Identifying Kinematics from Microelectrode Arrays in Motor Cortex via Wiener and Kalman Filters

Mark B. Rosenberg

Abstract—

Wiener and Kalman filters were trained from two sets of data, real and synthetic, and were applied on corresponding testing data. They predicted with noticeable accuracy the kinematic trajectory given neural firing data.

Index Terms—**Wiener Filter, Kalman Filter, Motor Cortex**

I. Introduction

An important goal in brain-machine interface research is to develop methods of decoding neural data into motor output. Such methods would aid in the design of brain-machine interfaces that respond accurately to a user's thoughts. The development of these brain-machine interfaces would be important to the quality of life of users with damaged nervous systems. This paper analyses two of these methods: the Wiener filter and the Kalman filter.

Two data sets are provided, labeled real and synthetic. The real data was obtained from measurements, and the synthetic data was generated using the leaky integrate-and-fire model. Each data set contained kinematic data and neural data. The kinematic data used was 3-dimensional position of an animal's wrist over time. The neural data was binned spikes (whole numbers) from neurons firing within short intervals. The real data had sampling frequency 10 samples/sec, and was decomposed into 10,000 samples of training data, and 5,000 samples of testing data. The synthetic data had sampling frequency 20 samples/sec, and 3120 samples whose division into training and testing data was a free parameter chosen to be one half. The purpose of partitioning the data into training and testing was to allow for supervised learning, which determined the parameters of the Wiener and Kalman filters.

The brain is one of the most complex systems that has been identified. It has been determined that the motor cortex is involved with movement. Therefore it is reasonable to hypothesize that the pattern of neuron firings in the motor cortex is related to the kinematics of the organism. Applying the Wiener and Kalman filters to other areas of the brain would be expected to result in no correlation between the neural data and the kinematics, if those parts of the brain are known to be uninvolved in movement. The ability to predict a mapping from neural data to behavior with significant accuracy would be a significant step forwards to developing brain-machine interfaces and to understanding the brain.

II. Wiener Filter

The Wiener filter [1] is a matrix of coefficients that multiply neural input to generate kinematics output. The motivation behind the Wiener filter for this application is to use a linear combination of the current and past L-1 (finite impulse response) neural inputs to generate the output. It is obtained by multiplying the inverse of a matrix R by a matrix P: $W = R^{-1}P$, where R and P are generated from training data. The Wiener filter is designed to minimize the mean squared error between the noisy input and the output. In this application, the input is noisy because the microelectrode array may detect a signal obtained from multiple sources within a small region. By writing out the expression for the mean squared error, taking the derivative, and setting it to zero, a solution can be obtained in the form described above, where R and P are specified in the following way.

With M as the number of neurons, R is an M*L square matrix that we interpret as an MxM array of LxL square matrices. Each LxL matrix is a Toeplitz matrix, which means there are 2L-1 unique elements – a fact that is helpful for reducing the computational complexity from O(L^2) to O(L).

With the introduction of the variable tau, which ranges from –(L-1) to +(L-1), each of the 2L-1 unique elements of the LxL matrix is the correlation between two neurons, i and j, with time lag tau, where i and j are the row and column indices of the MxM array. Since correlation is a symmetric operation, only the LxL matrices for unique unordered pairs of neuron indices need to be computed, and the rest of the R matrix can be filled in by recognizing that the transpose of any LxL matrix fills in the part of the R matrix for the opposite sequence of those neuron index pairings.

The P matrix can be interpreted as an MxC array of Lx1 vectors, where each Lx1 vector is the cross-correlation vector between a neuron and one dimension of hand positions, and C is the number of dimensions of the kinematic data: 3. In generating P it is important to remember that the delay tau, which ranges from 0 to L-1 is such that past neural data, rather than future neural data, is correlated with current kinematic data.

The Wiener filter has a free parameter L, which is the number of taps of an FIR filter. Performance measures of correlation coefficient and mean squared error are determined over values of L from 1 to 20 for both the real and synthetic data sets. From the plots of these in Fig. 1-4, it is determined that the real data performs best with L=1, and the synthetic data with L=10. High correlation and low MSE are desirable.

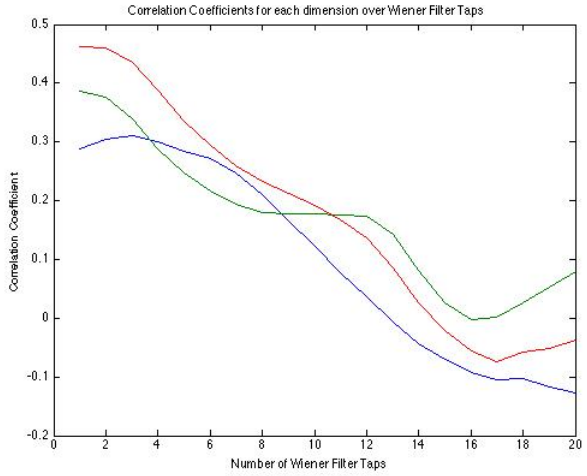

Figure 1: Correlation Coefficients for Real Data

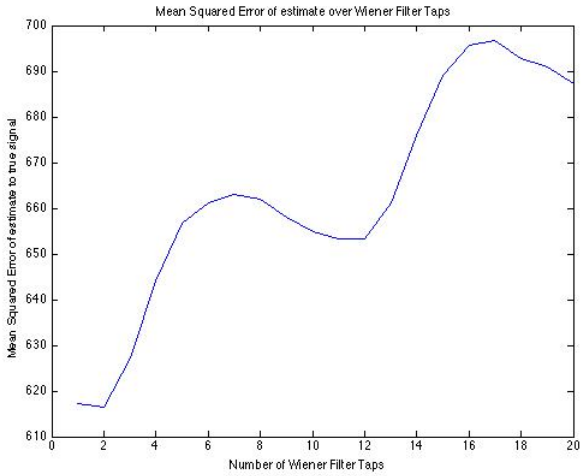

Figure 2 MSE for Real Data

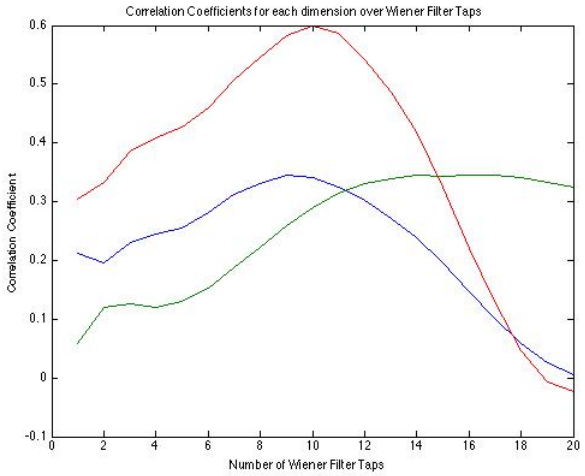

Figure 3: Correlation Coefficients for Synthetic Data

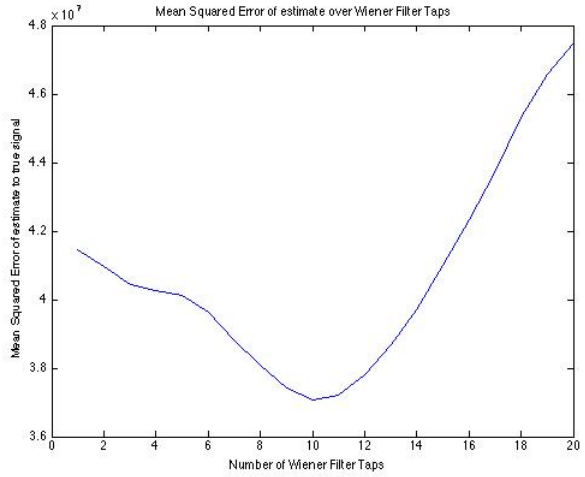

Figure 4: MSE for Synthetic Data

To compute the W matrix, the R matrix has to be inverted, so it must be non-singular. However, if there are neurons that do not have any spikes throughout the entire length of time measured, not

only do they not provide any useful information, but also they prevent the R matrix from being invertible. To address this issue, neurons with no data are removed from the training and testing data before applying filters. This has the additional affect of reducing the (high) dimensionality of the Weiner filter, since the variable M decreases. Since it is desirable to have lower dimensionality for computational purposes, neurons with mean firing values that are less than one are removed. As a result, computations are very fast, on the order of seconds.

III. Kalman Filter

In this application, the Kalman filter requires the interpretation that the kinematics are the state variable, and the neural data are the observations. This model does not imply that the relationship between the state and the observations is such that the state causally determines the observations; it only implies that there exists some linear relationship, and that we have knowledge of the observations and are trying to predict the state.

The Kalman filter [2] involves a generative model of how the neural observations are obtained from the kinematic state: $z_k = H_k x_k + q_k$, and how the kinematic state evolves over time: $x_{k+1} = A_k x_k + w_k$. The values q_k and w_k respectively come from zero-mean normal distributions with covariance matrices Q_k and W_k. By assuming the matrices A, H, Q, and W are constant over time, we can estimate them simply. A and H minimize the mean squared error of the two sides of the model's equations. Their closed-form solution is: $A = X_2 X_1 \left(X_1 X_1^T \right)^{-1}$ and $H = Z X^T \left(X X^T \right)^{-1}$, where $X_0 = \left[x_1 \ \ldots \ x_{L-1} \right]$ and $X_1 = \left[x_2 \ \ldots \ x_L \right]$. Furthermore, W and Q are obtained from A and H such that $W = \left(X_2 - A X_1 \right)\left(X_2 - A X_1 \right)^T / (M-1)$ and $Q = \left(Z - HX \right)\left(Z - HX \right)^T / M$, where M is the number of time steps, not to be confused with the variable M used in the Wiener filter.

The Kalman filter uses the four matrices described above in two steps. The first step predicts an estimate of the state and its error covariance matrix: $\hat{x}_k^- = A\hat{x}_{k-1}$, $P_k^- = A P_{k-1} A^T + W$, and the second step uses that estimate and the neural data to update the estimate with the following equations:

$$K_k = P_k^- H^T \left(H P_k^- H^T + Q \right)^{-1}$$

$$\hat{x}_k = \hat{x}_k^- + K_k \left(z_k - H\hat{x}_k^- \right)$$

$$P_k = \left(I - K_k H \right) P_k^- .$$

To initialize the process, $P_0 = 0$ and the first kinematic estimate was taken to be the initial value of the kinematic trajectory.

IV. Results

The results for the real data is that the Wiener filter and Kalman filter that were implemented performed on par with the Wiener filter from the protected MATLAB script. This is a validation that the Wiener filter I coded works correctly. The Kalman solution had nearly identical correlation coefficients to the Wiener solution (Fig. 7), but its peaks were slightly less prominent than those of the Wiener solution (Fig. 7). This is also demonstrated by comparing Fig. 5 and Fig. 6, as the Kalman solution (red, Fig. 6) has smaller amplitudes than the Wiener solution (red, Fig. 5).

Although the Wiener solution of the synthetic data in Fig. 8 looks like it does not match the true test trajectory, Fig. 10 indicates that it's correlation coefficient for the z-coordinate is 0.6, which is relatively good, and the plot of the z-coordinate solution in Fig. 10 matches the true signal to a noticeable degree. And even though the amplitude of the Kalman solution is very small (Fig. 9, Fig. 10), resulting in a larger MSE than the Wiener solution, the correlation coefficient is also 0.6, which is a positive indicator that some aspect of the Kalman filter works well in this scenario.

In both the real and synthetic data sets, the 3-dimensional plots of the trajectory show that the Kalman solution had a lower amplitude than did the Wiener solution. Perhaps one reason why the Wiener solution outperformed the Kalman solution with respect to MSE is because the Wiener solution had a free parameter that was tuned for optimal results, whereas the Kalman solution did not.

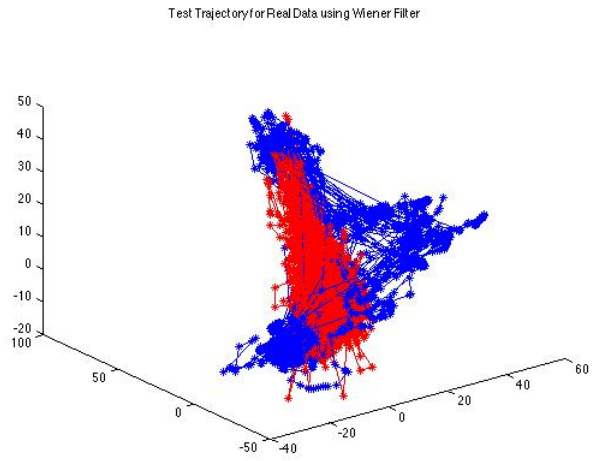

Figure 5: Test Trajectory for Real Data using Wiener Filter

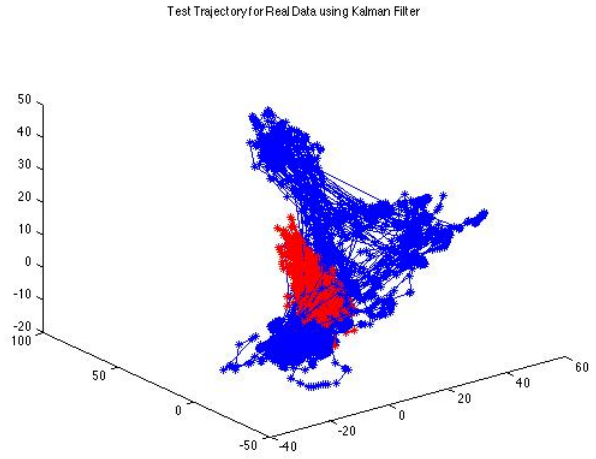

Figure 6: Test Trajectory for Real Data using Kalman Filter

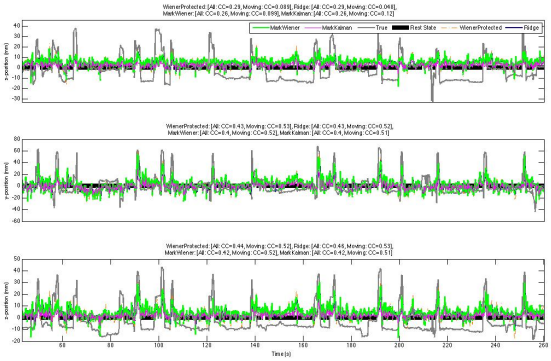

Figure 7: Test Trajectory for Real Data with both filters

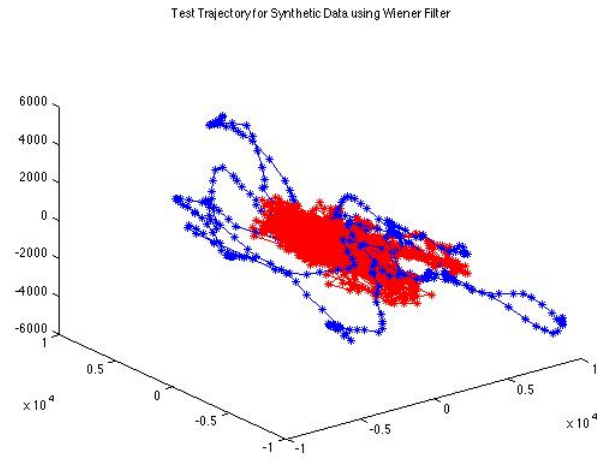

Figure 8: Test Trajectory for Synthetic Data using Wiener Filter

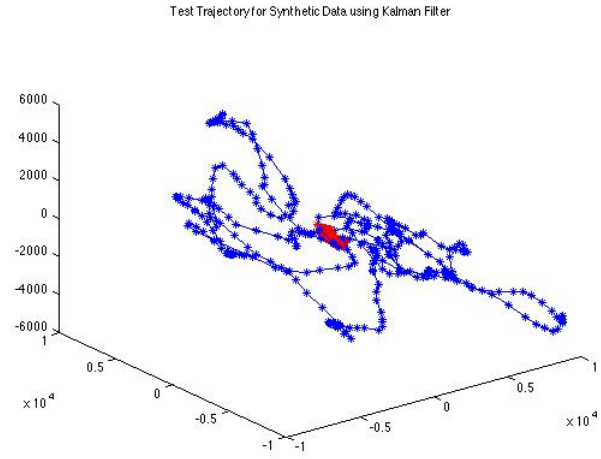

Figure 9: Test Trajectory for Synthetic Data using Kalman Filter

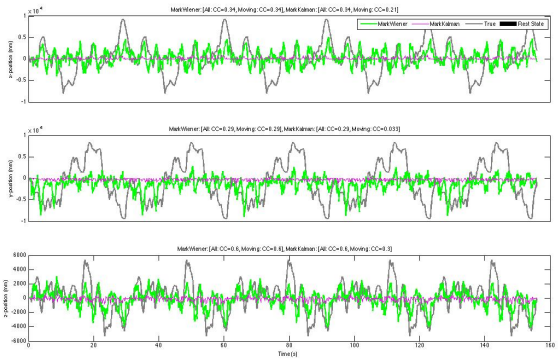

Figure 10: Test Trajectory for Synthetic Data with both filters

A potential reason why the Kalman solution performed worse for the synthetic data than for the real data may be demonstrated in Fig 11. and Fig 12. These figures show how the norm of difference between consecutive matrices decays exponentially for three matrices: K_k, P_k, and P_k^minus. For the real data, the matrices converge after about 20 or 40 time steps. But for the synthetic data, two of

them have only just started to converge after 1400 time steps, which is near the end of the trajectory.

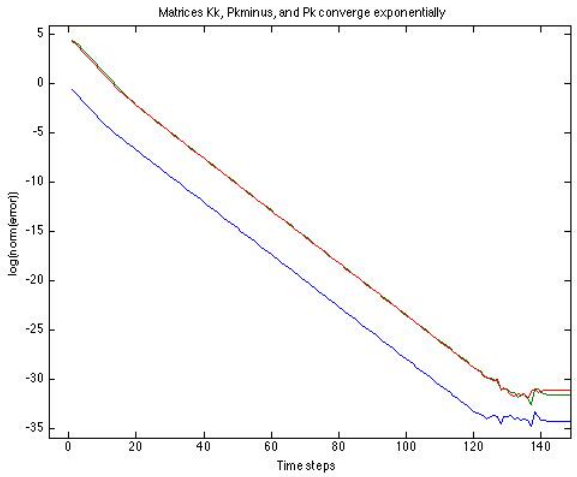

Figure 11: Convergence of Kalman Matrices for Real Data

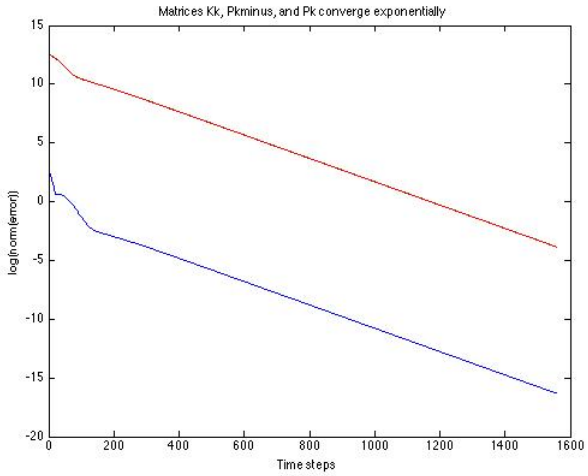

Figure 12: Convergence of Kalman Matrices for Synthetic Data

An interesting result occurred when a typo in the equation for P_k^minus occurred. Instead of P_k^minus being a function of P_(k-1), the typo scenario used P_(k-1)^minus. The effect of this was that the difference between the Kalman solution and the Wiener solution converged to zero; in other words, the Kalman solution converged to the Wiener solution. This is demonstrated in Fig. 13.

V. Conclusions

The results indicate that there is non-negligible success in applying the Wiener and Kalman solutions, which is a great result in the context that the brain is highly complex and that any method that demonstrates a noticeable relationship between

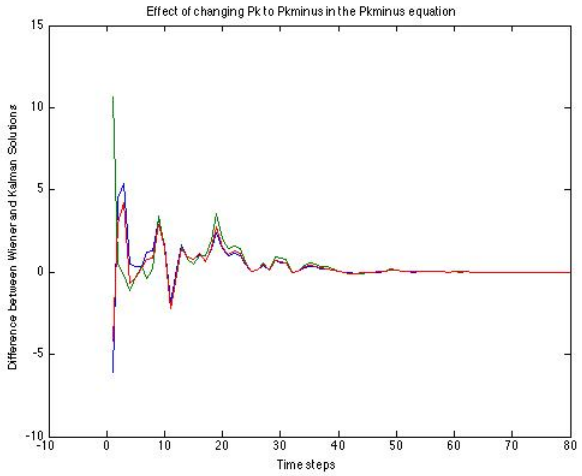

Figure 13: The effect of changing Pk to Pkminus in the Pkminus equation is convergence of the Kalman solution to the Wiener solution

neural data and kinematics deserves further study and refinement. One reason the filters performed less well on the synthetic data could be that the models used to generate the synthetic data had properties that were less conducive to the assumptions the Wiener and Kalman methods relied upon. In fact, the reason for significant difference in performance on the two data sets must be related to the differences in how the data was generated or obtained since the Wiener and Kalman methods applied were identical for both data sets. Both methods assume a linear relationship between the neural data and kinematics, which is a simplifying assumption that makes the mathematics more simple, which is an advantage at the expense of not being a true assumption. The important criteria to determine is how reasonable the linearity assumption is; this can be determined by how successful the models are at predicting kinematics, which in this study has been noticeable but less than ideal.

References

[1] Sanches, J.C., Principe, J.C. Brain-Machine Interface Engineering. 2007. Morgan and Claypool.

[2] Wu, W., et al. *Inferring hand motion from multi-cell recordings in motor cortex using a Kalman filter.* In SAB Workshop on Motor Control in Humans and Robots: on the Interplay of Real Brains and Artificial Devices. 2002. University of Edinburgh, Scotland.