3) Quantify how well the observed CCH reflects the typical behavior with a p-value

(4) Use graphical displays showing bands of typical fluctuation around the observed CCH

Exercise: Load in the file titled **Jitter_function.m**. This function will implement the jittering algorithm in Matlab. See the file for further instructions.

Estimating Sychrony from Macaque Motor Cortex:

It's time to work with real data! In each file starting with the name Hobbit, you will find a few cell array from neurons recorded from the Macaque motor cortex during a decision-making experiment, courtesy of the Stuphorn lab. These data consist of spike trains recorded during the 300ms period before the monkey makes a task-related decision, so there may be some interesting information in the neural data.

Exercise: Load in the file **Hobbit01-05-2019(01)_spiketraindata_premotor300_1a1b.mat** and compute a raw CCH (with no jittering), using all trials. Do the same for each neuron pair (each .mat file). Describe the results for each CCH. Note: When you download the .mat files, open them and inspect the contents. What you will find is a cell array containing single trial spike train data from each neuron, one in the first column and the second in the second column, and for each trial.

Once you have computed this CCH on our monkey data, you will see some very interesting results. But how do we know these results represent a statistically significant result, and not a spurious correlation induced by similarities in the firing rates of the two neurons? Well, you can use the jitter method and see if the correlations you see are a result of correlations on a broader time scale.

Exercise: Using the provided MATLAB script called **Jitter_CCH_with_acceptance_bands.m**, plug in the cross-correlation data from the previous exercise and run the file. But, wait! Before you do this, you will need to insert a couple lines of code in order for the file to run smoothly. See the file for further instructions.

As you can now see, the file has created a jitter-corrected CCH with two sets of light and dark gray bands. These bands represent the threshold that the original CCH needs to cross in order for a result to be statistically significant. The dark gray bands represent a single hypothesis test for each individual lag. If the CCH crosses this band at any lag, it means that the CCH calculated using the original data has a value that exceeds 95% of the previously generated Monte Carlo data corresponding to a p-value of < 0.05. But the story is not finished. Since we ran a hypothesis test for each lag, it follows that we'll have over 200 hypothesis tests, and some will be significant by chance. Thus we need to correct for multiple hypothesis tests. The procedure we use for this correction is complicated and beyond the scope of this tutorial, but the result the outer set of light gray bands. Note, these bands are more extreme and thus the CCH values will have to be larger in order to cross the significance threshold. But, as you can see, many CCHs from our monkey data have lags that cross the light gray bands easily. Thus, we have statistically significant results! This means that there is excess synchrony between two neurons that cannot be explained by correlations that occur on timescales larger than delta, therefore they must be a result of precise timing.

Congratulations! You have finished this tutorial and now know a rigorous statistical method for estimating the amount of synchrony between two neurons. Be sure to use it during your PhD should you ever encounter spiking data from simultaneously recorded neurons!

Epilogue:

If you have finished with time to spare, we give a couple open-ended exercises.

Exercise: Make more synthetic data with a variety of firing rates. One suggestion would be to change the firing rate across trials, simulating an inhomogeneous Poisson process. Afterwards, check the CCH (w acceptance bands). Are there any significant results? If not, explain why.

Exercise: Make more synthetic data, but this time, with the goal of replicating the sharp peaks you observe in the real data. What might be the simplest way to achieve this results?