ANALYZING METHODS TO DETERMINE PAIRWISE CORRELATIONS BETWEEN NEURONS

LUNA BOZEMAN AND ADRIANA MORALES

Abstract. How can sampling methods affect our results? In predicting pairwise correlations between neurons, the accuracy of the sampling method is crucial, as many studies suggest that understanding the relationship between neurons allows us to better understand and interpret the information and computations performed within our brains. In their recent paper, Okun *et al.* presented a new method for predicting these pairwise correlations, which requires the use of three simple parameters from a raster. In this paper, we will discuss and analyze Okun *et al.*'s sampling method, which allows for an error on one of the parameters, and investigate the impact this error has on the pairwise correlations.

1. Introduction

Neurons in your brain fire signals known as spikes to communicate with each through an electrochemical process. Thus, for any given time interval, each neuron has a corresponding spike train, which is a sequence of spikes over time. Okun *et al.* represent spike trains by a binary sequence, where 1 represents a spike and 0 represents no spike. In an attempt to explain relationships between neurons based on these spike trains, Okun *et al.* discussed these complex individual neuronal activities and how they might be coordinated within our brains [4]. They found that neighboring neurons were correlated based on the firings of the overall population, and also found that this provided a compact summary of the population activity, which can be a complicated phenomenon.

Here, we focus specifically on the methodology used to predict these correlations between each pair of neurons, which we refer to as the pairwise correlations. These neurons come from a given *raster*, which is a sample of spike trains for the observed neurons. The following are the questions that we will address throughout this paper and will provide insight into: Okun *et al.* allow for a small error when computing pairwise correlations, but would we be able to obtain more consistent and accurate results by not allowing for this error? Is there a biologically plausible matrix where their method creates a significant difference in correlation values?

In answering these questions, we utilized MATLAB code provided by Dr. Okun. [1] We modified Dr. Okun's code to no longer allow for the mentioned error, and also created a program that would generate multiple sample matrices of various sizes from a provided raster in order to obtain our results. We found that especially for smaller matrices, the error can have a significant impact on the pairwise correlations, but that this error no longer becomes a concern as we continue to increase the length of the spike trains for each neuron.

In the following section, we provide a more in-depth look at Okun *et al.*'s method, known as the raster marginals model with coupling terms, and describe our notation. Section 3 will

Date: July 21, 2016.

[1]Dr. Okun's code may be found at `https://sites.google.com/site/michaelokunsite/`

discuss the modified code and the new program that we utilized. Section 4 will discuss the outputs from the previous program and the results from our research. Finally, Section 5 will provide possible future directions.

2. Background

2.1. **Raster Marginals Model with Coupling Terms.** In order to predict pairwise correlation values, Okun *et al.* utilized the raster marginals model with coupling terms. First, they used a raster file which contained recordings of the spike trains of several neurons in the form of a 0-1 matrix, where each row represents a neuron's spike train and each column represents the existence or nonexistence of spikes for every $20ms$ interval. They concluded that only three parameters from such a matrix were needed in order to predict a large portion of the pairwise correlations: the row sum **s**, the column sum **c**, and the inner product **d** of each row with the column sum. The **s** constraint represents the number of spikes of a given neuron over the observed time, **c** the number of neurons that spiked at a given time, and **d** the spike trigger population rate for the corresponding neuron. In particular, this last parameter provides insight on the leader-follower relationship between each pair of neurons [4].

Example 2.1. Here, we provide a small example of a raster plot and its corresponding parameters.

Neurons	**Raster Plot**								**s**	**d**
1	0	1	0	1	0	0	1	1	4	4
2	1	0	1	0	1	1	0	0	4	8
3	1	0	1	0	1	1	0	0	4	8
c	2	1	2	1	2	2	1	1		

Focusing on neuron 1, first note that its spike train is 01010011, as seen in the first row. This is interpreted to mean that neuron 1 did not fire at the first time interval, fired for the second time interval, did not fire for the third time interval, and so on. As you can see, $\mathbf{s}(1)$ is equal to 4 as neuron 1 fired four times. Now, observing the first time interval, we see that **c** is equal to 2 as only neuron 2 and 3 fired during this time interval. For the inner product parameter for neuron 1, we compute $\begin{bmatrix}0 & 1 & 0 & 1 & 0 & 0 & 1 & 1\end{bmatrix} \cdot \begin{bmatrix}2 & 1 & 2 & 1 & 2 & 2 & 1 & 1\end{bmatrix}$ to obtain the value of $\mathbf{d}(1)$, which in this case is 4. The same process is applied to the other neurons and time intervals to obtain the parameter values.

2.1.1. *Ryser's Algorithm.* Utilizing the raster file, they first generated a matrix that satisfied the **s** and **c** constraints using Ryser's algorithm, a specific method used to create 0-1 matrices with constraints on row and column sums. Let us assume that we have a matrix of size $n \times m$.

Algorithm 1: Ryser's Algorithm

Input : row sum **s** and column sum **c**

Output: matrix with desired **s*** and **c***

1 Rearrange the **s** and **c** in a non-increasing manner. We will refer to these new vectors as $\mathbf{s}^*$ and $\mathbf{c}^*$

2 Create a perfectly nested $n \times m$ matrix with row sum **s*** such that the 1's in every row are in the initial positions.

3 Shift 1's to the right in order to obtain the desired column sum **c*** in the following manner:

a) Shift 1's beginning from the rows with the largest sums until we reach the desired $\mathbf{c}^*(m)$. Note that when there are multiple rows with the largest sum, we will choose the lowest row first.

b) Continue this process for $\mathbf{c}^*(m-1), \ldots, \mathbf{c}^*(1)$ to finish with a matrix with desired **s*** and **c***

Note that **s** and **c** are not identical to $\mathbf{s}^*$ and $\mathbf{c}^*$, respectively, but they are equivalent up to permutation.

They then put this new matrix into canonical form. Note that this means that we revert back to our original permutation for the row sums. Next, they performed a spike change across neurons [3], where, as the name implies, spikes of neurons are exchanged. This is done in a way so that $\mathbf{s}^*$ and $\mathbf{c}^*$ are preserved. Figure 1 is a representation of this exchange, where the solid rectangles represent spikes and each line represents a different neuron.

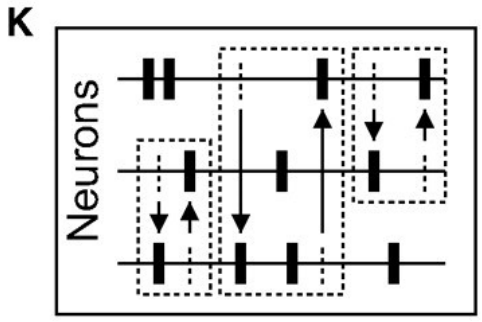

FIGURE 1. A representation of spike exchange across neurons [3]

Example 2.2. Here, we utilize the above Example 2.1 to demonstrate Ryser's algorithm, and then the spike exchange across neurons. First, we will rewrite $\mathbf{c} = \begin{bmatrix} 2 & 1 & 2 & 1 & 2 & 2 & 1 & 1 \end{bmatrix}$ so that the terms are in non-increasing order. Thus, we have $\mathbf{c}^* = \begin{bmatrix} 2 & 2 & 2 & 2 & 1 & 1 & 1 & 1 \end{bmatrix}$. Note that we would do the same for **s** as well, but in our case, **s** is already organized in an non-increasing order, thus we will call it $\mathbf{s}^*$. We utilize Ryser's Algorithm [1] to obtain the

following matrix:

$$\left[\begin{array}{cccccccc\|c} 1&1&1&1&0&0&0&0&4\\ 1&1&1&1&0&0&0&0&4\\ 1&1&1&1&0&0&0&0&4 \end{array}\right] \rightarrow \left[\begin{array}{cccccccc\|c} 1&1&1&1&0&0&0&0&0&4\\ 1&1&1&1&0&0&0&0&0&4\\ 1&1&1&0&0&0&0&0&1&3 \end{array}\right]$$

$$\rightarrow \left[\begin{array}{cccccc|cc\|c} 1&1&1&1&0&0&0&0&4\\ 1&1&1&0&0&0&1&0&3\\ 1&1&1&0&0&0&0&1&3 \end{array}\right]$$

$$\rightarrow \left[\begin{array}{ccccc|ccc\|c} 1&1&1&0&0&1&0&0&3\\ 1&1&1&0&0&0&1&0&3\\ 1&1&1&0&0&0&0&1&3 \end{array}\right]$$

$$\rightarrow \left[\begin{array}{cccc|cccc\|c} 1&1&1&0&0&1&0&0&3\\ 1&1&1&0&0&0&1&0&3\\ 1&1&0&0&1&0&0&1&2 \end{array}\right]$$

$$\rightarrow \left[\begin{array}{ccc|ccccc\|c} 1&1&0&1&0&1&0&0&2\\ 1&1&0&1&0&0&1&0&2\\ 1&1&0&0&1&0&0&1&2 \end{array}\right]$$

$$\rightarrow \left[\begin{array}{cc|cccccc\|c} 1&1&0&1&0&1&0&0&2\\ 1&0&1&1&0&0&1&0&1\\ 1&0&1&0&1&0&0&1&1 \end{array}\right]$$

$$\rightarrow \left[\begin{array}{c|ccccccc\|c} 1&1&0&1&0&1&0&0&1\\ 1&0&1&1&0&0&1&0&1\\ 0&1&1&0&1&0&0&1&0 \end{array}\right]$$

$$\rightarrow \left[\begin{array}{\|ccccccc\|c} 1&1&0&1&0&1&0&0&0\\ 1&0&1&1&0&0&1&0&0\\ 0&1&1&0&1&0&0&1&0 \end{array}\right]$$

Note that this new matrix satisfies our $\mathbf{s}^*$ and $\mathbf{c}^*$ constraints. Now we must put our matrix into canonical form. Canonical form in this context means reverting back to our original permutation for $\mathbf{s}$ so that we can keep track of the order of neurons. Note that $\mathbf{c}^*$ will stay as it is. By performing a random spike exchange across neurons [3], here is one possible matrix that we may obtain:

Neurons	**New Raster Plot**								$\mathbf{s}^*$	$\mathbf{d}^*$
1	1	1	1	1	0	0	0	0	4	8
2	0	0	0	0	1	1	1	1	4	4
3	1	1	1	1	0	0	0	0	4	8
$\mathbf{c}^*$	2	2	2	2	1	1	1	1		

2.1.2. *Population Coupling Algorithm.* After using Ryser's algorithm, Okun *et al.* performed sequential steps in order to satisfy the last constraint of inner product $\mathbf{d}$, up to an error of n for each entry of $\mathbf{d}$, where n represents the number of neurons observed in a given raster.

Algorithm 2: Population Coupling Algorithm

Input : An $n \times m$ matrix that satisfies the $\mathbf{s}^*$ and $\mathbf{c}^*$ constraints
Output: A new $n \times m$ matrix that satisfies the $\mathbf{s}^*$, $\mathbf{c}^*$, and $\mathbf{d}^*$ constraints.

1 Take a smaller 2×2 matrix between rows that have inner products that are over and under the allowed tolerance.
2 For each of these rows, columns were found so that each row and column of this new 2×2 matrix contained 0 and 1.
3 Exchange the 0's and 1's in order to maintain the satisfaction of **s*** and **c***, while improving the satisfaction of $\mathbf{d}$.
4 Continue this process until all values of **d*** are within the allowed tolerance, creating a new matrix.

Example 2.3. Here we return to our previous Examples 2.1 and 2.2. Note that our $\mathbf{s}^*$ and $\mathbf{c}^*$ constraints are satisfied, but $\mathbf{d}$ is not. Observing specifically the first and second neurons, $\mathbf{d}(1) = 4$ but $\mathbf{d}^*(1) = 8$, and $\mathbf{d}(2) = 8$ but $\mathbf{d}^*(2) = 4$. Since our n is 3 for this matrix, the values of $\mathbf{d}^*(1)$ can only be 4 ± 3 and the values of $\mathbf{d}^*(2)$ can only be 8 ± 3. Thus we are not within our tolerance.
We will now perform sequential steps in order to fix our $\mathbf{d}^*$. Since $\mathbf{d}^*(1)$ is over the tolerance and $\mathbf{d}^*(2)$ is under the tolerance, we will perform a 2×2 sub-matrix exchange with neuron 1 and 2. Note that the boxed sub-matrix below has a 0 and a 1 in every row and column.

Neurons	**New Raster Plot**								**s***	**d***	**d**
1	1	1	1	1	0	0	0	0	4	8	4
2	0	0	0	0	1	1	1	1	4	4	8
3	1	1	1	1	0	0	0	0	4	8	8
c*	2	2	2	2	1	1	1	1			

Exchange the 0's and the 1's in the boxed sub-matrix above:

Neurons	**New Raster Plot**								**s***	**d***	**d**
1	1	1	1	0	1	0	0	0	4	7	4
2	0	0	0	1	0	1	1	1	4	5	8
3	1	1	1	1	0	0	0	0	4	8	8
c*	2	2	2	2	1	1	1	1			

Note that every element of $\mathbf{d}^*$ is now within the ± 3 tolerance.

Finally, with the parameters within the specified constraints, they computed the correlation between each pair of neurons from this new matrix using the Pearson correlation.

Definition 2.1 (Pearson Correlation)**.** The *Pearson correlation* is a measure of strength of the linear relationship between two given variables. Possible values of the *Pearson correlation coefficient*, r, range from -1 to 1, where -1 represents a negative association, 0 represents no association, and 1 represents a positive association. For a given sample, we can calculate

this value for variables x and y using the following formula:

$$r = \frac{\sum_{n=1}^{m}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{n=1}^{m}(x_i - \bar{x})^2}\sqrt{\sum_{n=1}^{m}(y_i - \bar{y})^2}} \tag{1}$$

where m represents the number of data points from x and y, and $\bar{x}$ and $\bar{y}$ represent the sample mean for x and y respectively.

Example 2.4. We return to our previous Example 2.3. As an example, we will compute the correlation between neuron 1 and neuron 2. First we will compute the sample mean.

$$\bar{x} = \frac{\mathbf{s}^*(1)}{8} = \frac{1}{2}$$
$$\bar{y} = \frac{\mathbf{s}^*(2)}{8} = \frac{1}{2}$$

Now we can compute the correlation, $r_{1,2}$, using equation 1, to obtain $r_{1,2} = -1$. Note that this is an estimation as we are only using a sample of the neurons' spike trains and this is only one result from the raster marginals model with coupling terms method.

We would like to note that up to the final step of computing the pairwise correlations values between each pair of neurons, the computational work was done via MATLAB. Each run outputs one matrix that satisfies the three parameters mentioned above. The various codes were written and provided by Dr. Okun.

In summary, the raster marginals model with coupling terms first requires a raster plot that contains recordings of the spike trains of several neurons. We extract the parameters $\mathbf{s}$, $\mathbf{c}$, and $\mathbf{d}$, and first generate a matrix satisfying just the modified $\mathbf{s}^*$ and $\mathbf{c}^*$ using Ryser's algorithm, and reorganize the matrix into canonical form. After performing a spike exchange across neurons, they use the population coupling algorithm in order to satisfy the $\mathbf{d}$ constraint up to an error of n. Using this new matrix, correlations between each pair of neurons were calculated to serve as an estimate.

3. Modified Code and Program

The main concern that we had with the raster marginals model with coupling terms method came from the $\pm n$ error allowed on $\mathbf{d}$. Since the other constraints did not allow for such error, we hypothesized that we would obtain more accurate and consistent results if we did not allow this error.

Thus our main goal throughout this research was to determine whether the $\pm n$ error allowed on $\mathbf{d}$ made a difference in our correlation results or not. In order to investigate the impact that this error had on the correlations, we focused on the following two goals:

- Create code that would perform the raster marginals model with coupling terms but without the error on $\mathbf{d}$.
- Create a program that would generate multiple sample matrices of various sizes, both with and without the $\pm n$ on $\mathbf{d}$, and output their corresponding correlations in order to more accurately compare the methods.

3.1. **Raster Marginals Model with Coupling Terms Modified.** We first modified Dr. Okun's MATLAB code and modified it so that it would only output matrices that satisfied all three parameters with no error. With the modified code, we created other small code and implemented them into one large program in order to generate numerous matrices using both the original and modified code and to compare the predicted pairwise correlations.

Algorithm 3: Raster Marginals Model with Coupling Terms Modified

Input : The raster file, the number of columns that you would like to observe, the number of sample matrices you would like from Dr. Okun's original code that allows for the tolerance on $\mathbf{d}$, and the number of sample matrices you would like from our modified code that does not allow for an error on $\mathbf{d}$.

Output: The graphs of correlations and the standard deviations of the correlations obtained from both methods

1 The first sub-function generates a new raster file for you. Using the input for the number of columns that you would like to observe, this sub-function randomly selects columns from the inputted raster file to create a new matrix, with the condition that no row contains all 0's, as this would lead to a nonexistent correlation. We select columns from the original raster file to ensure that the new matrix is biologically plausible. This sub-function allows us to observe smaller matrices to see how size could have an affect on our correlation outcomes.

2 Extract the three parameters from the new raster file and run Dr. Okun's code and our modified version of his code the specified number of times.

3 For every matrix that is outputted from each run of the code, calculate the correlation coefficient matrix using Pearson's correlation. This allows us to compare the various correlations that we obtain from both code.

4 To determine how consistent the correlations are from each code, calculate the sample standard deviations of the correlations for both samples and compute the difference. This proved useful for larger matrices, where we obtained roughly normal distributions.

5 For a visual reference, plot the correlations using red dots for matrices generated from Dr. Okun's code and blue dots for matrices generated from our modified code. In addition, to serve as a reference, we plot the correlations from our new raster file before running it through the raster marginals model with coupling terms, represented by green dots. There will be n graphs, where each graph represents the correlation between neuron i and all of the other neurons. The x-axis represents the neurons and the y-axis represents the correlation.

4. Results

The original raster file provided by Dr. Okun is a $10 \times 170,000$ matrix. Each column represents a $20ms$ time bin, thus the raster is a recording of 10 neurons over roughly 57 minutes.

Here, we will start by generating smaller matrices as our raster file, and work our way up to larger ones to compare our outputs. Recall that these matrices are sub-matrices of Dr. Okun's raster. To be consistent, we will generate 100 samples each using Dr. Okun's code and our modified code, for a total of 200 sample matrices.

4.1. **Example Outputs.** We begin with a 10×30 matrix as our raster. This represents roughly .6 seconds worth of recordings. Here we provide an example of a possible output from our program.

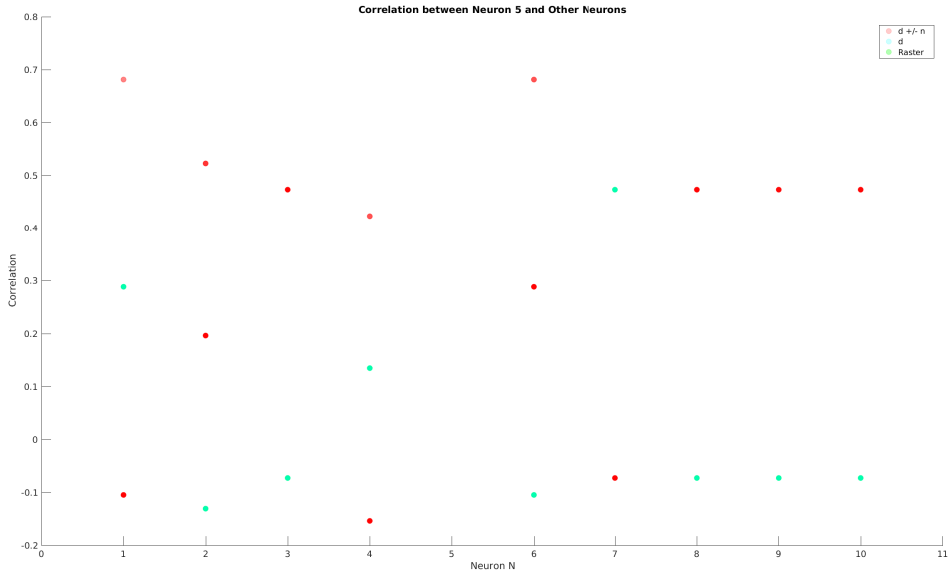

FIGURE 2. An example of a 10×30 matrix's output

With this being a small matrix, it may not be of much biological interest, but it is interesting to see how varied the correlations are. It is worth noting that though the correlations range vastly for certain pairs of neurons, there tend to only be 1 or 2 possibilities for the values, since although we have plotted correlations from all 200 matrices, many have the same correlations. The darkness of the plots above indicate the number of overlaps that occurred in correlations, where darker points indicate more overlaps. We also see that the correlations obtained from our modified code tend to be more consistent with the correlations obtained from the new raster file, seen by the overlap of the blue dots with the green dots. Through this example, we clearly see that the tolerance on $\mathbf{d}$ has an impact on the correlations.

The following is an example of an output from a 10×300 matrix, representing 6 seconds worth of recordings. Overall, we note that the values are within a smaller range, indicating more consistency than the example in figure 2. But as before, we see that the values obtained from the modified code offer more accuracy than the original.

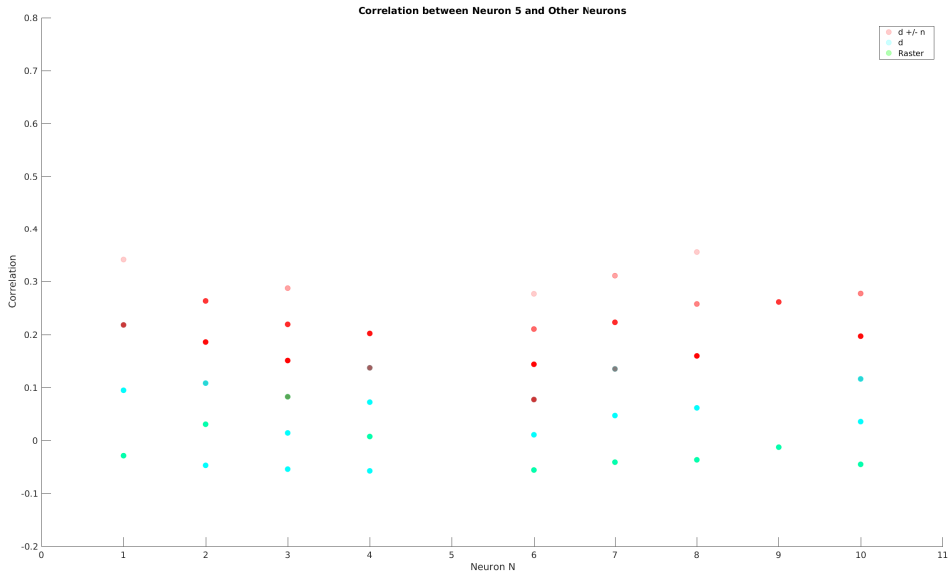

FIGURE 3. An example of a 10×300 matrix's output

As we continue to increase the size of our new raster file, we note a similar occurrence. We also see that the correlations obtained from both codes become more consistent with each other, seen both from the graphs and from the standard deviation values.

Next, we have an example of an output from a 10×3000 matrix. Note the change in scale of the y-axis.

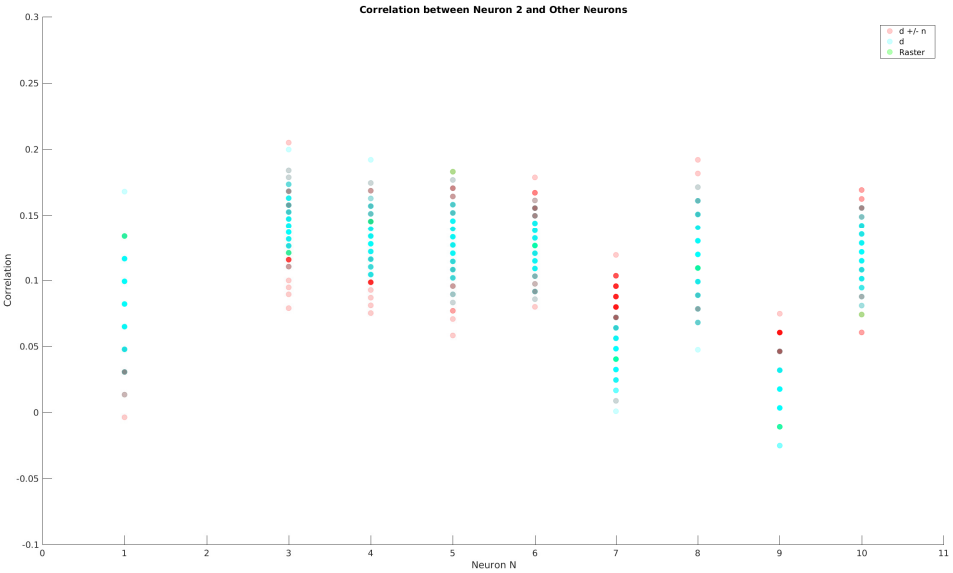

FIGURE 4. An example of a 10×3000 matrix's output

Next, we have an example of an output from a 10×10000 matrix. Again, note the change in scale of the y-axis. We still note a similar occurrence as before.

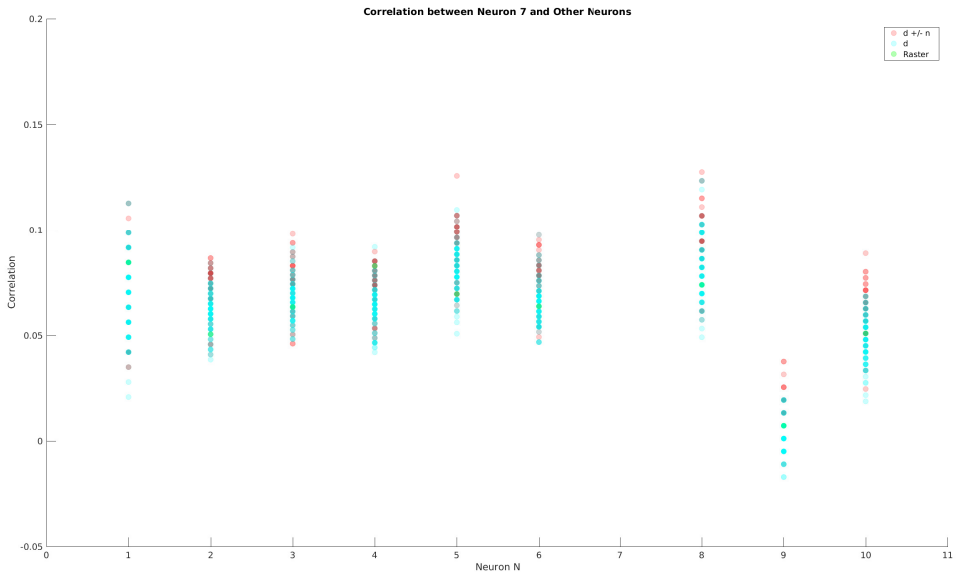

FIGURE 5. An example of a 10×10000 matrix's output

Finally, we have an example of a 10×170000 matrix. In this case, we did not generate a new raster file but instead, simply used the provided raster from Dr. Okun. Now, unlike before, we see that the correlations are consistent regardless of which code they came from. They all overlap within a small range, indicating that the raster marginals model with coupling terms is an appropriate method to determine pairwise correlations for rasters of this size and data.

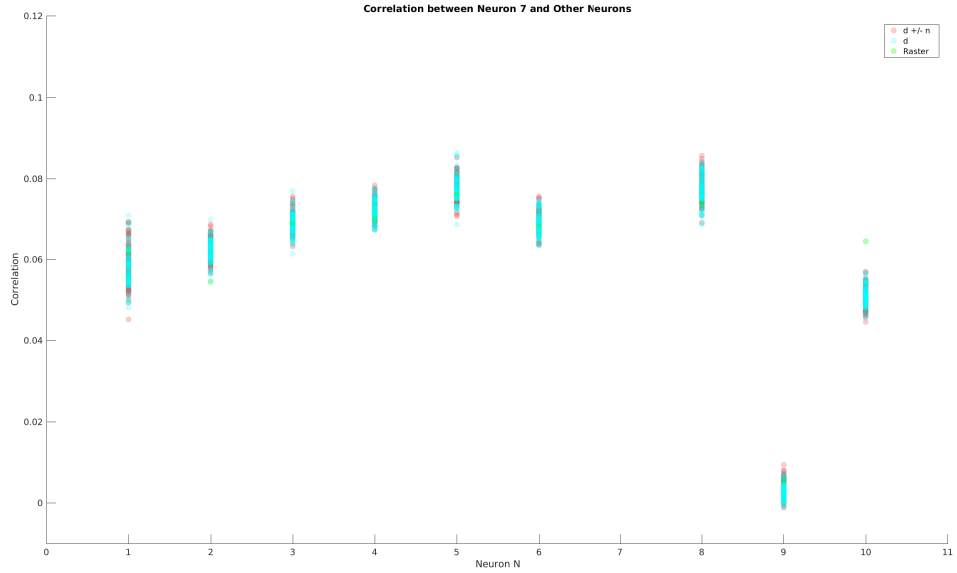

FIGURE 6. An example of a 10×170000 matrix's output

Here is a closer look at the correlation between neuron 7 and neuron 8 to show how the correlations from both code are rather similar. Correlations between other neurons proved to provide similar results as below.

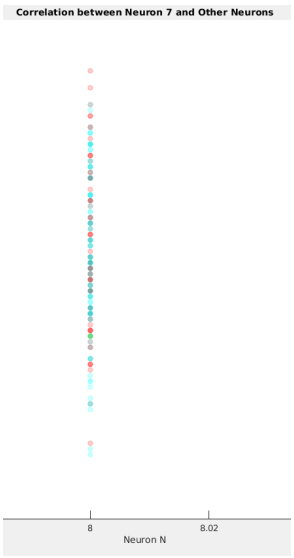

FIGURE 7. A closer look at the correlation between neuron 7 and 8

4.2. **Results and Observations.** From these examples above, which are consistent with our findings from other examples that we obtained, we can first note that the tolerance on **d** matters less as we increase the number of columns of our matrix. We believe that this has to do with both the size of the matrix and the values of **d**. Thus, we see that the raster marginals model with coupling terms, with or without the tolerance, is an appropriate model for sufficiently large matrices that contain similar data to that of the provided raster file.

In addition to this, we can note that the standard deviations for smaller matrices were vastly different between the samples, since the modified code offered smaller standard deviations. But, as we increased the number of columns that we were observing, the difference in the standard deviations became negligible, which was an unexpected finding.

As our results can only be truly applied to rasters that are similar in nature, we believe that our new program can serve as a check to see if the raster marginals model is a good fit for specific rasters or not. By running multiple samples from both Dr. Okun's code and our modified code, you will be able to see how consistent your results are. If they are, then you will be able to obtain a more accurate estimation for the correlations through our program.

Finally, we would like to note a few key observations in comparing Dr. Okun's code and our modified code:

(1) In general, the run time for the modified code is longer than the original code. On average, using the provided raster file, we saw that the modified code took roughly 1.25 times longer to run than the original code. This comes from the fact that the program had to perform more 2×2 sub matrix exchanges in order to obtain the desired values for **d**.

(2) We also saw that the code, while very rarely, was more likely to fail when we did not allow for the tolerance on **d**. The code would be stuck in a seemingly infinite loop while performing the 2×2 sub matrix exchanges, showing that sometimes, the desired values could not be obtained. We noticed that this was more likely to happen

to the smaller matrices that we observed. We were able to account for this failure by utilizing a while loop to regenerate a matrix to make up for it.

5. Discussion

In analyzing the raster marginals model with coupling terms method, we were only able to use one sample raster file, provided by Dr. Okun. Thus, our results can only be applied to similar rasters, not necessarily for rasters where the data is more sparse or more compact, not for rasters with strong positive correlations or negative correlations, and not for rasters that contain data on more neurons. Especially for rasters that observe more neurons, the model would allow for a greater error and could possibly produce significantly different results. For this reason, we believe that an important future direction would be to test our claims and results on various rasters that we did not have access to. By doing so, we believe that we could further analyze this method while confirming its validity.

Another observation that we have made for smaller matrices is how often, the correlation values are not consistent and how the values could range from zero correlation to a strong, positive, linear relationship between one pair of neurons, which is quite problematic. While smaller matrices, in general, may not be of much interest, it may be beneficial to determine if the provided three parameters are actually enough to determine pairwise correlations between pairs of neurons. It may be of interest to find more parameters that would allow us to have consistent results to more accurately predict such correlations values. If the model does not work well with various other rasters as mentioned above, perhaps new parameters could allow the model to be stronger.

A final direction would be to look into the solution spaces for matrices with prescribed row sum, column sum, and inner product constraints. In Okun *et al.*'s previous paper where they also estimated the pairwise correlations, they generated matrices that satisfied only the row sum and column constraints [5]. Since there are estimates of solution spaces for matrices with prescribed row sum and column constraints, it would be interesting to note the differences in solution space sizes of matrices that now included the new $\mathbf{d}$ constraint. In addition, there are several papers that discuss sampling methods from matrices with prescribed row sums and column sums, but not for matrices with prescribed row sums, column sums, and inner product constraints [2]. Thus, it may be beneficial to explore such sampling to better understand Okun *et al.*'s sampling method.

Acknowledgements

We would like to first and foremost thank our mentor, Dr. Anne Shiu. This research experience would not have been possible if not for her mentorship and guidance. We would also like to thank Kaitlyn Phillipson, Ola Sobieska, and Robert Williams for their assistance, as well as Mitchell Eithun for helpful discussions. Finally, we would like to thank Dr. Michael Okun for generously answering our questions and for providing data for this research.

This research was conducted as part of the NSF-funded REU program in Mathematics at Texas A&M University (DMS-1460766), Summer 2016.

References

[1] Richard A. Brualdi. Algorithms for constructing (0, 1)-matrices with prescribed row and column sum vectors. *Discrete Mathematics*, 306(23):3054–3062, 2006.

[2] Persi Diaconis and Bernd Sturmfels. Algebraic algorithms for sampling from conditional distributions. *The Annals of Statistics*, 26(1):363–397, 1998.

[3] Sonja Grün. Data-driver significance estimation for precise spike correlation. *J Neurophysiol*, 101(3):1126–1140, 2009.

[4] Michael Okun, Nicholas A. Steinmetz, Lee Cossell, M. Florencia Iacaruso, Ho Ko, Pter Barth, Tirin Moore, Sonja B. Hofer, Thomas D. Mrsic-Flogel, Matteo Carandini, and Kenneth D. Harris. Diverse coupling of neurons to populations in sensory cortex. *Nature*, 521(7553):511–515, 2015.

[5] Michael Okun, Pierre Yger, Stephan L. Marguet, Florian Gerard-Mercier, Andrea Benucci, Steffen Katzner, Laura Busse, Matteo Carandini, and Kenneth D. Harris. Population rate dynamics and multi-neuron firing patterns in sensory cortex. *The Journal of Neuroscience*, 32(48):17108–17119, 2012.

Department of Mathematical Sciences, Clemson University, Clemson, SC 29634
E-mail address: lbozema@g.clemson.edu

Department of Mathematics, University of Puerto Rico, Rio Piedras Campus, San Juan, PR 00931
E-mail address: adriana.morales@upr.edu