An Efficient Approximation for the Real-Time Implementation of the Mixture of Trajectory Models Decoder

William Bishop*, Byron M. Yu†¶||, Gopal Santhanam†, Afsheen Afshar†‡,
Stephen I. Ryu†§, and Krishna V. Shenoy†¶
*The Johns Hopkins University Applied Physics Laboratory, Laurel, Maryland;
and †Department of Electrical Engineering, ‡Medical Scientist Training Program,
§Department of Neurosurgery, and ¶Neurosciences Program, Stanford University, Stanford, California;
and ||Gatsby Computational Neuroscience Unit, University College London, London, United Kingdom

Abstract— **The Mixture of Trajectory Models (MTM) decoder has been used to reconstruct arm trajectories from neural activity. While it produces reasonable results, the computational demands of previously published versions may be too high for many real-time systems. We have developed a novel method of approximating the MTM state posteriors that does not require the use of Newton's method. We show that this method results in only a small decrease in decoding performance yet reduces computational cost by 56.4%. Additionally, an MTM algorithm using this method of approximating the state posteriors produces more accurate decoded trajectories when using small bin sizes than an MTM algorithm using a Gaussian observation model. The more efficient formulation of the MTM algorithm presented here provides an alternative approximation of this algorithm for use on resource constrained embedded systems.**

I. Introduction

Neural decode algorithms have been used with much success in recent years to estimate arm and hand state. The mixture of trajectory models (MTM) decoder is one such algorithm for estimating end effector trajectory during goal directed reaches [1]. A real-time implementation of this algorithm has recently been demonstrated [2] using a personal computer (PC). While this is a positive step, it is not clear the implementation demonstrated in [2] would successfully transition to platforms with limited processing capacity or restricted power budgets.

A key concern when considering the use of previously demonstrated versions of the MTM algorithm for real-time decoding is the reliance of those implementations on Newton's method to find the mode of the state posteriors [1], [2]. Be definition, a real-time system must approximate each state posterior within a fixed time step (1 ms in previous implementations [2]). For Newton's method, the number of iterations required to reach convergence is unknown *a priori*, and it is possible that the mode of a state posterior may not be found in the time allotted for these calculations. Additionally, given fixed processing power, the use of Newton's method reduces the number of trajectory models that can considered in the algorithm[1].

This work was sponsored by the Johns Hopkins University Applied Physics Laboratory under the DARPA Revolutionizing Prosthetics program, contract N66001-06-C-8005, the National Defense Science and Engineering Graduate Fellowships (BMY, GS), National Science Foundation (NSF) Graduate Research Fellowships (BMY, GS), Gatsby Charitable Foundation (BMY), Stanford Bio-X Graduate Fellowship (AA), National Institute of Neurological Disorders and Stroke/Collaborative Research in Computational Neuroscience Grant 5-R01-NS054283-03, Medical Scientist Training Program, Christopher and Dana Reeve Paralysis Foundation (SIR,KVS), Burroughs Wellcome Fund Career Award in the Biomedical Sciences, Stanford Center for Integrated Systems, NSF Center for Neuromorphic Systems Engineering at Caltech, Office of Naval Research, Sloan Foundation, and Whitaker Foundation.
Please address correspondence to william.bishop@jhuapl.edu.

These concerns have led us to derive a MTM algorithm that does not require the use of an iterative peak finding procedure. The proposed method allows for the continued use of a Poisson observation model and produces results which are only slightly less accurate than the original MTM formulation. We additionally examine MTM algorithms that incorporate Gaussian observation models (for which analytical solutions to the state posteriors exist) and compare the performance of these to the formulation proposed here as well as the original MTM algorithm.

II. Methods

The MTM state posteriors are defined by:

$$P(\vec{x}_t|\{\vec{y}\}_1^t, m) = \frac{P(\vec{y}_t|\vec{x}_t)P(\vec{x}_t|\{\vec{y}\}_1^{t-1}, m)}{P(\vec{y}_t|\{\vec{y}\}_1^{t-1}, m)} \tag{1}$$

The first term in the numerator is the observation model where $\vec{y}_t$ is the vector of observed firing rates and $\vec{x}_t$ is the arm state at time t. There are M movement regimes. In this work we will compare results between MTM implementations using Poisson and Gaussian observation models. We assume all units are conditionally independent and define the observation model (for both the Poisson and Gaussian case) as:

$$P(\vec{y}_t|\vec{x}_t) = \prod_{s=1}^{S} P(y^s_{t-lag_s}|\vec{x}_t) \tag{2}$$

[1]However, as noted in [1], MTM is an inherently parallelizable algorithm and more trajectory models can always be added if more processsing power is available.

For the Poisson case:

$$P(y^s_{t-lag_s}|\vec{x}_t) = \text{Poisson}(\exp(\vec{c}^{\,\prime}_s\vec{x}_t + d_s)\Delta) \tag{3}$$

and for the Normal case:

$$P(y^s_{t-lag_s}|\vec{x}_t) = \mathcal{N}((\vec{c}^{\,\prime}_s\vec{x}_t + d_s)\Delta, \sigma^2_s) \tag{4}$$

The second term in the numerator of equation 1 is the linear Gaussian model embodying kinematics for the m^{th} movement regime.

When possible in the rest of this paper we maintain the notational convention used in the original MTM paper [1]. For reasons which will become apparent later, we will follow the convention of Eden et al. [3] and refer to the covariance of the one-step prediction as $\mathcal{Q}_{t|t-1,m}$ and the mean as $\vec{x}_{t|t-1,m}$.

A. Derivation of MTM State Posteriors Approximation

In this section we will derive a new Gaussian approximation for the MTM state posteriors. We denote the mean and covariance of the the Gaussian approximating the m^{th} state posterior at time t as $\vec{x}_{t|t,m}$ and $\mathcal{Q}_{t|t,m}$, respectively. We also desire to estimate the normalizing constant, $P(\vec{y}^t|\{\vec{y}\}^{t-1}_1, m)$, for each model. The method of derivation is based on previous work by Eden et al. [3] who use the same method to approximate a different probability density function (pdf).

A Gaussian is fully specified by its mean and covariance. In other words, a Gaussian is fully specified by the first and second derivatives of its log-density. Previous MTM implementations have used Newton's method to find the mode of the log-posteriors and matched the first and second derivatives of the log-posteriors to the log-Gaussian approximations at this point. In this way, the means of the approximations are located at the mode of the true state posteriors (left panel of Fig. 1). While this is one way of approximating a Gaussian, others are possible.

If we presuppose the state posteriors are Gaussian, we can arrive at approximations for the mean and covariance parameters anywhere the first and second derivatives of the state posteriors are known (right panel of Fig. 1). If we chose that point to be $\vec{x}_{t|t-1,m}$, the equations become particulary simple. Further, as $\vec{x}_{t|t-1,m}$ is known from the one-step predictions, the proposed method removes the need for Newton's method. It should be noted that while the approximations produced by each method will normally differ, identical approximations will be produced when $\vec{x}_{t|t-1,m}$ coincides with the mode of the true state posterior.

Using the definitions of the observation model and the trajectory model for the m^{th} model, we can write:

$$\begin{aligned} &P(\vec{x}_t|\{\vec{y}\}^t_1, m) \propto \\ &\left(\prod_{s=1}^{S} P(y^s_{t-lag_s}|\vec{x}_t)\right) \times \\ &\exp\left(-\frac{1}{2}(\vec{x}_t - \vec{x}_{t|t-1,m})^{\prime}\mathcal{Q}^{-1}_{t|t-1,m}(\vec{x}_t - \vec{x}_{t|t-1,m})\right) \propto \\ &\exp\left(-\frac{1}{2}(\vec{x}_t - \vec{x}_{t|t,m})^{\prime}\mathcal{Q}^{-1}_{t|t,m}(\vec{x}_t - \vec{x}_{t|t,m})\right) \end{aligned} \tag{5}$$

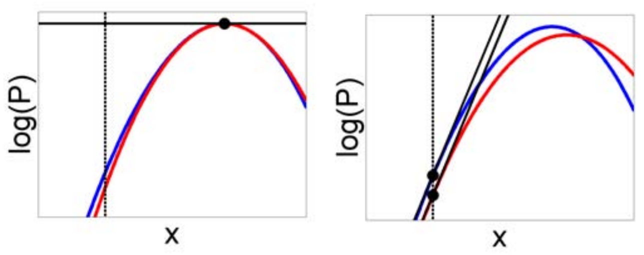


Fig. 1. One-dimensional illustration of approximating a non-Gaussian (blue) probability density function (pdf) as Gaussian (red) using Laplace's method (left) and the method presented in this paper (right). Natural log of the value of each pdf plotted on the ordinate axis. Vertical black lines indicate location of the mean of the one-step prediction. Black dots indicate the points the first derivatives (indicated by solid black lines) and second derivatives (not illustrated) are matched at.

The second line of equation 5 is the MTM state posterior for the m^{th} model, while the third line is an equation for a Gaussian approximation to this distribution.

If we take the log of lines 2 and 3, differentiate with respect to $\vec{x}_t$, solve around $\vec{x}_t$ equal to $\vec{x}_{t|t-1,m}$ and rearrange we find:

$$\vec{x}_{t|t,m} = \vec{x}_{t|t-1,m} + \mathcal{Q}_{t|t,m}\sum_{u=1}^{U}\left[\frac{\partial \log P(y^s_{t-lag_s}|\vec{x}_t)}{\partial x_i}\right]_{\vec{x}_{t|t-1,m}} \tag{6}$$

Using the same method, we can find second derivatives around $\vec{x}_{t|t-1,m}$ and rearrange to find:

$$\mathcal{Q}^{-1}_{t|t,m} = \mathcal{Q}^{-1}_{t|t-1,m} - \sum_{u=1}^{U}\left[\frac{\partial^2 \log P(y^s_{t-lag_s}|\vec{x}_t)}{\partial x_i \partial x_j}\right]_{\vec{x}_{t|t-1,m}} \tag{7}$$

These equations define a new method of approximating the MTM state posteriors. Equation 7 is first used to approximate the posterior covariance for the m^{th} model and equation 6 is then used to find an approximation for the mean. In order to use equations 6 and 7, we need to substitute for $P(y^s_{t-lag_s}|\vec{x}_t)$. Using the definition of the observation models, we find:

For the Poisson observation model:

$$\frac{\partial \log P(y^s_{t-lag_s}|\vec{x}_t)}{\partial x_i} = c_{s,i}\left[y^s_{t-lag_s} - \Delta\exp(\vec{c}^{\,\prime}_s\vec{x}_t + d_s)\right] \tag{8}$$

$$\frac{\partial^2 \log P(y^s_{t-lag_s}|\vec{x}_t)}{\partial x_i \partial x_j} = -c_{s,i}c_{s,j}\Delta\left[\exp(\vec{c}^{\,\prime}_s\vec{x}_t + d_s)\right] \tag{9}$$

and for the Normal observation model:

$$\frac{\partial \log P(y^s_{t-lag_s}|\vec{x}_t)}{\partial x_i} = c_{s,i}\Delta\left[\frac{y^s_{t-lag_s} - \left(\vec{c}^{\,\prime}_s\vec{x}_t + d_s\right)\Delta}{\sigma^2_s}\right] \tag{10}$$

$$\frac{\partial^2 \log P(y^s_{t-lag_s}|\vec{x}_t)}{\partial x_i \partial x_j} = \frac{-c_{s,i}c_{s,j}\Delta^2}{\sigma^2_s} \tag{11}$$

The normalizing constant for all three posteriors can be approximated by a "Laplace like" method:

$$P(\vec{y}^t|\{\vec{y}\}_1^{t-1}, m) = P(\vec{y}_t|\vec{x}_{t|t})P(\vec{x}_{t|t}|\{\vec{y}\}_1^{t-1}, m)\sqrt{\det\left(2\pi\mathcal{Q}_{t|t,m}\right)} \tag{12}$$

Bayes' theorem can be used to show that equation 12 can be applied when $\vec{x}_{t|t}$ is not a mode of the true state posteriors, as it is when performing a strict Laplace approximation.

B. *Testing*

All algorithms were implemented in MATLAB 2008a (Mathworks, Nantucket, MA). Data from two subjects performing a delayed center out task to eight targets (described elsewhere [1]) was used in 10-fold cross validation to examine the performance of each formulation of the MTM algorithm. Dataset *G* consisted of 144 trials per reach target with recordings from 98 units. Dataset *H* consisted of 104 trials per reach target with recordings from 199 units.

When fitting models, the optimal lag for each unit was determined by examining observation model deviance. In all cases 10 ms non-overlapping bins were used to match the bin sizes used in the original MTM paper [1]. To examine the effect of bin size on decoder accuracy when using a Gaussian observation model, tests were also run with a bin size of 50 ms. Spike counts were square root transformed [4] when using a Gaussian observation model so that the variance of the transformed data better fit a Gaussian model.

To compare real-time performance, each implementation of the MTM algorithm was implemented in a virtual integration environment (VIE) [5] and run in real-time on a personal computer with a 3.2 GHz Pentium 4 processor. The realtime implementations of each algorithm were optimized by grouping constant terms together and precomputing these before run-time.

III. Results

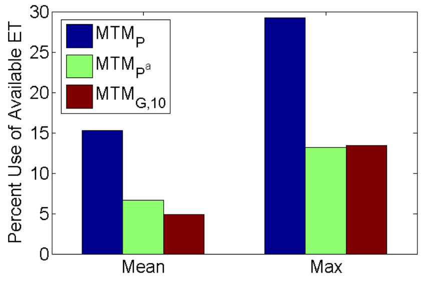


Fig. 2. Mean percent use of available execution time during peri-movement decode for the MTM_P, MTM_{P^a} and $MTM_{G,10}$ implementations of the MTM algorithm. Results were calculated from the decode of one fold of trials from dataset G. Standard error for mean values was .03%, .01% and .01%, respectively.

For the remainder of this paper, we will refer to the original implementation of the MTM algorithm [1] as MTM_P, the implementation using the new method of approximating the state posteriors as MTM_{P^a} and implementations using a Gaussian observation model with bin sizes of 10 ms and 50 ms as $MTM_{G,10}$ and $MTM_{G,50}$, respectively.

To compare computational expense, each implementation's use of available execution time (ET) [5] during real-time execution was calculated. As the innovation presented in this paper relates to the decode of peri-movement neural activity and not the decode of delay activity, usage of available ET was calculated only for the portion of the decodes taking place during the peri-movement period.

By this measure, on average the MTM_{P^a} implementation was 56.4% more efficient than the MTM_P implementation, and the $MTM_{G,10}$ version was 67.8% more efficient (Fig. 2).

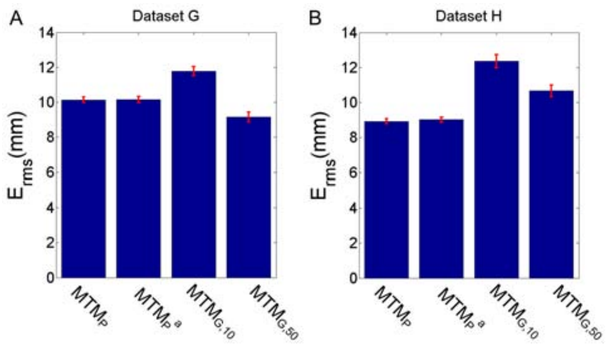
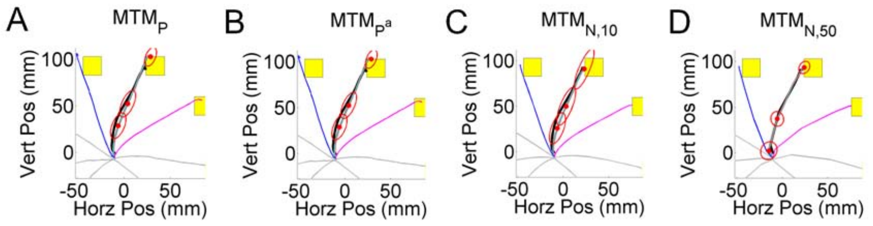


Fig. 3. Root mean square error with standard error for four implementations of the MTM decoder for datasets G (98 units) and H (199 units).

The root mean square error (E_{RMS}) between predicted and actual trajectories was calculated for each implementation of the MTM algorithm, and results can be seen in Fig. 3. While the results of the MTM_{P^a} decode were statistically different from those of the MTM_P implementation (Wilcoxon signed rank test, P $<$ .0001), the mean decrease in performance associated with the new method of approximation was only .05 $\pm$ .01 millimeters. Compared to the average E_{RMS} values for each decode (9.65 $\pm$ .08 mm) this is a very minor decrease in performance. E_{RMS} values were higher for the $MTM_{G,10}$ algorithm (Wilcoxon signed rank test, P $<$ .0001), but increasing bin size reduced decodinig error as shown by the performance of the $MTM_{G,50}$ algorithm (Wilcoxon signed rank test, P $<$ .0001).

Representative trajectories from each implementation were examined (Fig. 4). The number of decoded trajectories that ended closest to a target other than the one to which the monkey reached were counted for dataset G (H). For the MTM_P, MTM_{P^a}, $MTM_{G,10}$ and $MTM_{G,50}$, 2.4% (0.6%), 2.4% (1.5%), 8.7% (10.4%) and 8.1% (10.5%) of trials ended in this condition, respectively. The slight increase in missed targets between the MTM_{P^a} and MTM_P implementations for dataset H is accounted for by 9 trials that "snapped to" the wrong trajectory during the final time steps of each reach when the monkey's hand was essentially at rest.

Histograms of all neural activity occurring during the peri-movement period were also produced. The fit of each obser-



4. A representative trial for each implementation of the MTM decoder. True trajectory is in in black and estimated trajectory in red. Red ellipses denote 95% confidence ellipses at specific points along the decoded trajectory. For panels A-C, these ellipses were plotted at 130, 170 and 330 ms. For panel D, these ellipses were plotted at 150, 200 and 350 ms. Component trajectories are also plotted in grey with the the three component trajectories with highest prior probability plotted in cyan, blue and magenta.

vation model type was assessed by calculating the probability of observing a spike count based on arm state for each peri-movement bin. These probabilities were averaged over all bins, and the results for one unit are presented in Fig. 5. It can be seen that the Gaussian observation models predicted non-zero probability for negative spike counts and tended to underestimate the probability of positive spike counts.

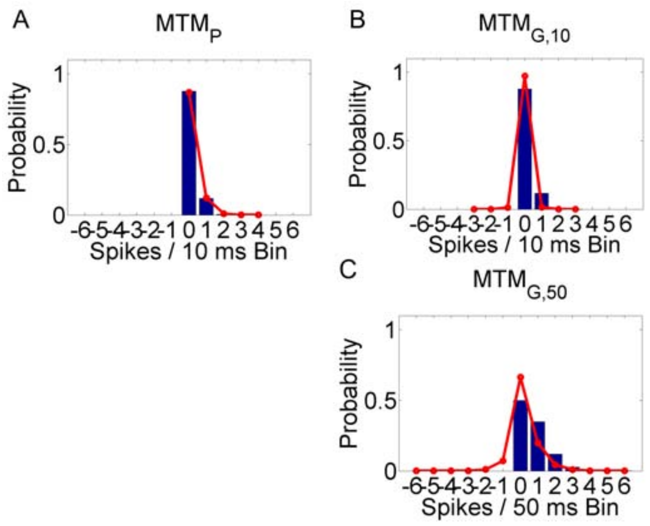


Fig. 5. The probability of observing a peri-movement spike count calculated from all trial data for one unit of dataset G. Blue bars indicate normalized spike histograms. Red lines indicate probability of each spike count calculated using fit observation models and recorded arm states.

IV. Discussion

We have presented an approximation for the MTM state posteriors and examined the use of MTM algorithms with Gaussian observation models.

The MTM_{P^a} implementation may be a good choice when computational resources are constrained and temporal and spatial resolution must be simultaneously maintained. In these situations, the use of MTM_{P^a} may produce results that are only slightly less accurate than MTM_P but at a significant discount in computational cost.

For situations where processing power is not a concern or real-time decoding is not required, the MTM_P implementation guarantees that the mean of the approximations will be located at the mode (or single most likely point) of the state posteriors. Additionally, compared to MTM_{P^a}, the MTM_P implementation never exhibited any tendency to "snap to" the wrong movement regime at the end of a reach.

While more investigation is needed into this behavior, at the end of a reach, trajectory models will be near a stable point and always predict little end-effector motion. However, due to the possibility of neural activity unaccounted for in the observation model, predicted end-effector state may diverge from that of the one-step prediction. In these situations, it appears more accurate decoded trajectories are produced by approximating the state posteriors around their modes.

The MTM_G implementation are alternative approximations for the MTM algorithm that further reduce computational expense. However, for small bin sizes, decodes show an increase in E_{rms} values, and for both bin sizes examined, the percentage of trials that ended near incorrect targets was fairly high.

It is important to keep in mind that these are decodes of highly stereotyped, ballistic movements, and it remains to be seen how these implementations would perform on the decode of non-stereotyped movements. This, along with the study of other efficient approximation methods, such as the unscented Kalman filter, should be the subject of future work.

References

[1] B. M. Yu, C. Kemere, G. Santhanam, A. Afshar, S. I. Ryu, T. H. Meng, M. Sahani, and K. V. Shenoy, "Mixture of trajectory models for neural decoding of goal-directed movements." *J Neurophysiol*, vol. 97, no. 5, pp. 3763–3780, May 2007. [Online]. Available: http://dx.doi.org/10.1152/jn.00482.2006

[2] W. Bishop, B. M. Yu, G. Santhanam, A. Afshar, S. I. Ryu, K. V. Shenoy, J. R. Vogelstein, J. Beaty, and S. Harshbarger, "The use of a virtual integration environment for the real-time implementation of neural decode algorithms," in *IEEE Intl Conf Symposium on Eng Med Biol (EMBC 2008)*, 2008.

[3] U. T. Eden, L. M. Frank, R. Barbieri, V. Solo, and E. N. Brown, "Dynamic analysis of neural encoding by point process adaptive filtering." *Neural Comput*, vol. 16, no. 5, pp. 971–998, May 2004. [Online]. Available: http://dx.doi.org/10.1162/089976604773135069

[4] M. S. Bartlett, "The square root transformation in analysis of variance," *Supplement to the Journal of the Royal Statistical Society*, vol. 3, no. 1, pp. 68–78, 1936. [Online]. Available: http://www.jstor.org/stable/2983678

[5] W. Bishop, R. Armiger, J. Burck, M. Bridges, M. Hauschild, K. Englehart, E. Scheme, J. Vogelstein, J. Beaty, and S. Harshbarger, "A real-time virtual integration environment for the design and development of neural prosthetic systems," in *IEEE Intl Conf Symposium on Eng Med Biol (EMBC 2008)*, 2008.