**Trajectory Piecewise-Linear Model Order Reduction**

**for Neural Mass Modeling**

By

Alexander Herzog

A Thesis Submitted to the Graduate
Faculty of Rensselaer Polytechnic Institute
in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

Major Subject: MECHANICAL ENGINEERING

Approved by the
Examining Committee:

_____
Suvranu De
Thesis Advisor

_____
Wei Ji, Member

_____
Zahra Sotoudeh, Member

Rensselaer Polytechnic Institute
Troy, New York
November 2012
(For Graduation December 2012)

**CONTENTS**

**LIST OF TABLES**

**LIST OF FIGURES**

## ABSTRACT

Model order reduction (MOR) methods approximate the input-output relationship of high-dimensional dynamical systems using less complex systems. MOR works best with linear systems but a method for reducing weakly nonlinear systems also exists. The established approaches of using linear algebraic methods for linear systems and Taylor series expansions spanning higher order derivatives for weakly nonlinear systems do not work well for strongly nonlinear systems.

The trajectory piecewise-linear (TPWL) method reduces the computational costs of the dynamics of a nonlinear system. It does this by partitioning the state space into discrete regions of reduced linear models. The reduced linear models are generated along a trajectory through the state space of the full nonlinear system. TPWL simulates the full system's nonlinear output using the weighted sum of nearby reduced models.

Recently, neural mass models have grown in popularity as a means of simulating neural firing rates. Neural mass models represent the activation dynamics of small brain regions that can be connected together in order to simulate large networks of brain regions. The dynamics of these models are chaotic, highly nonlinear, and strongly coupled making them a challenge for MOR.

This thesis details an implementation of TPWL for simple neural mass models at small time scales that yields reductions in simulation time and state space dimensionality. In particular it is noted that the full nonlinear system is more efficient to compute at small simulation times but by approximately 0.11 seconds and longer of simulation time, the TPWL method overtakes it in computational speed. Outstanding challenges to MOR for neural mass models and future work are also discussed.

## 1 Introduction and Historical Review

Current models that attempt to accurately represent electrical signals in the human brain discretize the brain into regions of interest but the question of element size becomes important [1]. Individual neurons are too numerous to simulate while behaviorally distinct regions of the brain, as defined by psychologists, are too large to produce accurate results. The solution may be found in the field of multiscale modeling, since electrical models of the brain are derived at the microscopic spatial scale and functional models are defined over the macroscopic scale. The goal of the model is to attain a sufficiently accurate model that achieves effective coarsification with minimal complexity. In order to help determine the mapping between structure and function, neural mass models are being used by researchers at the Human Research and Engineering Directorate at the Army Research Lab in Aberdeen Proving Ground, MD to create networks of small neural brain regions to assist in the structure-function connectivity identification [2].

A neural mass model (NMM) represents the activation dynamics of a small brain region [3]. NMM units can be connected to simulate large networks of brain regions. Networks like this can be used in the verification process of the method to map structure with function. However, the simulation of a large network of brain regions is very computationally expensive. Each neural mass unit can hold multiple sub-populations, each representing different neural frequencies. This enables a single unit to have a unique frequency spectrum. Each sub-population has six dynamic state equations and a normally distributed random extrinsic input which create a chaotic, highly nonlinear, and strongly coupled system. A network of NMM units connected together, each with multiple subpopulations, can mean a very complex and large system. This makes simulating over long time periods computationally expensive. With multiple NMM units, the system becomes a multiple output system. The connection between separate NMM units within a network is also governed by a time delay function which further complicates the system. For these reasons, in this thesis, we only address the dynamics of a single NMM unit with multiple sub-populations in order to keep the system a SISO system with no time delays.

In order to reduce the computational costs of simulating complex systems, model order reduction methods can be implemented in order to capture essential features of a complex system in a simplified manner [4]. Model order reduction (MOR) can be used to represent a larger and more complicated system as an approximated smaller and simpler system. This is done commonly with linear time invariant systems by using projection methods and other linear algebra techniques such as using Krylov subspace projections and Arnoldi reduction algorithms

[5], [6], [7]. These methods take the state space of a complex system and project it onto a state space of reduced size so the matrix operations are significantly reduced.

For weakly nonlinear systems, linear MOR techniques may be sufficient for simple inputs or small time scales. Alternatively, the MOR techniques can be broadened using Taylor series expansions to span higher order derivatives. This allows the techniques to capture more of the nonlinear quality of the system. However, this will not work for highly nonlinear or coupled dynamical systems. This is because, with Taylor series expansions, the solution is only accurate while it remains near the state that the system is expanded about. If the system being reduced diverges sufficiently far from the initial expansion point, such as the neural mass model system, then a MOR technique about a single expansion will not be accurate for the entire simulation.

The method used in this thesis is the trajectory piecewise-linear (TPWL) model order reduction method [8]. TPWL represents a nonlinear system by dividing the state space into reduced, linear models generated around expansion points [9], [10], [11]. These expansion points in the state space are placed along a trajectory in the state space relevant to the solution of the full nonlinear system. By limiting the expansion points in this manner, the reduced order model minimizes the amount of computation without evaluating irrelevant points in the state space. The trajectory is determined to be relevant by simulating the full nonlinear system ahead of time using a simple training input signal that will be similar in magnitude to an input signal experienced by the system. The reduced order model solution is calculated by solving the linear models. Weights are given to the linear models in a manner that gives greater significance to the models that are closer to the current state. As mentioned before with regard to a Taylor series, when the state is sufficiently close to an expansion point, it will be more accurate. This method allows the nonlinear behavior of the system to be represented in the reduced order model without actually computing the nonlinear system. The weighted sum of linear models requires less computational cost than the full nonlinear system.

The implementation of TPWL for NMM units has also required additional tools to complement the MOR method. Since the NMM states are chaotic, there is a requirement for many more extraction points than other simpler systems. This is not difficult to create, but the simulation costs increase greatly when it has to compute the weighted sum of each linear model. Instead, a nearest neighbors approach can be used to keep track of the linear models that are closest to each other. At any time in the simulation, the nearest linear model is tracked and the nearest neighbors approach is then able to indicate which linear models.

This thesis also explores the challenges associated with using model order reduction methods for chaotic systems. The neural mass models are randomly excited oscillators with

multiple frequencies which makes it difficult to plot trajectories through the state space that are relevant to all possible solutions.

This thesis is organized into the following three chapters. Chapter 2 details the trajectory piecewise-linear model order reduction method, how to create a TPWL reduced order model, and how the TPWL method was altered in order to run faster simulations. Chapter 3 begins by describing the neural mass model system used as an example in this thesis. It then presents the results of using the TPWL on this system. Chapter 4 discusses the results presented in the previous chapter, concludes the work of the thesis and discusses future work.

## 2 Method: Trajectory Piecewise-Linear Model Order Reduction

The TPWL method for MOR attempts to compensate for the deficiencies in MOR methods for weakly nonlinear representing strongly nonlinear systems. The weakly nonlinear methods, which typically use a linear or quadratic approximation of the full nonlinear system at the initial condition, are incapable of depicting the nonlinearity in the input-output relationship of the system as they are only accurate while close to the point around which the system is created. Nonlinear systems can be represented more accurately with higher order approximations but this is too computationally expensive.

Below, this chapter describes how a TPWL model is generated for general nonlinear systems. The chapter first details how to represent a nonlinear ODE as piecewise-linear. It then presents how to generate a TPWL model. Finally, it presents how the TPWL method can be altered in order to simulate faster.

### 2.1 Representing a System as Piecewise-Linear

We can begin the process of understanding TPWL by looking at a general and simple, single-input and single-output, first order differential equation shown below,

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}) + B(\mathbf{x})u(t); \quad y(t) = C^T\mathbf{x}(t); \tag{1}$$

where $\mathbf{x} = \mathbf{x}(t) \in R^N$ is the state vector at time t, $f: R^N \rightarrow R^N$ is a nonlinear vector-valued function, $u \in R$ is the input to the system, $B \in R^N$ is a state dependent input matrix, $y \in R$ is the output, and $C \in R^N$ is the output matrix.

Using a first order Taylor series expansion about some state $\mathbf{x} = \mathbf{x_0}$, we can create a linear approximation of (1) as the equation below,

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x_0}) + A_0(\mathbf{x} - \mathbf{x_0}) + B_0u \tag{2}$$

where $A_0 = A(x_0) \in R^{N\times N}$ is the Jacobian of $f(\mathbf{x})$ evaluated at $\mathbf{x} = \mathbf{x_0}$ and $B_0 = B(\mathbf{x_0})$. The output equation remains unchanged and is therefore omitted. This equation gives a good approximation of the system dynamics as long as the state, $\mathbf{x}$, remains sufficiently close to $\mathbf{x_0}$.

In order to allow the state to deviate further from $\mathbf{x_0}$ while still maintaining system

accuracy, more linear models can be placed throughout the state space. Suppose *s* linear models are created. We can approximate the dynamics of the system at state **x** by using the following equation which evaluates a weighted combination of all of the linear models.

$$\frac{d\mathbf{x}}{dt} = \sum_{i=0}^{s-1} w_i(\mathbf{x})(f(\mathbf{x_i}) + A_i(\mathbf{x} - \mathbf{x_i}) + B_i u) \quad (3)$$

where the $w_i(\mathbf{x})$s are scalar weights of the current state vector, **x**, for each linear model. In this thesis, it is assumed that the sum of all $w_i(\mathbf{x})$s is equal to 1.

The choice of weighting is important. The formulation of the weights should be so that linear models far from the current state are negligible and the closest linear models are weighed more favorably. In this thesis, the weights are determined by the following procedure [8]:

1) For i = 0:(s-1), compute $d_i = \|\mathbf{x} - \mathbf{x_i}\|_2$
2) $m = \min_{i=0:(s-1)} d_i$
3) For i = 0:(s-1), compute $\hat{w}_i = e^{-\beta d_i/m}$
4) Normalize the set $\hat{w}_i$ so that $\sum_{i=0}^{s-1} \hat{w}_i = 1$

In general, β is arbitrarily selected depending on how much incorporation of multiple linear models is desired in the TPWL system where a larger β represents less incorporation. In this thesis, we use β=85 to represent less incorporation of neighboring linear models in the results of the TPWL simulation. This weighting scheme allows the weighting to be divided among the linear models based on their relative proximity to the current state of the system. The approximation will be most accurate when the state is sufficiently close to a single linear model but if there are multiple models competing for closeness, then a combination of the models will aid in approximating the nonlinear behavior of the system.

### 2.2 Generating Piecewise-Linear Models

As mentioned above, when the system is sufficiently close to a linear model, the approximation will be most accurate. It then stands to reason that the state space should be fully populated with linear models in a way that there will always be a model nearest to the state regardless of where the state travels during simulation. This would be the most accurate method but a system with N states has an N-dimensional state space. For a small N, this may be feasible,

but as N grows, the number of linear models required in order to populate the N-dimensional state space increases exponentially. Computation of the functions and Jacobians for every linear model would be required to create the TPWL model as well as the computation costs of all of the weighting evaluations at each time step in the simulation. This would be too computationally expensive, so a more resourceful method should be used.

Rather than populating the entire state space with linear models, a better solution would be to populate linear models along a trajectory through the state space that follows the solution of the full nonlinear system. In order to accomplish this, we need to perform an initial simulation of the system with a training input signal. The input signal should be simple while still being of a relevant magnitude to the system.

The training process starts by creating a linear model at the initial state for the system, $\mathbf{x}(t_0)$. The nonlinear system is then simulated using the training input as long as the current state, $\mathbf{x}$, is sufficiently close to the previous linearization point. The system is simulated as long as the following inequality remains true,

$$\frac{\|\mathbf{x}-\mathbf{x_0}\|}{\|\mathbf{x_0}\|} < \delta \tag{4}$$

where $\delta \in R$ is typically between 0.01 and 0.10. In this thesis, we use $\delta = 0.01$ to allow more linear models to be generated for higher accuracy. This operation is repeated until the prescribed number of linearization points are created with linearized models of the nonlinear system.

Once all of the models are created, the size of the models can be reduced. In order to do this, we project the models from the full state space onto a reduced-order state space. To do this, we can use the following variable transformation:

$$\mathbf{x} = V\mathbf{z} \tag{5}$$

where $\mathbf{z} \in R^q$ is a projection of state $\mathbf{x}$ onto a state space of size q where $q < N$, and $V \in R^{N \times q}$ is an orthonormal transformation matrix. The smaller q is with respect to N, the greater the reduction in the system but the amount of reduction must be regulated for the sake of accuracy. The orthonormal transformation matrix, V, is generated using the MGS-Arnoldi algorithm detailed below.

MGS-Arnoldi Algorithm
Input: System matrix A, input vector B, reduced system order q.
Output: Projection matrix V

1) $\mathbf{v_1} = B/\|B\|_2$
2) for i = 1 to q
3) $\quad \mathbf{x} = \mathbf{v_i}$
4) $\quad$ Solve: $A\mathbf{v} = \mathbf{x}$
5) $\quad$ Orthogonalize $\mathbf{v}$
$\quad$ for j = 1 to i
6) $\quad\quad \alpha = \mathbf{v}_j^T\mathbf{v}$
7) $\quad\quad \mathbf{v} := \mathbf{v} - \alpha\mathbf{v}_j$
8) $\quad$ Set $\mathbf{v_i} = \mathbf{v}/\|\mathbf{v}\|_2$
9) Set: $V = [\mathbf{v}_1, \ldots, \mathbf{v}_q]$

Using the transformation (5), we can complete our generation of the TPWL model. By making the substitution (5), we can represent (3) as a weighted sum of reduced order linear models as shown with the following system,

$$\frac{d\mathbf{z}}{dt} = \sum_{i=0}^{n-1} w_i(\mathbf{z})\left(V^T f(\mathbf{x_i}) + V^T A_i V(\mathbf{z} - \mathbf{z_i}) + V^T B_i u\right); \quad y = C^T V\mathbf{z}. \tag{6}$$

This formulation represents a complete trajectory piecewise-linear reduced order model for a simple nonlinear ODE. This model will evolve at each time step when new weights are evaluated, allowing the model to better represent the nonlinear characteristics of the system while still reducing the order of the full nonlinear system accurately.

### 2.3 Faster Simulation Techniques

With a properly constructed the formulation of a TPWL model of the full nonlinear system, the consideration of the number of linear models becomes important to the reduction in computational costs during simulation. The number of linear models, s, has two major impacts on the TPWL model. The first is a matter of storage. A larger s will require more space to store the matrices and information pertaining to each linear model. Storage, however, should not affect simulation costs though. The second is a matter of computations per simulation time step. As s grows larger, the amount of computation needed to evaluate the weighting values increases as well as the total summation for the weighted linear models. Each weighting evaluation computes a two-norm as well as an exponential function. Restricting the amount of computation required to compute the weights needed to simulate each time step of the TPWL model is crucial to

computational efficiency.

Constraining the computational costs associated with the evaluations of the linear model weights could be done by limiting the number of linear models generated. However, this severely limits the capability of TPWL to represent the nonlinearity of the system. For a system that does not reach a steady state value and requires a long simulation time, limiting the number of linear models limits the time accuracy of the TPWL model to only as long as the training simulation was allowed to run. It seems that we need to make a compromise between having a large number of linear models and a small amount of model evaluations per time step. This can be done by using a Nearest Neighbors (NN) approach when evaluating weighting values.

In calculating the weights for each time step of the TPWL model, relatively few linear models are given a large enough weighting value so as to actually contribute to the simulation dynamics. More striking is that even though most of the linear models will be evaluated for a weighting value that is negligible to the dynamics of the reduced order system, the computational costs to determine their contribution is equivalent to the costs of evaluating the weights for the models that drive the dynamics of the system at that time step. The NN approach allows the TPWL model to create as many linear models as it needs to produce accurate results and then evaluates the nearness of each linear model to all other models. For each model, it is then determined which models are most near. These models are referred to as neighbors. In this thesis, five neighbors were found for each model. During the simulation of the TPWL model, the nearest linear model to the current state is tracked and only the nearest model and its neighbors were used in the weighting evaluations. A re-evaluation of the nearest neighbor must be performed after the current state has changed but this is not computationally expensive as it only needs to be compared to the previously nearest model and that model's neighbors. The use of the NN approach in the TPWL MOR method represents a significant reduction in simulation computation at the expense of increased storage.

## 3 Neural Mass Modeling, Results and Discussion

As previously mentioned, the differential equations for neural mass models are nonlinear, strongly coupled and chaotic. By changing the system constants, each subpopulation can represent a specific frequency. The dynamics of a subpopulation are represented by six states and a unit can have multiple subpopulations. Figure 3.1 shows a representation of the dynamical system of a single subpopulation. The figure demonstrates that the equations represent different types of neuron cells within the neural mass. To further understand the system, the system equations are expanded upon below:

$$\dot{x}_1 = x_4$$
$$\dot{x}_2 = x_5$$
$$\dot{x}_3 = x_6$$
$$\dot{x}_4 = \frac{H_e}{t_e} S_3(x_2 - x_3) - \frac{2}{t_e} x_4 - \frac{1}{t_e^2} x_1$$
$$\dot{x}_5 = \frac{H_e}{t_e} (p + S_1(x_3)) - \frac{2}{t_e} x_5 - \frac{1}{t_e^2} x_2$$
$$\dot{x}_6 = \frac{H_i}{t_i} S_2(x_1) - \frac{2}{t_i} x_6 - \frac{1}{t_i^2} x_3$$
$$y = x_2 - x_3$$
$$S_i(v) = \frac{c_i^1 e_0}{1+\exp(r(v_0 - c_i^2 v))}; \quad \textbf{for } i = 1, 2, 3$$

The input to the system, p=p(t), represents an extrinsic input to the NMM model, and is evaluated as a random number normally distributed with a mean of 220 and a standard deviation of 22. The constants $H_{e,i}$ and $t_{e,i}$ are system parameters that determine the frequency of the population. With regards to $H_{e,i}$ and $t_{e,i}$, (e) represents excitatory and (i) represents inhibitory populations of neurons. In the system of equations, the states $x_1$, $x_2$, and $x_3$ are average membrane voltages of their respective neuron cell type as depicted in Figure 3.1. For instance, $x_1$ represents the average membrane voltage of the excitatory pyramidal cells. The symbols $c_i^{1,2}$, r, $v_0$, and $e_0$ are constants used to shape the sigmoid function $S_i(v)$ which transforms the average membrane potential of the neuron cell groups into an average firing rate of the cell groups [3]. Refer to Table A.1 in Appendix A for the values of these constants for this thesis.

Referring to (2), we can begin to observe how the NMM system equations would fit into a TPWL implementation. We can do this by analyzing what B(**x**) and A(**x**) are for the system above. B(**x**) is a very simple matrix, since the input is only in one equation. We can evaluate it

as:

$$B(\mathbf{x}) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \frac{H_e}{t_e} \\ 0 \end{bmatrix}$$

Removing the input $B(\mathbf{x})p(t)$ from the NMM equations gives us the nonlinear vector function $f(\mathbf{x})$. Evaluating the Jacobian of $f(\mathbf{x})$ for the NMM equations yields $A(\mathbf{x})$. $A(\mathbf{x})$ is a 6x6 matrix and lends itself well to a block matrix form as seen below.

$$A(\mathbf{x}) = \begin{bmatrix} 0_{3\times3} & I_{3\times3} \\ A_{21} & A_{22} \end{bmatrix}$$

$$A_{21} = \begin{bmatrix} -\frac{1}{t_e^2} & \frac{H_e}{t_e}\frac{\partial}{\partial x_2}S_3(x_2 - x_3) & \frac{H_e}{t_e}\frac{\partial}{\partial x_3}S_3(x_2 - x_3) \\ 0 & -\frac{1}{t_e^2} & \frac{H_e}{t_e}\frac{\partial}{\partial x_3}S_1(x_3) \\ \frac{H_i}{t_i}\frac{\partial}{\partial x_1}S_2(x_1) & 0 & -\frac{1}{t_i^2} \end{bmatrix}$$

$$A_{22} = \begin{bmatrix} -\frac{2}{t_e} & 0 & 0 \\ 0 & -\frac{2}{t_e} & 0 \\ 0 & 0 & -\frac{2}{t_i} \end{bmatrix}$$

A unit with multiple subpopulations designates weights for each subpopulation and, in doing so, is able to create a unique frequency spectrum. Each subpopulation uses the same state space representation as in Figure 3.1 except that for $S_i(v)$ and $y(t)$ the corresponding states for all subpopulations are summed with respect to their subpopulation's weight. For a single NMM unit with 2 subpopulations with weights $w^1$ and $w^2$ corresponding to subpopulation 1 and subpopulation 2, respectively, the sigmoidal function (7) and output would take the form:

$$y(t) = \sum_{i=1}^{2} w^i\left(x_2^i - x_3^i\right). \tag{7}$$

For a large network of neural mass units, this can result in a large number of states. A large number of states can be handled with the TPWL model order reduction method as mentioned above. However, the biggest challenge is dealing with the chaotic nature of the neural mass dynamics.

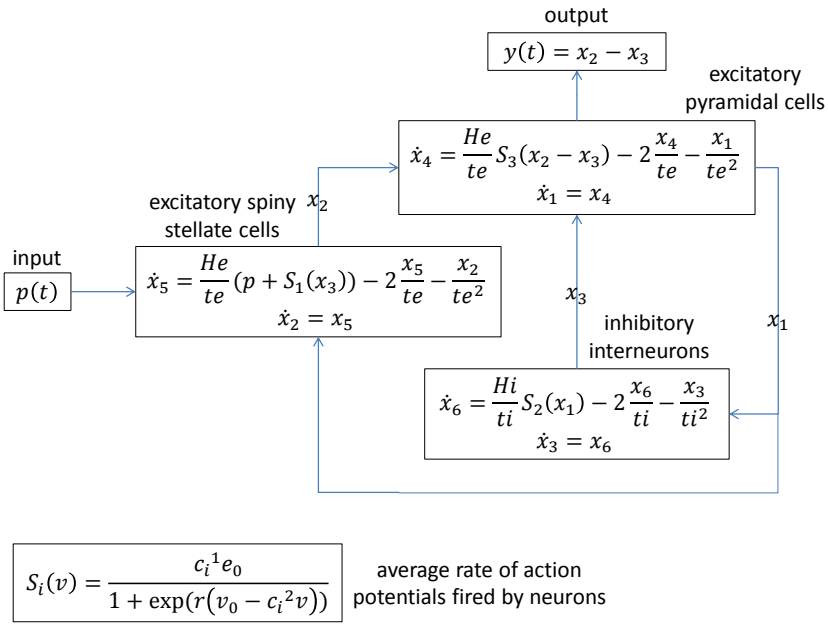


Figure 3.1. State-space representation of a neural mass model with one subpopulation

With the general system matrices created for a TPWL implementation, the TPWL generation process can begin by selecting an initial state and evaluating f(**x**), A(**x**), and B(**x**) at the initial state. It should be noted though that for NMM equations, B(**x**) will never change. The system can now be simulated using a training input in order to generate further linear models as detailed in Chapter 2. NMM is a complicated system, however, and using a training input that doesn't incorporate the randomness of the actual NMM input, p(t), results in no dynamic response. For this reason, the training input for the TPWL generation should remain the same as the input for the full nonlinear system.

For a system to easily work well with TPWL, it should have a distinctive trajectory for the training input simulation to follow. Figure 3.2 shows a parametric view of the six states of a 10 Hz neural mass unit. By graphing every possible pair of states, we can see which state interactions result in a cycle-like behavior that would allow TPWL to work efficiently. States $x_1$, $x_3$, $x_4$, and $x_6$ show cycle-like trajectories when paired with each other. However, states $x_2$ and $x_5$ are driven by the input to the system. Because the input, p(t), is a normally distributed random number, the two states driven by the random input are chaotic and appear random as well. This will require the TPWL model generation phase to create many more linear models to represent this random space.

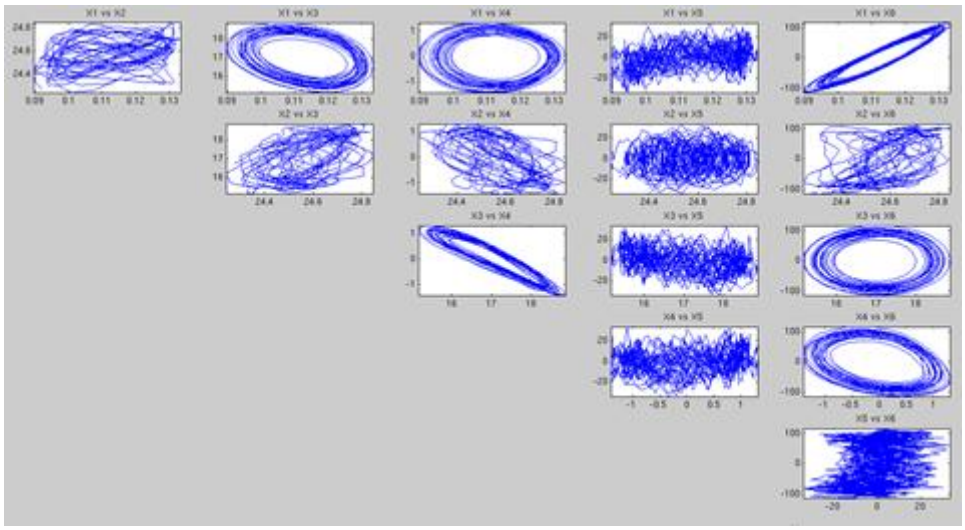

Figure 3.2. A parametric view of the states of a 10 Hz neural mass unit. This shows the states that behave in a limit cycle like manner and those that do not. The plots of a state with respect to itself are omitted

In order to test the chaotic dynamics of the neural mass model equations, this thesis simplifies the overall system while still creating a complex system. To begin with, only a single NMM unit unconnected with other units is tested. This keeps the system a SISO system rather than a SIMO or MIMO system. Additionally, only a single frequency will be tested. In this case, a 10 Hz NMM unit will be used in the simulations for the full nonlinear and TPWL models. The reason for this is to isolate the dynamics of the equations rather than being distracted by complications in combining frequencies within a single unit. In order to create a single unit with a single frequency that has many states, the unit will be created with multiple subpopulations with the same constants and same initial conditions. This system will have dynamics identical to a unit with a single 10 Hz subpopulation, but the number of states will be six times the number of multiple subpopulations and will thus be sufficient for evaluating the effectiveness of the TPWL method.

For simplicity, a system with only two subpopulations is examined. As mentioned above, starting both subpopulations at the same state will ensure that the system behaves similarly to a unit with a single 10 Hz subpopulation but with twice as many states and will allow us to see how well the reduction algorithm works while still being able to identify whether or not the system is behaving accurately. In order to simulate the full nonlinear system and the TPWL system accurately with the same ODE solvers, a set of open source MATLAB codes called STPWL [12] was modified for the implementation of the NMM system.

The first aspect of model order reduction we looked at was how much of the state space

could be reduced accurately. We tested the system over a small time scale and with different orders of reduction. The system initially begins with 12 states. Figure 3.3 shows the results of reducing the size of the state space in even numbers. Each graph in the figure shows 3 sub-plots. The top plot is a time series plot of the output of the full nonlinear system (blue) and the reduced system (red), the bottom left is a plot of the percent error between the nonlinear and reduced system with respect to time, and the bottom right is a plot of the full nonlinear output with respect to the reduced model output where a straight line corresponds to good accuracy. The reduction of the system to 6 states is the best result as it demonstrates a stable reduced order model with relative error below 1%. This demonstrates that it should be realistic to reduce the size of the system by at least half of the number of states and still maintain good accuracy and stability.

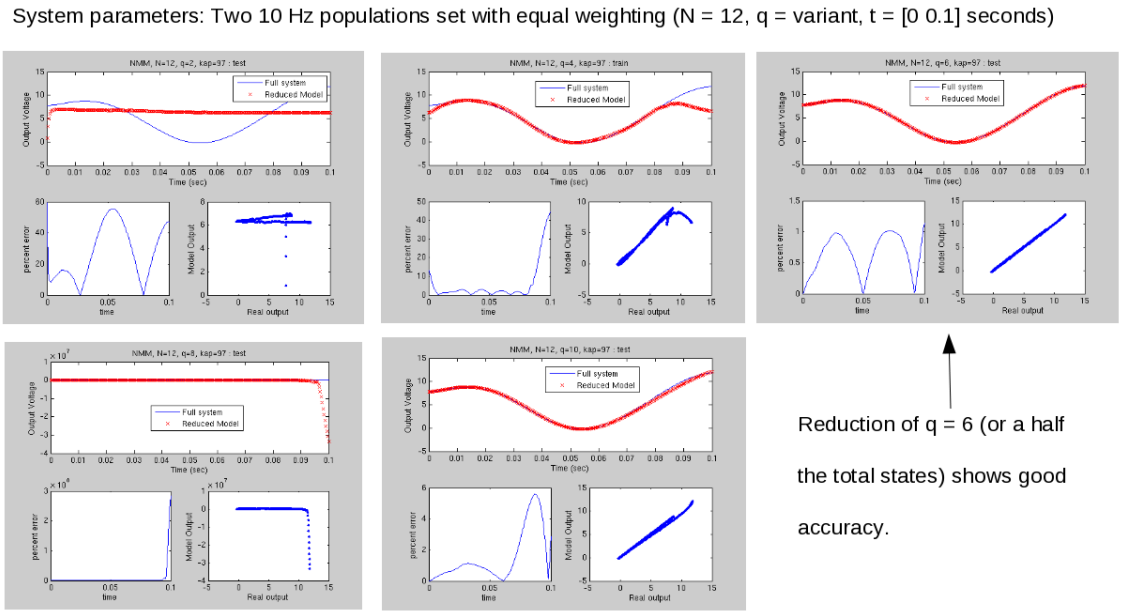


Figure 3.3. Varying size of the state space to test reduction accuracy in the system. From the top left to right, the reduced state sizes are 2, 4, 6, 8, and 10

Accuracy for a NMM system is a matter of frequency rather than exact positional accuracy. The frequency of the system is far more important and the shape of the plot is a good indicator of his. For this reason, we shall assume that anything with relative error less than 5% is good accuracy for the TPWL reduced order model.

The computational efficiency of the TPWL method can be seen by tracking the computation times for the full nonlinear system and the TPWL reduced order system with respect to the amount of simulation time. The NMM system was simulated 5 times for varying amounts of simulation time. The amount of time it took to compute the full nonlinear and TPWL systems

was averaged and the results are shown in Figure 3.4.

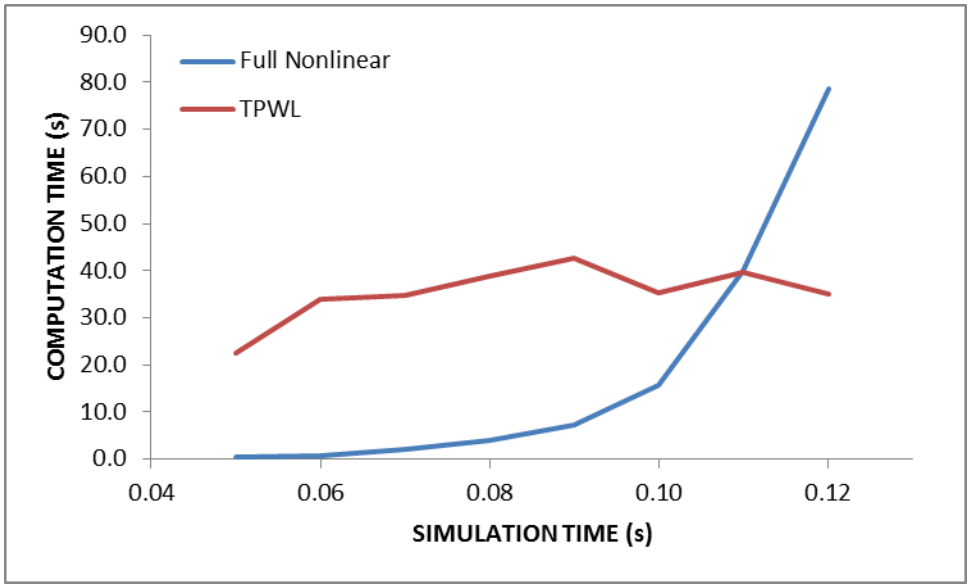


Figure 3.4. Comparison of computational times of the full nonlinear (blue) and TPWL (red) systems with respect to the amount of simulation time for the model

At very low simulation times of 0.05 seconds, the full nonlinear system is quickly solved in an average of 0.5 seconds while the TPWL system is solved in an average of 22.5 seconds. As the simulation time increases, however, the TPWL computation time increases relatively slowly while the full nonlinear system increases rapidly and exponentially. The TPWL simulation times seem to stabilize around 39 seconds regardless of the amount of simulation time. By the simulation time of 0.11 seconds, the full nonlinear model takes 40.0 seconds to compute while the TPWL system takes only 39.7 seconds. In the same amount of time it took the TPWL computation time to increase by 76.4%, the full nonlinear computation time increased by 7900%. It should be noted that as the simulation time increased, the computational time using STPWL continued to increase in the same manner for the full nonlinear system and so the larger simulation times were not used for comparison since the discrepancy was so great.

The TPWL method has shown that the computation times can be reduced for a neural mass model simulation. It has been shown that the size of the state space can be reduced by half without significant loss to the accuracy of the system. The size of the reduction is a significant result, but it is meaningless without some evidence of computational efficiency to back it up. This is why the results showing that after 0.11 seconds of simulation time it becomes more efficient to use the TPWL reduced model to simulate the system rather than the full nonlinear

model are important to the validation of the TPWL's use for NMM simulation. The toughest problem encountered with TPWL is the fact that simulating the full nonlinear system is currently necessary for the creation of the TPWL reduced order model. This makes simulating the TPWL model with simulation times much greater than 0.11 seconds difficult. This is due to the cost of training the TPWL model. Once the TPWL generation is achieved, however, it has been shown that the simulation of the reduced order model will be less computationally expensive than simulating the full nonlinear system.

## 4 Conclusions and Future Work

In this thesis, we have introduced the need for a reduced computational cost in brain modeling simulations and the shortcomings that some model order reduction methods will have when dealing with nonlinear, coupled differential equations. The trajectory piecewise-linear model order reduction method was detailed by demonstrating how a nonlinear system can be represented as piecewise-linear. Using a training input to simulate the full nonlinear system in order populate the state space with reduced linear models shows how to create the TPWL model for any ordinary differential equation and the implementation of Nearest Neighbors during simulation further reduces the computational power needed to simulate the system. The complexities of the neural mass model were demonstrated by observing the equations and looking at the interactions between states. Finally, this thesis shows that the NMM example used here is capable of reducing the state space size in half and using the TPWL method will see improved computational efficiency when simulating the system beyond 0.11 seconds.

The two major aspects that should be addressed in future work for the TPWL implementation of NMM systems are accuracy and training time. Accuracy is crucial to reduced order modeling success. Without an accurate result, a reduced order model is worthless no matter how much computational cost is reduced. To this point, we note that there is an issue with the original TPWL system and it comes from (4), the inequality used to determine when a new linear model should be added during the TPWL training phase. As you can see from Figure 3.2, the six states that drive the dynamics of the 10 Hz NMM unit vary in magnitude. One state varies +/- 100 while another varies only +/- 1. Neither state's variation is more important than the other's with respect to the accuracy and dynamics of the system. However, with regard to (4), the state which varies +/- 100 is far more important to the training phase. The inequality (4) determines that a new model should be created when its relative state change has exceeded a certain value. The change in the +/- 100 state has a greater impact on the relative state change than the +/- 1 state does. This can be a problem if the smaller state's oscillating does not exceed the inequality (4) even though the oscillating is important to driving the system output. In future work, it will be important to implement a scaling factor into (4) that will allow large relative changes within a single state to be more important regardless of the magnitude of that state. Incorporating the importance of variations in small magnitude states will result in a need for more linear reduced models in the TPWL system, but this is acceptable as long as a Nearest Neighbors approach is used for minimizing the number of linear models used per time step. This change in (4) will need to be observed as a function of TPWL accuracy.

Training time is very important to the computational expediency of using a TPWL implementation for NMM systems. If it takes too long to create a TPWL model, then many may choose to use to forgo MOR and just simulate the full nonlinear system anyway. With STPWL, we are able to observe the efficiencies of MOR when the full nonlinear system and the reduced order model are run using the same ODE solver. This is a great comparison and the results are easily observable. However, we also observe the enormous increase in computational time associated with simulating the full nonlinear model. Since the full nonlinear model is still simulated once to create the TPWL model, the computational time to simulate the full nonlinear system is directly tied to the efficiency of creating the TPWL model. There may be more efficient ways of simulating the NMM systems, though. When creating a tool for simulating reduced order models of NMM systems for researchers to use, it may be better to first run the TPWL training simulation using the more efficient NMM simulation methods and extracting the linear models from the record of the state space as determined by the TPWL training parameters.

This thesis demonstrates a significant improvement in reduction in computational times for the implementation of NMM systems using a TPWL approach for model order reduction. The ability to reduce the size of the state space and implement the NMM dynamic equations as a weighted sum of reduced linear models has resulted in a significant improvement in the amount of time it takes to compute the NMM simulations. We've shown that after a simulation time of 0.11 seconds, the TPWL reduced order model is faster and more efficient while maintaining a steady computation time even while the full nonlinear system's computation time continues to grow with the addition of more time steps in the simulation. Moving forward, addressing issues of improved accuracy and decreased computation costs in the TPWL generation phase will promote the use of TPWL as a preferred choice for researchers using NMM systems

**References**


[1] O. Sporns, G. Tononi, R. Kötter, "The human connectome: A structural description of the human brain," *PloS Comput. Biol.* vol 1, no. 4, pp. e42, Sep. 2005.

[2] J. Vettel, A. Dagro, et al. "Brain structure-function couplings (FY11)," Army Research Lab HRED, Aberdeen Proving Grounds, MD, Tech. Rep. ADA556969, Jan. 2012.

[3] O. David and K. J. Friston, "A neural mass model for MEG/EEG: coupling and neuronal dynamics," *NeuroImage*, vol. 20, no. 3, pp. 1743-1755, Jul. 2003.

[4] W. Schilders, H. van der Vorst, J. Rommes, *Model Order Reduction: Theory, Research Aspects and Applications*, Berlin, Germany:Springer, 2008.

[5] C. de Villemagne, R. E. Skelton, "Model reduction using a projection formulation," *Int. J. of Control*, vol. 46, pp. 2141-69, Feb. 1987.

[6] E. J. Grimme, "Krylov projection methods for model reduction," Ph.D. dissertation, Dept. Elect. Eng., Univ. of Ill. at Urb.-Cham., Urbana, IL, 1997.

[7] A. A. Mohammad, J. A. De Abreu-Garcia, "A transformation approach for model order reduction of nonlinear systems," in *Proc. 16th Annu. Conf. IEEE Industrial Electronics Society*, Pacific Grove, CA, 1990, pp. 380-383.

[8] M. Rewienski, "A trajectory piecewise-linear approach to model order reduction of nonlinear dynamical systems," Ph.D. dissertation, Dept. Elect. Eng., Mass. Inst. of Tech., Cambridge, MA, 2003.

[9] Y. Chen, "Model order reduction for nonlinear systems," M.S. thesis, Dept. Math., Mass. Inst. of Tech., Cambridge, MA, 1999.

[10] Y. Chen, J. White, "A quadratic method for nonlinear model order reduction," in *Proc. Int. Conf. Modeling and Simulation Microsystems*, San Diego, CA, 2000, pp. 477-480.

[11] J. Chen, S. M. Kang, "An algorithm for automatic model-order reduction of nonlinear MEMS devices," in *Proc. IEEE Int. Symp. Circuits and Systems*, Geneva, 2000, pp. 445-448.

[12] B. N. Bond. (2010, March 31) *STPWL: A MATLAB tool for Stable Trajectory PieceWise Linear model reduction* [Online]. Available: http://bnbond.com/software/stpwl. Date Last Accessed, 11/04/2012.

**Appendix A: Neural Mass Model Constants**

| Constant | Value and units |
|---|---|
| $H_e$ | 3.25 mV |
| $H_i$ | 22 mV |
| $t_e$ | 10 ms |
| $t_i$ | 20 ms |
| $c_1^1$ | 135 |
| $c_1^2$ | 108 |
| $c_2^1$ | 33.75 |
| $c_2^2$ | 33.75 |
| $c_3^1$ | 1 |
| $c_3^2$ | 1 |
| $v_0$ | 6 mV |
| $e_0$ | 5 $s^{-1}$ |
| r | 0.56 $mV^{-1}$ |

Table A.1. Values used for NMM Constants