Stochastic Neural Mass Modeling of a Theta Neuron Network

AMATH 422

John Kruper & Maxwell Weil

12/10/19

INTRODUCTION

Within the field of computational neuroscience, researchers strive to develop techniques for modeling a variety of diverse brain activities. From individual synapses to large-scale brain waves, thousands of equations and methods have been used to work towards this goal. With recent advances in imaging and recording technology, models first made over 30 years ago have been refined to better describe some of the delicate intricacies of the brain. One interesting application of these models was first done by Hugh Wilson and Jack Cowan in 1972, where these researchers attempted to describe a simple system of interacting excitatory and inhibitory neurons [1]. In this work, Wilson and Cowan were able to illustrate how neurons can reach stable, oscillating states using simple mathematical models. This discovery provided a foundation for characterizing well-known rhythmic brain behaviors seen through EEG. This, in turn, paved the way for what are now known as neural mass models. Neural mass models aim to explain how phenomena like neural oscillations occur, by analyzing how populations of connected neurons evolve throughout time. This is often done by investigating dense, spatially continuous neural networks, that is, networks with many neurons that are all interconnected. However, computing such a network is often complex and time-consuming. Assuming that synaptic connections only flow in one direction, a network of only 100 neurons would have 9900 connections to account for (N*(N-1)). To alleviate this issue, researchers opted to simplify

models by averaging network characteristic (firing rate, connectivity, etc.) over the whole population. This reduces the overall degrees of freedom of the system, allowing for a streamlined model. This approach, known as mean field reduction, has been shown to provide accurate averaged models for complex systems. Recently, Stephen Coombes and Áine Byrne used the mean field reduction method to represent an interconnected network of 500 theta neurons [2]. Our work aims to build upon this research by examining how noise of individual neurons may be added back into the system, without heavy computation. But to best understand how this model works, a brief explanation of the theta model is necessary

The theta neuron model created by the work of George Ermentrout and Nancy Kopell, which was looking at characterizing bursting activity found in the ganglion cells of *Aplysia* sea slugs [3]. Bursting is when a neuron fires rapidly and then rests in a steady manner. This phenomenon is thought to be associated with robust information transfer and network wide firing patterns, though studies are ongoing. In the theta model, the authors used a stable limit cycle to model the rhythmic behavior seen in vivo. The equation describing this behavior is given by:

$$\textbf{(eq. 1)} \quad \frac{d}{dt}\theta = 1 - cos\ \theta + (1 + cos\ \theta)I(t)$$

Where I(t) is the driving current and θ lies along a unit circle and is proportional to the voltage of the neuron. Whenever $\theta = \pi$, the neuron is said to spike. Changing I(t) in this model alters the movement of θ by changing the presence of stable and unstable nodes along the limit cycle. For $I(t) < 0$, a stable and unstable node appear, representing a resting voltage and threshold voltage, respectively. At $I(t) = 0$, the nodes collide, creating a saddle node bifurcation along the limit cycle. And finally, when $I(t) > 0$, there are no nodes, resulting in consistent movement of theta in

counterclockwise fashion. This behavior results in bursting as I(t) changes, as seen below in Figure 1.

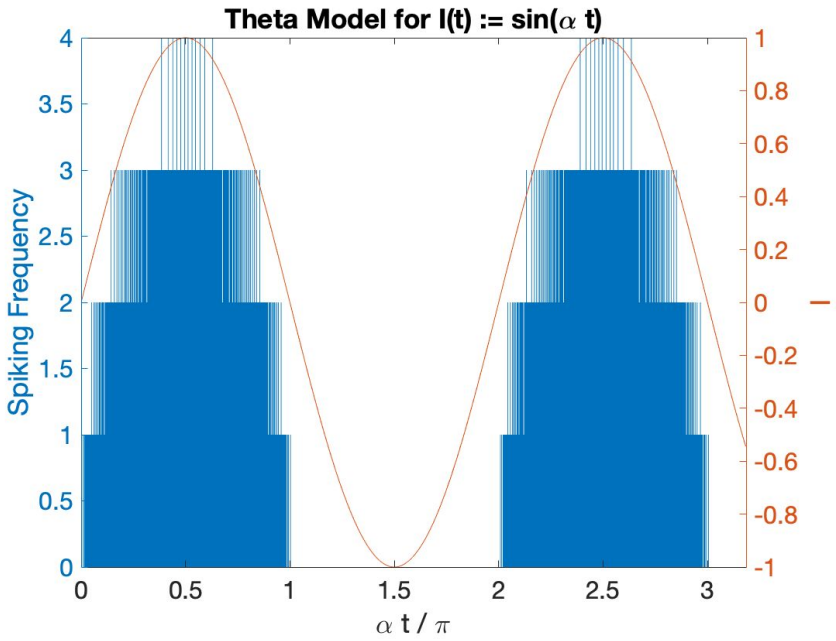

Fig. 1. Spiking frequency (shown in blue) and sinusoidal driving current (shown in orange) over time for a neuron using the theta model. The sinusoidal current causes the neuron to spike in bursts. When the current is less than 0, the spiking frequency is 0. There is a bifurcation when the current is 0. When the current is greater than 0, the neuron can spike. The spiking frequency increases as the current increases once the current is greater than 0.

With this theta model, modeling a system of rhythmically active neurons become quite simple, as Coombes and Byrne demonstrated [2]. However, our project aims to explore just how well a reduced model can represent a fully interconnected network, by reproducing some of the noise seen in individual neurons, but without sacrificing simplicity. To do this, we first detail and reproduce the work of Coombes and Byrne, on which our model is based. Next, a new stochastic model is designed and simulated. Finally, we will take a look at how these models stack up against one another, and whether stochasticity can partially capture the complex interactions of a

full network. Ultimately, we hope that our results can help demonstrate the importance of stochasticity and dynamical model to neuroscience as a whole.

MODEL EQUATIONS

In their paper, Stephen Coombes and Áine Byrne question whether a neural mass model could be created to represent a network of theta neurons. To first approach this mathematical query, the authors had to define what exactly their theta neuron network would look like. To accomplish this, the same equation from above was used to model each theta neuron, but the I(t) was set to depend on the inputs from all other connected neurons (for simplicity, all neurons were fully connected). The synaptic input current for any given neuron *j* is then given by:

$$\textbf{(eq. 2)} \quad I_j(t) = g(t)(v_{syn} - v_j) + \eta_j$$

Where g(t) is the conductance, v_{syn} is the membrane reversal potential (0 for excitatory neurons), v_j is the voltage, and η_j is the constant drive of the neuron. This equation can now be combined with the original theta neuron model from equation 1. Expanding this, we get:

$$\textbf{(eq. 3)} \quad \frac{d}{dt}\theta_j = (1 - cos\ \theta_j) + (1 + cos\ \theta_j)(\eta_j + g(t)v_{syn}) - g(t)sin\ \theta_j$$

As a model equation to represent the activity of each neuron in a network of theta neurons. We now have to quantify how the synaptic conductance changes as a function of the firing rate.

The function $s(t) = \alpha^2 t e^{-\alpha t}\Theta(t)$ (α being a scaling parameter and Θ being the Heaviside function) is a standard choice to match the shape of a realistic post-synaptic conductance. If we consider the conductance change as a result of a series of action potentials, consider s(t) as Green's function of a linear differential operator, and assume a neuron is typically close to rest, we can solve for this equation as in [2]:

(eq. 4) $$Qg = f(g)$$

With f(g) as the population firing rate given the conductance, κ as the coupling strength, g is the conductance, and Q = δ/s. is (this is a standard choice for s). Given s(t), Q can also be solved for:

(eq. 5) $$Q = (1 + \frac{1}{\alpha}\frac{d}{dt})^2$$

For a series of firing times, the population firing rate on the right hand side of eq. 4 is equal to the sum of firing times averaged over all neurons, multiplied by a constant:

(eq. 6) $$Qg(t) = \frac{\pi}{N}\sum_{j=1}^{N}\sum_{m\in Z}\delta(t - T_j^m)$$

Where T_j^m is the *m*th firing time of the *j*th neuron. We will use equations 3, 5, and 6 to model 500 theta neurons. Equation 3 describes the state of each of the 500 neurons, and equations 5 and 6 describe the synaptic conductance. The frequencies of the neurons (η_j in eq. 3) are drawn from the Lorentzian distribution:

(eq. 7) $$L(\eta) = \frac{1}{\pi}\frac{\Delta}{(\eta-\eta_0)^2+\Delta^2}$$

Following the steps to reach the mean field reduction of equations 3 and 6 in [2], a set of equations can be reached to explain the overall dynamics of the system. However, and important element introduced into these equations is the Kuramoto order parameter, Z, which represents the average phase and synchrony of firing in the network. The Kuramoto order parameter is given in equation 8 as:

(eq. 8) $$Z(t) = \int_0^{2\pi} d\theta \int_{-\infty}^{\infty} (\eta, \theta, t)e^{i\theta}d\eta$$

Where $\varrho(\eta,\theta,t)$ represents the continuous probability distribution function, and η is a constant driving mechanism, equivalent to I(t) in equation 1. Using this, we can then define the final equations for the mean field reduced model:

$$\textbf{(eq. 9)} \quad \frac{d}{dt}Z = \mathcal{F} +$$

$$\textbf{(eq. 10)} \quad \mathcal{F} = -i\frac{(Z-1)^2}{2} + \frac{(Z+1)^2}{2}(-\Delta + i\eta_0)$$

$$\textbf{(eq. 11)} \quad = i\frac{(Z+1)^2}{2}v_{syn}g + \frac{Z^2-1}{2}g$$

$$\textbf{(eq. 12)} \quad Qg = f(Z)$$

$$\textbf{(eq. 13)} \quad f(Z) = \frac{1-|Z|^2}{1+Z+\bar{Z}+|Z|^2}$$

While apparently complex, these equations are relatively simple to break down into understandable pieces. Equation 9 is used to describe the dynamics of Z, the phase and synchrony of the network. Equations 10 and 11 are extensions of this, to better divide up the parameters. Note that Δ and η_0 are taken from the Lorenzian distribution in equation 7, where η_0 is the center of the distribution, and Δ is the width of the curve at half of the maximum. One can also see that equation 12 is the same form as equation 4, except with the introduction of the Kuramoto order parameter. Equation 13 still represents the firing rate of the network, as in equation 4, and is composed of the real and imaginary (Z bar) components of Z. Q is the same as it appears in equation 5, giving us a system of ODEs for g and Z, coupled by the coupling parameter κ.

Reproducing the above equations, both a network of 500 theta neurons and the reduced mean field model were successfully recreated in MATLAB. We will detail our methods for computing this simulation, and explain our findings that reflect those seen in the original work.

REPRODUCTION OF RESULTS

We simulated the mean field reduced model described in equations 9-13 with Q defined in equation 5. We used used constants $\eta_0 = 20$, $\Delta = 0.5$, $v_{syn} = -10$, $\kappa = 1$, $\alpha = 0.95$. The paper did not specify the initial conditions or the time [2]. We chose initial conditions $g_{initial} = 1.65$, $dg/dt_{initial} = 0.5$, $Z_{initial} = 0$ and time from 0 to 100 . We chose the initial conditions based off of what appears to be the initial conditions in figure 1 of [2]. For time, we chose a range that is long enough for all models to reach equilibrium. We used Euler's method with a time step of 0.001. We tuned the time and time steps to those values with guess and check to try to match figure 1 in [2]. We simulated the network of 500 θ-neurons using equations 3 and 6. We used the same Q, constants, initial conditions, time, and time steps. We sampled the η for all neurons from equation 7. We initialized the theta neurons by sampling from a uniform distribution of random numbers from 0 to 2π. As in the paper, we do not specify any units for the simulations. The results of the simulations of the two models for parameters Z and g are shown in figure 2.

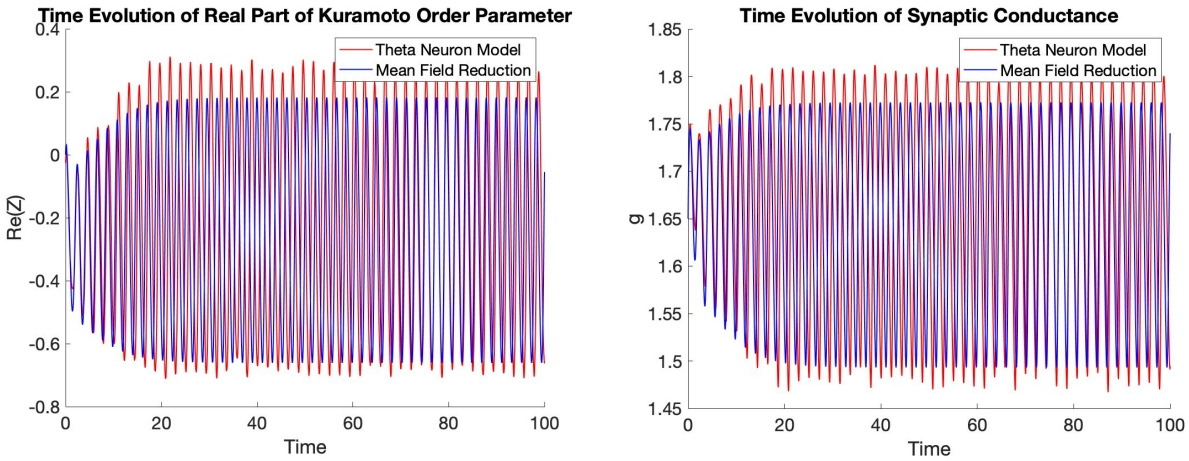

Fig. 2. Left: Variation in real part of the Kuramoto order parameter (Z) changes over the simulation time. Right: Variation the synaptic conductance (g). The reduced mean field model is in blue and a simulation of a

network of 500 θ-neurons is in red. Notice that all parameters quickly converge to a stable oscillation. Notice the network of 500 θ-neurons has more variations than the mean field reduction. Models use constants $\eta_0 = 20$, $\Delta = 0.5$, $v_{syn} = -10$, $\kappa = 1$, $\alpha = 0.95$. For the network simulation, we define $Z = N^{-1} \sum_{j=1}^{N} e^{i\theta_j}$.

Both models reach a stable oscillation by around time 20. Once the models are in this stable oscillation, they agree well. However, the theta neuron model seems to oscillate higher and with more stochasticity that is not captured by the reduced mean field model. To see the limit cycles these models reach, we plot phase plane projections in figure 3.

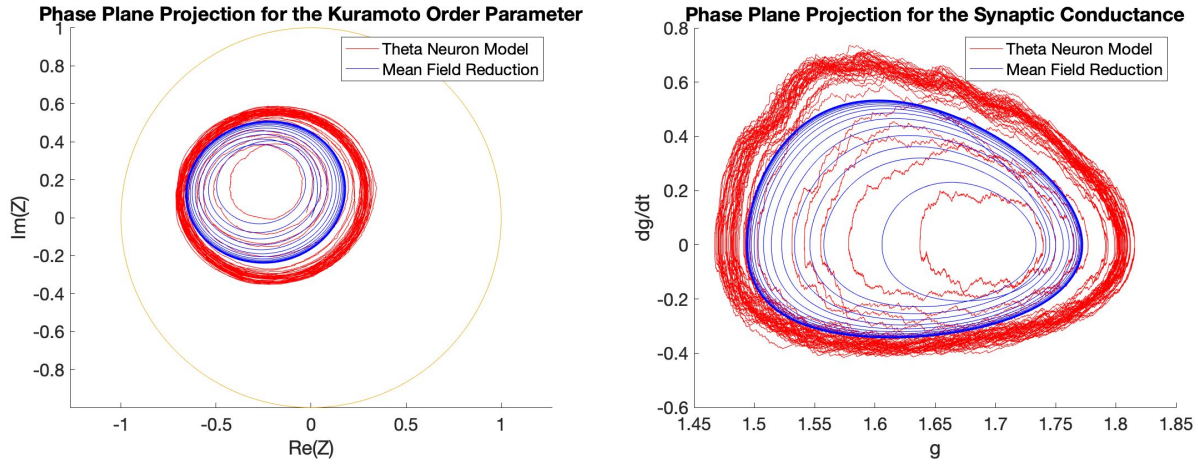

Fig. 3. Left: Phase plane projection of the real and imaginary part of the Kuramoto order parameter (Z). The orange line represents Z values with a magnitude of 1. Right: Phase plane projection of the synaptic conductance (g) and its derivative with respect to time (dg/dt). The reduced mean field model is in blue and a simulation of a network of 500 θ-neurons is in red. Notice the network of 500 θ-neurons has more variations than the mean field reduction and converges to a wider oscillation.

The theta neuron model has wider oscillations for this simulation. Notice the Z parameter for the theta neuron model (red and on the left) makes more smooth oscillations than the g parameter for the same model (red and on the right). Both models circle outward towards their eventual oscillation. To see this clearly, we make the same plot, but we only plot for $t > 20$ in figure 4.

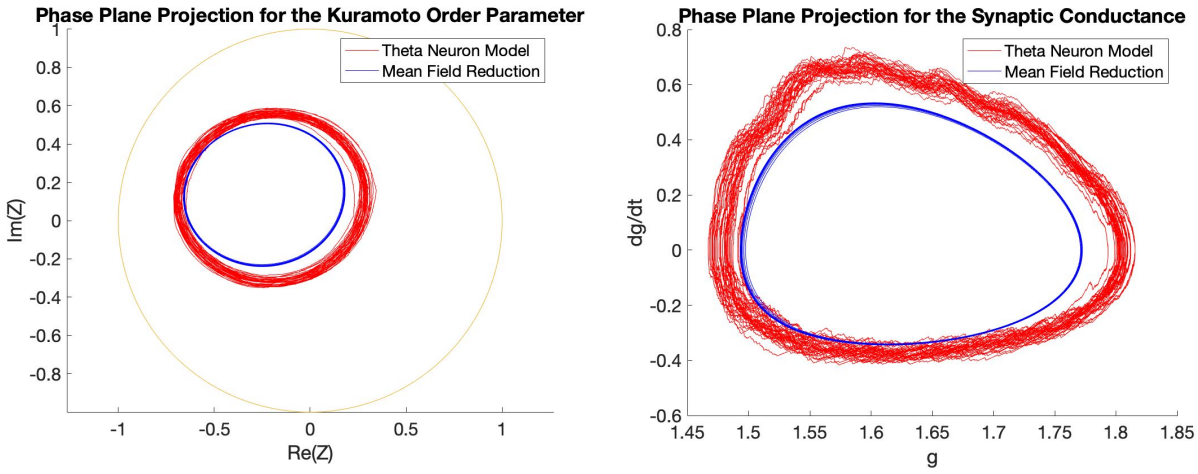

Fig. 4. Left: Phase plane projection of the real and imaginary part of the Kuramoto order parameter (Z). The orange line represents Z values with a magnitude of 1. Right: Phase plane projection of the synaptic conductance (g) and its derivative with respect to time (dg/dt). The reduced mean field model is in blue and a simulation of a network of 500 θ-neurons is in red. Plot is the same as fig.3 but with only t > 20. Notice the only difference between this figure and figure 3 is that there are no longer initial outward oscillations.

In figure 4 we can see the same phenomenon as in figure 3, except more clearly because there is no longer initial outward oscillations. Figure 4 is a reproduction of figure 1 in [2] that is the focus of this section. However, in figure 1 of [2], the theta neuron model and reduced mean field model seem to agree more closely. Additionally, the limit cycle of the theta neuron model is smaller than the limit cycle of the reduced mean field model in [2] but is larger in our plot. We ran the simulation multiple times and found that the exact limit cycle the theta neuron model finds can vary, shown in figure 5.

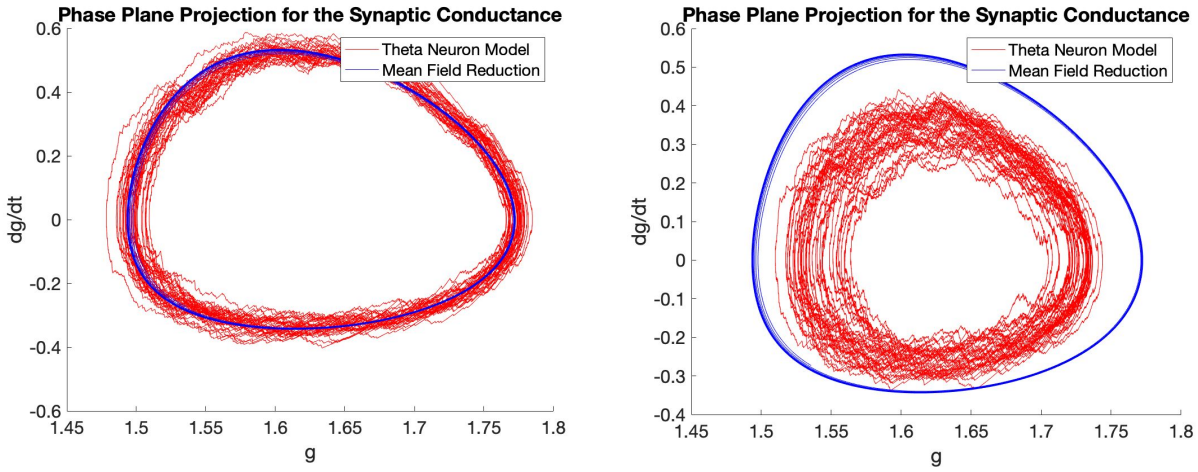

Fig. 5. Phase plane projection of the synaptic conductance (g) and its derivative with respect to time (dg/dt). The reduced mean field model is in blue and a simulation of a network of 500 θ-neurons is in red. On the left and right we see two representative results of running the simulation multiple times. Comparing these two plots and the right side of figure 4, notice that (1) the limit cycle of the reduced mean field model does not vary significantly from simulation to simulation and (2) the limit cycle of the theta neuron model does.

Remember that, before simulating the theta neuron model, we determine the frequencies its constituent neurons by sampling from the Lorentzian distribution described in equation 7. If we keep the same frequencies and run the simulation again, the results do not vary. Thus, the disagreement between our figure 4 and figure 1 in [2] is likely due primarily to differences in the sampled frequencies.

Before we explored novel results, we wanted to understand the effect of the Kuramoto order parameter more intuitively. This parameter is meant to capture the synchrony of the 500 theta neurons. In figure 6, we overlay the phase plane projection of Z for one of our simulations on top of the right hand side of equation 12.

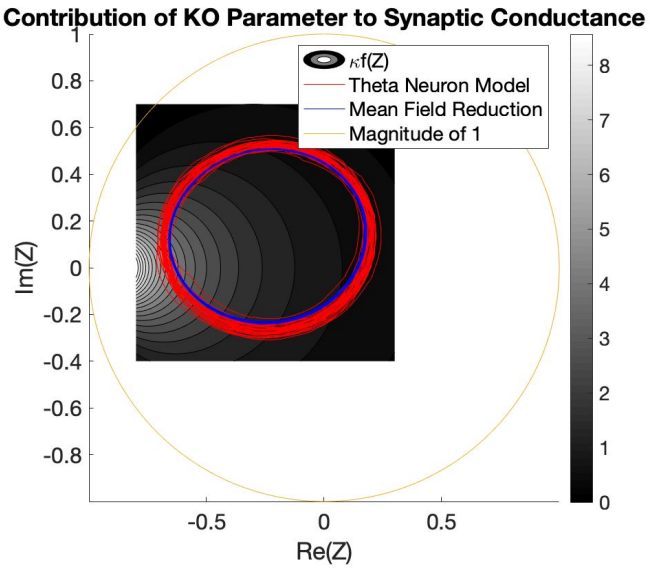

Fig. 6. Phase plane projection of the real and imaginary part of the Kuramoto order parameter (Z). It is overlaid on a grayscale colormap of $\kappa f(Z)$, the right hand side of equation 12. The orange line represents Z values with a magnitude of 1. The reduced mean field model is in blue and a simulation of a network of 500 θ-neurons is in red. Only $t > 20$ is shown. Notice when Re(Z) is low and Im(Z) is close to 0, f(Z) takes on large values.

From figure 6 we see $\kappa f(Z)$ is large when Re(Z) is low and Im(Z) is close to 0. According to equation 12, larger values of $\kappa f(Z)$ contribute positively to changes in g. We can think of large values of $\kappa f(Z)$ as representing the 500 θ-neurons firing in sync, and small values of $\kappa f(Z)$ as representing the neurons firing out of sync. Thus, oscillations in the real and imaginary components of Z represent the 500 θ-neurons falling in and out of sync.

NOVEL RESULTS

The work done by Coombes and Byrne helps elucidate the general dynamics of complex oscillatory neuronal systems with a simple set of equations. However, we felt that this model could be improved upon by maintaining some of the noise seen in the original 500 neuron population. From figure 4 we see that noise is one of the main differences between the reduced

mean field model and the 500 θ-neuron model. While re-adding this noise may initially seem counterintuitive, it could help create a system with more realistic, less deterministic dynamics that we see in simulated neuron populations, while preserving the ease of computation. Our choice to add noise to the synaptic conductance term was aimed at mimicking the contribution of single neurons to the whole network. Synaptic noise is a common phenomenon, caused by a variety of factors including improper calcium or neurotransmitter release [4]. The result of adding this noise is shown in figure 7.

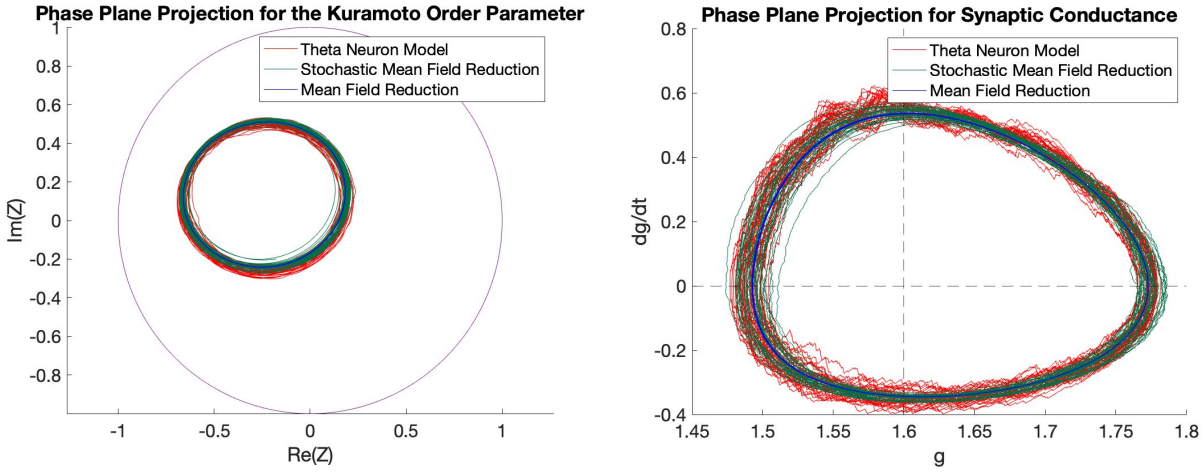

Fig. 7. Left: Phase plane projection of the real and imaginary part of the Kuramoto order parameter (Z). The purple line represents Z values with a magnitude of 1. Right: Phase plane projection of the synaptic conductance (g) and its derivative with respect to time (dg/dt). Dotted lines represent g = 1.6 and dg/dt = 0. The reduced mean field model is in blue, the same model with stochasticity added to the synaptic conductance is in green, and a simulation of a network of 500 θ-neurons is in red. Only t > 20 shown. Models use constants $\eta_0 = 20$, $\Delta = 0.5$, $v_{syn} = -10$, $\kappa = 1$, $\alpha = 0.95$. For the network simulation, we define $Z = N^{-1} \sum_{j=1}^{N} e^{i\theta_j}$. Notice that the stochastic model attempts to capture the variation in the 500 θ-neuron model that the standard mean field reduction does not. Noise was added to the synaptic conductance with $\varepsilon = 0.007$.

We see the reduced mean field model better matches the theta neuron model when stochasticity is added. We used the Euler–Maruyama method with $\varepsilon = 0.007$ on the synaptic conductance. In order to choose ε, we looked at cross sections of the right side of figure 7. We

chose the right side as there is more variation in that plot. We chose the four easiest cross sections as defined by the dotted lines. As the models cross these dotted lines, we can store their values in a histogram. These histograms are shown in figure 8.

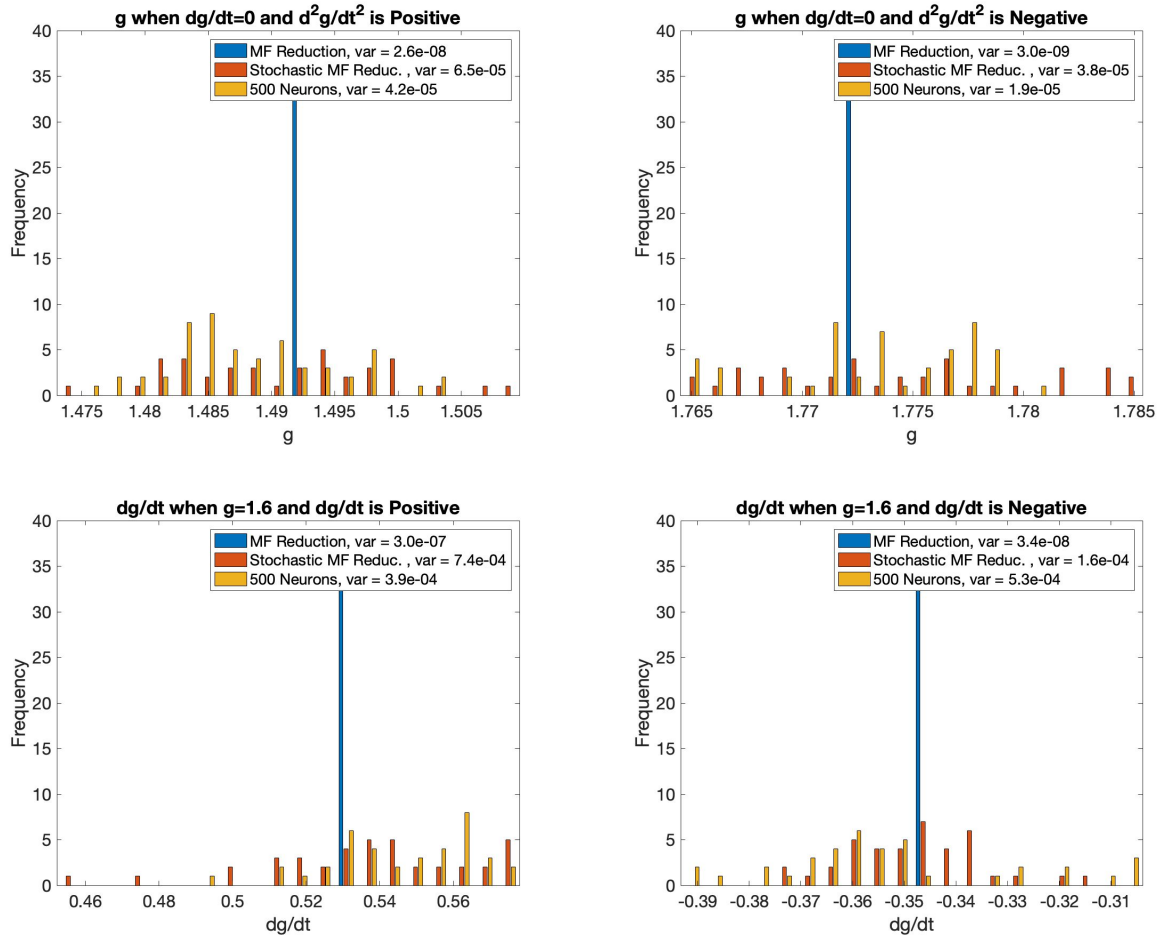

Fig. 8. Top: values of g when dg/dt is 0. Bottom: values of dg/dt when g is 1.6 . The limit cycle crosses each of these lines twice. Left: Crossing on the left or top. Right: Crossing on the right of bottom. Variance for each model is shown in the legend. The reduced mean field model is in blue, the same model with stochasticity added to the synaptic conductance is in red, and a simulation of a network of 500 θ-neurons is in orange. Notice that the variance for the stochastic model is on the same order of magnitude as the variance in the 500 θ-neuron model, while the reduced mean field model has a much smaller variance.

Each of the histograms in figure 8 is used to calculate a variance for the models. The variances of the stochastic reduced mean field model should change with ε. We want to choose ε

such that the variances of the stochastic model and the 500 θ-neuron model match. In figure 9, we see how the variance of the upper left plot in fig. 8 varies as ε varies.

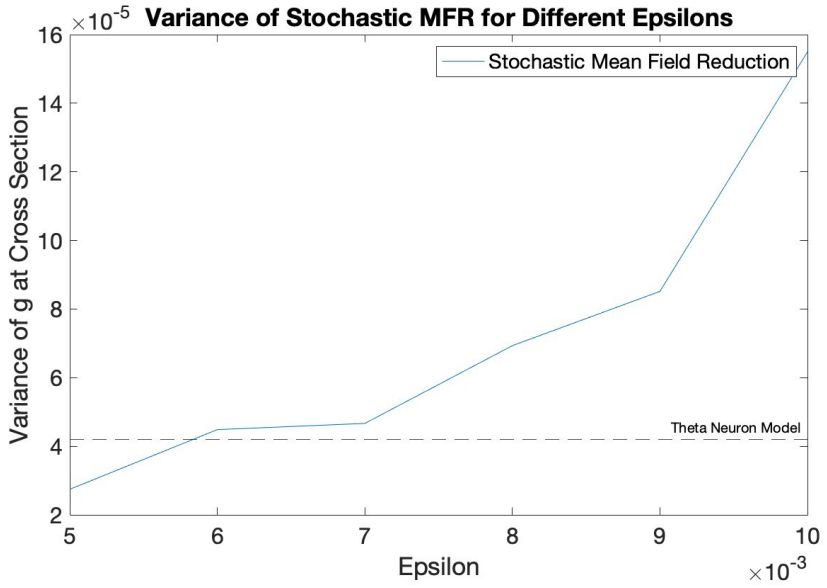

Fig. 9. Choice of ε versus variance in g at cross section. Cross section is when dg/dt is 0 and $d^2g/dt^2>0$. Variance for the stochastic reduced mean field model is in blue. Variance for the network of 500 θ-neurons is the dashed line. Notice that for $\varepsilon \sim 5.8\text{x}10^{-3}$ the variance for the stochastic model is on the same order of magnitude as the variance in the 500 θ-neuron model.

We see that values of ε around $5.8\text{x}10^{-3}$ match the variance of the stochastic reduced mean field model with the network of θ-neurons to within an order of magnitude. This is also true for the other histograms in figure 8 and over multiple simulations.

CONCLUSION

In our project, our goal was to determine whether stochasticity could be added into a mean field reduced neural mass model to reflect the noise seen in individual neurons. We compared the variance of our stochastic model to that of the 500 theta neurons. Our findings, illustrated in figure 8 above, show that the stochastic model closely simulates the variance seen

in the theta model. While not exactly the same, we can see that the variances are within the same magnitude, and tend to only differ by about a factor of two. To better understand how our model could be optimized to reflect the individual firing noise, we decided to analyze exactly what levels of noise would give us the best representation of the individual neurons. Our results for this process are shown in figure 9, where we show what levels of ε (noise strength) best reflect the variance seen in the 500 theta neuron network. The exact value of an optimal ε can change on a given run, but it tends to vary between $5*10^{-3}$ to $8*10^{-3}$. From this, we can see that it is possible to regain some of the noise initially present in the system, but further analyze should be done to determine how well this stochasticity truly represents individual neurons.

With this, potential future work could improve upon our analysis of the stochastic and network models. While variance is a fine measure for our purposes, there are most likely other artifacts in a simulated network that our model does not account for. Similarly, a metric other than variance could produce different results than those that were found here. Overall, we have only set a base for all of the work that may be done to explore this single neural mass model.

REFERENCES

[1] H. R. Wilson and J. D. Cowan, “Excitatory and Inhibitory Interactions in Localized Populations of Model Neurons,” Biophysical Journal, vol. 12, no. 1, pp. 1–24, Jan. 1972.

[2] S. Coombes and Á. Byrne, “Next Generation Neural Mass Models,” Nonlinear Dynamics in Computational Neuroscience, pp. 1–16, Jun. 2018.

[3] G. B. Ermentrout and N. Kopell, “Parabolic Bursting in an Excitable System Coupled with a Slow Oscillation,” SIAM Journal on Applied Mathematics, vol. 46, no. 2, pp. 233–253, Apr. 1986.

[4] Y. Yarom and J. Hounsgaard, “Voltage Fluctuations in Neurons: Signal or Noise?,” Physiological Reviews, vol. 91, no. 3, pp. 917–929, Jul. 2011.

CODE APPENDIX

Equation 10:

```
function f = curlyF(Z, eta_0, delta)
    f = -1i*(Z-1)^2/2+(Z+1)^2/2*(-delta+1i*eta_0);
```

Equation 11:

```
function g = curlyG(Z, g, v_syn)
    g = 1i*(Z+1)^2/2*v_syn*g-(Z^2-1)/2*g;
```

Equation 13:

```
function f = f_Z(Z)
    R = real(Z);
    I = imag(Z);
```

```
    mag_sq = R^2 + I^2;


    f = (1 - mag_sq)/(1+2*R+mag_sq);
```

Single step of the reduced mean field model:

```
function dy_dt = meanfield_ode(t, y, p)

    g = y(1);

    dg_dt = y(2);

    Z = y(3);

    eta_0 = p(1);

    delta = p(2);

    v_syn = p(3);

    kappa = p(4);

    alpha = p(5);


    dg_dt2 = alpha^2*(kappa*f_Z(Z)-2/alpha*dg_dt-g);

    dZ_dt = curlyF(Z, eta_0, delta) + curlyG(Z, g, v_syn);


    dy_dt = [dg_dt; dg_dt2; dZ_dt];
```

Single step of the network of 500 θ-neurons:

```
function dy_dt = theta_ode(t, y, p)

    N = p(1);

    v_syn = p(2);

    kappa = p(3);

    alpha = p(4);

    eta_i = p(5:end-1);

    step_size = p(end);
```

```
    theta = y(1:N).';
    g = y(N+1);
    dg_dt = y(N+2);

    dtheta_dt = (1-cos(theta))+(1+cos(theta)).*(eta_i+g*v_syn)-g*sin(theta);

    sum_th = 0;
    for i=1:N
        theta_old = floor((theta(i)-pi)/(2*pi));
        theta_new = floor((theta(i)+dtheta_dt(i)*step_size-pi)/(2*pi));
        sum_th = sum_th + (theta_new - theta_old)/step_size;
    end
    r_side = (pi*kappa/N)*(sum_th);
    dg_dt2 = alpha^2*(r_side-2/alpha*dg_dt-g);

    dy_dt = [dtheta_dt; dg_dt; dg_dt2];
```

Given one of the two above models, perform Euler–Maruyama method. With epsilons set to 0, this is just the euler method:

```
% similar API to ode45
function Y = sde(f, range, initial, p)
    % determine number of steps and size
    numsteps = size(range);
    numsteps = numsteps(2);
    init_size = size(initial);
    init_size = init_size(1);
```

```
    Y = zeros(numsteps, init_size);
    Y(1, :) = initial;

    epsilon = p(end-init_size:end-1)';
    h = p(end);
    p = p(1:end-init_size-1);

    %now, do the STOCHASTIC euler method!
    for n=1:(numsteps-1)
        Y(n+1,:)=Y(n,:) + h*f(0,Y(n,:),p).' +
epsilon.*sqrt(h).*randn(1,init_size);
    end
```

Plot figure 1, a simulation of a single theta neuron:

```
num_steps = 10000;
bin_size = 10;
alpha = 0.001;

thetas = zeros(num_steps, 1);
spikes = zeros(floor(num_steps/bin_size)+1, 1);
alpha_t_pi = zeros(floor(num_steps/bin_size)+1, 1);
I = sin(alpha*(1:num_steps));
for t = 2:num_steps
    dTheta_dt = (1 - cos(thetas(t - 1))) + (1 + cos(thetas(t - 1)))*I(t - 1);
    thetas(t) = thetas(t - 1) + dTheta_dt;
    thetas(t) = mod(thetas(t), 2*pi);
```

```
    if thetas(t-1) < pi && thetas(t) > pi
        spikes(floor(t/bin_size)+1) = spikes(floor(t/bin_size)+1) + 1;
    end
    alpha_t_pi(floor(t/bin_size)+1) = alpha*t/pi;
end

set(gca,'fontsize', 16);
yyaxis left
bar(alpha_t_pi, spikes, 10)
ylabel('Spiking Frequency')
yyaxis right
plot((1:num_steps)*alpha/pi, I)
ylabel('I')

title('Theta Model for I(t) := sin(\alpha t)')
xlabel('\alpha t / \pi')
```

Plots / Simulations in REPRODUCTION OF RESULTS section:

```
eta_0 = 20;
delta = 0.5;
v_syn = -10;
kappa = 1;
alpha = 0.95;

epsilon_mf = [0;0;0];

N = 500;
```

```
epsilon_th = zeros(N+2,1);

theta_i = rand(N, 1)*2*pi;

eta_i = trnd(1,N,1)*delta+eta_0;


g_i = 1.65;

dg_i = 0.5;

Z_i = 0;


t_i = 0;

t_f = 100;

step_size = 0.001;


%%

Y_mf = sde(@meanfield_ode, t_i:step_size:t_f, [g_i;dg_i;Z_i],...

   [eta_0;delta;v_syn;kappa;alpha;...

    epsilon_mf;step_size]);


Y_th = sde(@theta_ode, t_i:step_size:t_f, [theta_i; g_i; dg_i], ...

    [N;v_syn;kappa;alpha;eta_i;step_size;epsilon_th;step_size]);

Z = (1/N)*sum(exp(1i*Y_th(:, 1:N)), 2);

%%

t = t_i:step_size:t_f;

figure

hold on

plot(t, Y_th(:, N+1), 'LineWidth',1,'Color','r');

plot(t, Y_mf(:, 1), 'LineWidth',1,'Color','b');

hold off
```

```
xlabel('Time');
ylabel('g');
title('Time Evolution of Synaptic Conductance');
legend('Theta Neuron Model','Mean Field Reduction')
set(gca,'FontSize',16);

figure
hold on
plot(t, real(Z), 'LineWidth',1,'Color','r');
plot(t, real(Y_mf(:, 3)), 'LineWidth',1,'Color','b');
hold off
xlabel('Time');
ylabel('Re(Z)');
title('Time Evolution of Real Part of Kuramoto Order Parameter');
legend('Theta Neuron Model','Mean Field Reduction')
set(gca,'FontSize',16);

figure
hold on
plot(t, imag(Z), 'LineWidth',1,'Color','r');
plot(t, imag(Y_mf(:, 3)), 'LineWidth',1,'Color','b');
hold off
xlabel('Time');
ylabel('Im(Z)');
title('Time Evolution of Imaginary Part of Kuramoto Order Parameter');
legend('Theta Neuron Model','Mean Field Reduction')
set(gca,'FontSize',16);
```

```
%%
headstart = 1;
figure
hold on
plot(Y_th(headstart:end, N+1), Y_th(headstart:end, N+2),
'LineWidth',0.05,'Color','r');
plot(Y_mf(headstart:end, 1), Y_mf(headstart:end, 2),
'LineWidth',0.05,'Color','b');
hold off
xlabel('g');
ylabel('dg/dt');
title('Phase Plane Projection for the Synaptic Conductance');
legend('Theta Neuron Model','Mean Field Reduction')
set(gca,'FontSize',16);

figure
hold on
plot(real(Z(headstart:end)), imag(Z(headstart:end)),
'LineWidth',0.05,'Color','r');
plot(real(Y_mf(headstart:end, 3)), imag(Y_mf(headstart:end, 3)),
'LineWidth',0.05,'Color','b');
plot(sin(0:0.01:2*pi), cos(0:0.01:2*pi));
hold off
xlabel('Re(Z)');
ylabel('Im(Z)');
title('Phase Plane Projection for the Kuramoto Order Parameter');
```

```
legend('Theta Neuron Model','Mean Field Reduction')


axis equal
set(gca,'FontSize',16);
%%
headstart = floor(t_f/step_size/5);
figure
hold on
plot(Y_th(headstart:end, N+1), Y_th(headstart:end, N+2),
'LineWidth',0.05,'Color','r');
plot(Y_mf(headstart:end, 1), Y_mf(headstart:end, 2),
'LineWidth',0.05,'Color','b');
hold off
xlabel('g');
ylabel('dg/dt');
title('Phase Plane Projection for the Synaptic Conductance');
legend('Theta Neuron Model','Mean Field Reduction')
set(gca,'FontSize',16);


figure
hold on
plot(real(Z(headstart:end)), imag(Z(headstart:end)),
'LineWidth',0.05,'Color','r');
plot(real(Y_mf(headstart:end, 3)), imag(Y_mf(headstart:end, 3)),
'LineWidth',0.05,'Color','b');
plot(sin(0:0.01:2*pi), cos(0:0.01:2*pi));
hold off
```

```
xlabel('Re(Z)');
ylabel('Im(Z)');
title('Phase Plane Projection for the Kuramoto Order Parameter');
legend('Theta Neuron Model','Mean Field Reduction')

axis equal
set(gca,'FontSize',16);
```

Plot Figure 6, understanding f(Z):

```
step_size = 0.01;
R = -0.8:step_size:0.3;
I = -0.4:step_size:0.7;
leng = size(R, 2);
f = zeros(leng);
for i=1:leng
    for j=1:leng
        Z = R(i)+1i*I(j);
        if abs(Z) < 1
            f(i, j) = f_Z(Z);
        end
    end
end
%%
eta_0 = 20;
delta = 0.5;
v_syn = -10;
kappa = 1;
```

```
alpha = 0.95;


epsilon_mf = [0;0;0];


N = 500;
epsilon_th = zeros(N+2,1);
theta_i = rand(N, 1)*2*pi;
eta_i = trnd(1,N,1)*delta+eta_0;


g_i = 1.65;
dg_i = 0.5;
Z_i = 0;


t_i = 0;
t_f = 100;
step_size = 0.001;


Y_mf = sde(@meanfield_ode, t_i:step_size:t_f, [g_i;dg_i;Z_i],...
   [eta_0;delta;v_syn;kappa;alpha;...
    epsilon_mf;step_size]);


Y_th = sde(@theta_ode, t_i:step_size:t_f, [theta_i; g_i; dg_i], ...
    [N;v_syn;kappa;alpha;eta_i;step_size;epsilon_th;step_size]);
Z = (1/N)*sum(exp(1i*Y_th(:, 1:N)), 2);


%%
headstart = floor(t_f/step_size/5);
```

```
figure
hold on
contourf(R, I, f.', 20)
plot(real(Z(headstart:end)), imag(Z(headstart:end)),
'LineWidth',0.05,'Color','r');
plot(real(Y_mf(headstart:end, 3)), imag(Y_mf(headstart:end, 3)),
'LineWidth',0.05,'Color','b');
plot(sin(0:0.01:2*pi), cos(0:0.01:2*pi));
hold off
colorbar()
colormap(gray)
xlabel('Re(Z)');
ylabel('Im(Z)');
title('Contribution of KO Parameter to Synaptic Conductance');
legend('\kappaf(Z)', 'Theta Neuron Model','Mean Field Reduction', 'Magnitude of
1')

axis equal
set(gca,'FontSize',16);
```

Plots / Simulations in NOVEL RESULTS section:

```
eta_0 = 20;
delta = 0.5;
v_syn = -10;
kappa = 1;
alpha = 0.95;
```

```
epsilon_mf = [0;0;0];

epsilon_mf_s = [0.007;0;0];


N = 500;

epsilon_th = zeros(N+2,1);

theta_i = rand(N, 1)*2*pi;

eta_i = trnd(1,N,1)*delta+eta_0;


g_i = 2;%1.65;

dg_i = 0;%0.5;

Z_i = 0;


t_i = 0;

t_f = 100;

step_size = 0.001;

%%

Y_mf = sde(@meanfield_ode, t_i:step_size:t_f, [g_i;dg_i;Z_i],...

   [eta_0;delta;v_syn;kappa;alpha;...

    epsilon_mf;step_size]);


Y_mf_s = sde(@meanfield_ode, t_i:step_size:t_f, [g_i;dg_i;Z_i],...

   [eta_0;delta;v_syn;kappa;alpha;...

    epsilon_mf_s;step_size]);


Y_th = sde(@theta_ode, t_i:step_size:t_f, [theta_i; g_i; dg_i], ...

    [N;v_syn;kappa;alpha;eta_i;step_size;epsilon_th;step_size]);
```

```
%%
headstart = floor(t_f/step_size/5);
means = [1.6;0];
figure
hold on
plot(Y_th(headstart:end, N+1), Y_th(headstart:end, N+2), 'LineWidth',0.05,
'Color', 'r');
plot(Y_mf_s(headstart:end, 1), Y_mf_s(headstart:end, 2), 'LineWidth',0.05,
'Color', [0.01,0.44,0.28]);
plot(Y_mf(headstart:end, 1), Y_mf(headstart:end, 2), 'LineWidth',0.05, 'Color',
'b');
hold off
xlabel('g');
ylabel('dg/dt');
xline(means(1),'--k','LineWidth',0.03);
yline(means(2),'--k','LineWidth',0.03);
title('Phase Plane Projection for Synaptic Conductance');
legend('Theta Neuron Model','Stochastic Mean Field Reduction','Mean Field
Reduction')
set(gca,'FontSize',16);

Z = (1/N)*sum(exp(1i*Y_th(:, 1:N)), 2);
figure
hold on
plot(real(Z(headstart:end)), imag(Z(headstart:end)),
'LineWidth',0.05,'Color','r');
```

```
plot(real(Y_mf_s(headstart:end, 3)), imag(Y_mf_s(headstart:end, 3)),
'LineWidth',0.05, 'Color', [0.01,0.44,0.28]);
plot(real(Y_mf(headstart:end, 3)), imag(Y_mf(headstart:end, 3)),
'LineWidth',0.05,'Color','b');
plot(sin(0:0.01:2*pi), cos(0:0.01:2*pi));
hold off
xlabel('Re(Z)');
ylabel('Im(Z)');
title('Phase Plane Projection for the Kuramoto Order Parameter');
legend('Theta Neuron Model','Stochastic Mean Field Reduction','Mean Field
Reduction')

axis equal
set(gca,'FontSize',16);
%%
num_steps = 20;

cuts_mf = cut(Y_mf(headstart:end, 1:2), means);
cuts_mf_s = cut(Y_mf_s(headstart:end, 1:2), means);
cuts_th = cut(Y_th(headstart:end, N+1:N+2), means);

hist_titles = ["g when dg/dt=0 and d^2g/dt^2 is Positive",...
    "g when dg/dt=0 and d^2g/dt^2 is Negative",...
    "dg/dt when g=1.6 and dg/dt is Positive",...
    "dg/dt when g=1.6 and dg/dt is Negative"];
hist_axis = ["g", "g", "dg/dt", "dg/dt"];
for i = 1:4
```

```
% iterate and access cell arrays
cut_mf = cell2mat(cuts_mf(i));
cut_mf_s = cell2mat(cuts_mf_s(i));
cut_th = cell2mat(cuts_th(i));

% focus on variance, not mean
%cut_mf = cut_mf - mean(cut_mf);
%cut_mf_s = cut_mf_s - mean(cut_mf_s);
%cut_th = cut_th - mean(cut_th);

minimum = min([min(cut_mf), min(cut_mf_s), min(cut_th)]);
maximum = max([max(cut_mf), max(cut_mf_s), max(cut_th)]);
edges = minimum:(maximum-minimum)/num_steps:maximum;
h_mf = histcounts(cut_mf,edges);
h_mf_s = histcounts(cut_mf_s,edges);
h_th = histcounts(cut_th,edges);
figure
bar(edges(1:end-1),[h_mf; h_mf_s; h_th]')
title(hist_titles(i))
legend(['MF Reduction, var = ' num2str(var(cut_mf), '%10.1e')],...
    ['Stochastic MF Reduc. , var = ' num2str(var(cut_mf_s), '%10.1e')],...
    ['500 Neurons, var = ' num2str(var(cut_th), '%10.1e')])
xlabel(hist_axis(i));
ylabel('Frequency');
set(gca,'FontSize',16);

if i == 1
```

```
        targ_var = var(cut_th);
    end
end
%%
vars_mf = [];
eps_range = 0.005:0.001:0.01;
for eps_mf_s=eps_range
    epsilon_mf_s = [eps_mf_s;0;0];

    Y_mf_s = sde(@meanfield_ode, t_i:step_size:t_f, [g_i;dg_i;Z_i],...
       [eta_0;delta;v_syn;kappa;alpha;...
        epsilon_mf_s;step_size]);
    cuts_mf_s = cut(Y_mf_s(headstart:end, 1:2), means);
    cut_mf_s = cell2mat(cuts_mf_s(1));

    vars_mf = [vars_mf;var(cut_mf_s)];
end
%%
plot(eps_range, vars_mf)
yline(targ_var, '--k', 'Theta Neuron Model');
title('Variance of Stochastic MFR for Different Epsilons')
legend('Stochastic Mean Field Reduction')
xlabel('Epsilon');
% when dg/dt=0 and d^2g/dt^2 is Positive
ylabel('Variance of g at Cross Section');
set(gca,'FontSize',16);
```

Helper function to perform cross sections for figure 8:

```
function cuts  = cut(Y, means)
    left_cut = [];
    right_cut = [];
    high_cut = [];
    low_cut = [];
    for i=1:length(Y)-1
        % Y(1) passes mean while increasing, high cut
        if Y(i, 1) < means(1) && Y(i+1, 1) > means(1) && Y(i, 2) > means(2)
            high_cut = [high_cut; Y(i, 2)];
        end
        % Y(1) passes mean while decreasing, ; low cut
        if Y(i, 1) > means(1) && Y(i+1, 1) < means(1) && Y(i, 2) < means(2)
            low_cut = [low_cut; Y(i, 2)];
        end
        % Y(2) passes mean while increasing, left cut
        if Y(i, 2) < means(2) && Y(i+1, 2) > means(2) && Y(i, 1) < means(1)
            left_cut = [left_cut; Y(i, 1)];
        end
        % Y(2) passes mean while decreasing, right cut
        if Y(i, 2) > means(2) && Y(i+1, 2) < means(2) && Y(i, 1) > means(1)
            right_cut = [right_cut; Y(i, 1)];
        end
    end

    cuts = {left_cut, right_cut, high_cut, low_cut};
```