Computational Cognitive Neuroscience: Assignment 1

Xiao Heng (s2032451)

February 22, 2024

(1)

Interpretation of the various terms in Equations (1) and (2):

Dynamics of excitatory-inhibitory networks of firing rate neurons governed by the Wilson-Cowan equations are given as:

$$\tau_E \frac{dV_E}{dt} = -(V_E - V_{rest}) + W_{EE}\phi(V_E) - W_{EI}\phi(V_I) + u_E \tag{1}$$

$$\tau_I \frac{dV_I}{dt} = -(V_I - V_{rest}) + W_{IE}\phi(V_E) - W_{II}\phi(V_I) + u_I \tag{2}$$

Note that a classic firing rate network model could be written as:

$$\tau_m \frac{dV_i}{dt} = -(V_i - E_m) + \sum_j \omega_{ij}\phi(V_j) + I_{ext,i}(t) \tag{3}$$

This has been introduced in lecture 5, and it could be further modified by introducing the consideration of separate E and I populations, so that we obtain Equations (1) and (2). In this case, the definitions and explainations of associated variables are as following:

- $\boldsymbol{\tau_{E/I}}$: Membrane time constant of excitatory/inhibitory neurons, given by $\tau_m = C_m/g_m$, where C_m is membrane capacitance and $g_m = \sum_i g_i$ ($g = 1/R$) is the membrane conductance (related to its permeability to ions);
- $\boldsymbol{V_{E/I}}$: Membrane potential of excitatory/inhibitory neurons;
- $\boldsymbol{V_{rest}}$: Resting potential;
- $\boldsymbol{W_{EE/EI/IE/II}}$: W_{EI} indicates the strength of connection from inhibitory neurons to excitatory neurons, and similarly W_{EE}, W_{IE}, W_{II} represent $E \to E$, $E \to I$, $I \to I$ respectively;
- $\boldsymbol{\phi(V_{E/I})}$: Nonlinear (supralinear) function of the voltage, indicating firing rate of excitatory/inhibitory neurons;
- $\boldsymbol{u_{E/I}}$: External input to excitatory/inhibitory neurons. It could be potentially time-varying but deterministic component (the mean), and noise could be added.

Biologically unrealistic parts (approximations) within the model:

E-I separation: By separation the neuron networks into two groups, excitatory neurons and inhibitory neurons, we ignore sub-types (Rubin et al., 2015). Futhermore, within each group, we assume that the excitatory or inhibitory neurons have identical property (a homogeneous population of interconnected E/I neurons);

Same average excitation: All neurons receive the same number of excitatory and inhibitory afferents, that is, all cells receive the same average excitation;

Some other biologically unrealistic approximations: Interactions are modelled in a single layer, ignoring interlaminar processing; Assume that the net effect of externally driven input (external input) to this layer is excitatory (Rubin et al., 2015); Assume that the voltage fluctuations giving rise to the expansive input/output function are fast compared to the timescales of variability studied here (Hennequin et al., 2018).

Reasons to choose to study a model such as this rather than a more biologically realistic one:

As David Marr said, "The nature of computations that underlie perception depends more upon the computational problems that have to be solved than upon the particular hardware in which their solutions are implemented." Since we mainly focused on exploring the property of cortical networks called **stimulus-quenching of neural variability**, which could already be well-described by this model, there is no need to introduce additional complexity from those more biologically realistic ones, which might not only affect efficiency but even disturb the result, leading to complex figures.

(2)

Given the nonlinear (supralinear) function of the voltage:

$$\phi(V) = k[V - V_0]_+^n = \begin{cases} k(V - V_0)^n, & \text{for } V - V_0 > 0 \\ 0, & \text{for } V - V_0 \leq 0 \end{cases} \tag{4}$$

We could draw the corresponding function graph in Python, as Figure (1).

Biologically implausibility about this choice of transfer function:

With a V significantly lower than action potential, as long as it is higher than V_0, there would be a positive value of function result, indicating spikes exist, which is not very intuitive and would be explained in the following sub-sections. Then, by setting 0 when V is lower than V_0, it indicates that we assume the hyperpolarization cannot increase response.

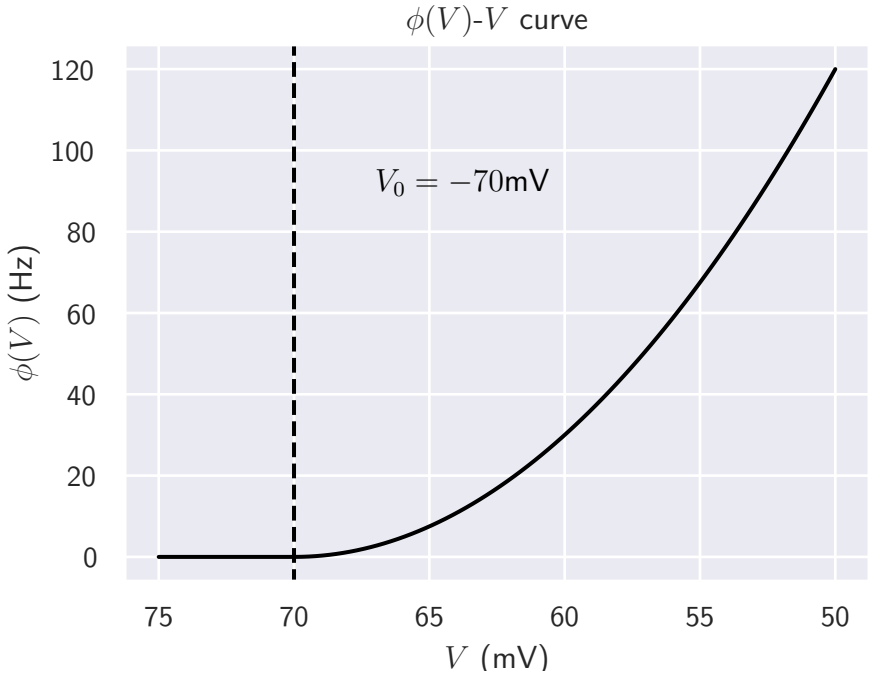

Figure 1: Nonlinear (supralinear) function determining the relationship between membrane potential and momentary firing rate of model neurons (Equation (2)).

What's more, the transfer function becomes steeper as V is increased due to the "supralinear" property (here we choose $n = 2 > 1$, which might cause highly unstable dynamics in the absence of inhibition) (Miller and Troyer, 2002). Besides, in the formula, k and n are generally taken identical for E and I neurons for simplicity, to focus on emergent network properties that arise even without cell-type differences (Rubin et al., 2015).

Biological processes or properties that might motivate such a transfer function:

In our model, neurons integrate their external and recurrent inputs linearly in their membrane potentials, V, and their output firing rates, $\phi(V)$, are chosen as a nonlinear function of the voltage (Hennequin et al., 2018), stated in Equation (4).

Under what conditions might this transfer function be a reasonable model for V1 neuronal firing rates:

A power-law relation between the mean input and mean response arises in the case that spiking is driven by input fluctuations (Miller and Troyer, 2002; Hansel and Vreeswijk, 2002), and similarly it is observed in V1 as the relation between the trial averaged mean voltage and mean response. Real neurons have an input/output function that ultimately saturates, but here it is completely acceptable because we only focus on an expansive, non-saturating input/output function due to the fact that V1 cortical neurons show such a relationship between mean voltage and firing rate across their full dynamic range, without saturation even for the strongest visual stimuli (Priebe and Ferster, 2008). In other words, the power law holds over the entire dynamic range of visual responses means that V1 neurons never reach firing rates at which intrinsic saturation of the input-output function plays a role (Ahmadian et al., 2013).

When might the approximations made by this choice of transfer function break down:

When the spiking is not driven by input fluctuations, the neurons need ultimately saturation, or hyperpolarization should be considered.

(3)

Set $u_E = \pm 2$, $u_I = \pm 2$. Simulate for $N_t = 1000$ time steps, and the result is shown in Figure (2).

Observations and what kinds of behaviour are you able to produce:

From Figure (2) we observe that larger excitatory input u_E leads to higher potential, while larger inhibitory input leads to lower potential. While in most cases the curves are smooth, with a positive excitatory input $u_E > 0$ and a negative inhibitory input $u_I < 0$ (which cannot inhibit but excite), the potential reaches a higher state, and unstable around the peak, followed by a fluctuated drop to a lower but stable potential. As all simulations finally converge to fixed values (though some lines go smoothly while some other lines experience a temporal fluctuation), such pattern should be the dynamics of fixed points (attractors), which may be useful for explaining memory. Considering the E-I interaction, it also shows a property of inhibitory stabilization, as the positive input of inhibitory neurons will inhibit excitatory ones, not allowing unrestricted growing.

How do the V terms differ from their $\phi(V)$ counterparts and how does this align with their biological interpretation:

Before discussing the difference, firstly, for the same patterns, notice that in the left subplot of Figure (2), as long as u_E is positive (excitation input), then both the V_E and V_I would be higher than resting potential (under common conditions that the inhibition effect is not dominating like this example), and the corresponding u_I only affects the degree of inhibition

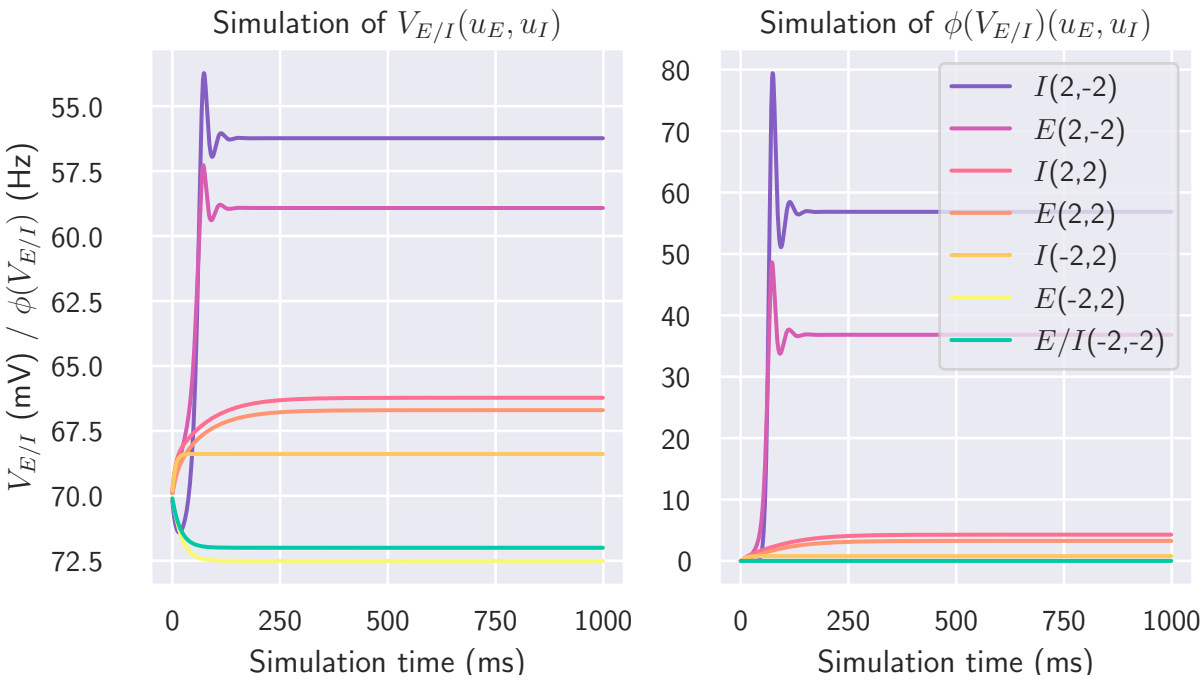

Figure 2: Simulation of $V_{E/I}$ and $\phi(V_{E/I})$ for 1000 steps (ms) under $u_{E/I} = \pm 2$.

(negative inhibition input leads to excitatory effect). Then, we could find also the positive values in the right subplot (even the same shape) when $u_E > 0$. Now we focus on the difference. When $u_E < 0$, $V_E < -70mV$ (resting potential), indicating a hyperpolarization state. As $V_E < V_0$, $\phi(V_E) = 0$. Under this condition, only when inhibitory input $u_I > 0$ that $V_I > 0$ so that $\phi(V_I)$ would be slightly greater than 0.

As for the biological interpretation implied by difference between V and $\phi(V)$, for $u_E = -2$ and $u_I = 2$, E-neurons are in hyperpolarization state so no spiking and $\phi(V_E) = 0$, while this hyperpolarization is due to the inhibition from I-neurons (driven by internal input), which is active (producing inhibit neurotransmitters), so $V_I > V_{rest}$ and $\phi(V_I) > 0$ (spiking). While for the case that $u_E = u_I = -2$, both groups are in hyperpolarization, driven by external input, so no one would spike, and $\phi(V_E) = \phi(V_I) = 0$

(4)

The external input with Gaussian white noise follows Equations (5) and (6):

$$u_E = \mu_E + (1/\sqrt{\delta t})\sigma_E \eta_E \tag{5}$$

$$u_I = \mu_I + (1/\sqrt{\delta t})\sigma_I \eta_I \tag{6}$$

By setting $\mu_E = \mu_I = 2$, $\sigma_E = 1$, $\sigma_I = 0.5$, $\eta_E \sim N(0,1)$, $\eta_I \sim N(0,1)$ and $\delta t = 1ms$, we could draw the plots with 5 different random seeds, shown in Figure (3).

(5)

With $\mu_E = \mu_I = 0 \to 2 \to 15$ under noise-free and noisy conditions respectively, we could draw the plots as Figure (4).

By visual inspection, how do the response statistics appear to change with increasing μ:

First we consider the mean of results, from the left two graphs (which are noise-free). Compared with the "benchmark" of external input $u_{E/I}$ in purple dashed lines, with the inputs increasing from 0 mV to 15 mV, values of $V_{E/I}$ does not grows proportionally (under suppression for high excitation) while the trends of $\phi(V_{E/I})$ almost keep the same changing amplitude.

Then in the right graph we focus on the fluctuations. When $\mu_E = \mu_I = 0$, there is no external-input-driven effect on the network, and V_E and V_I fluctuate around $Vrest = -70mV$ within a small amplitude (low variance), and there is almost no fluctuation for $\phi(V_{E/I})$ (even when $V_{E/I} > 0$ driven by fluctuation, the quadratic function of a decimal number will still give a result close to 0). With external input increases to 2 mV, the fluctuation of $V_{E/I}$ increases a lot while the fluctuation of $\phi(V_{E/I})$ increase not too much. When the external input turns into 15 mV, the fluctuation of $V_{E/I}$ decreases (magnitude was strongly quenched) while the fluctuation of $\phi(V_{E/I})$ increase dramatically. Besides, all fluctuations keep a high correlation between E and I groups.

(6)

Vary $\mu_{E/I}$ systematically as $\mu_E = \mu_I = 0 \to 20$, we plot the mean and standard deviation of $V_{E/I}$ and $\phi(V_{E/I})$ respectively, shown in Figure (5).

Discuss the dependence of these response statistics on μ:

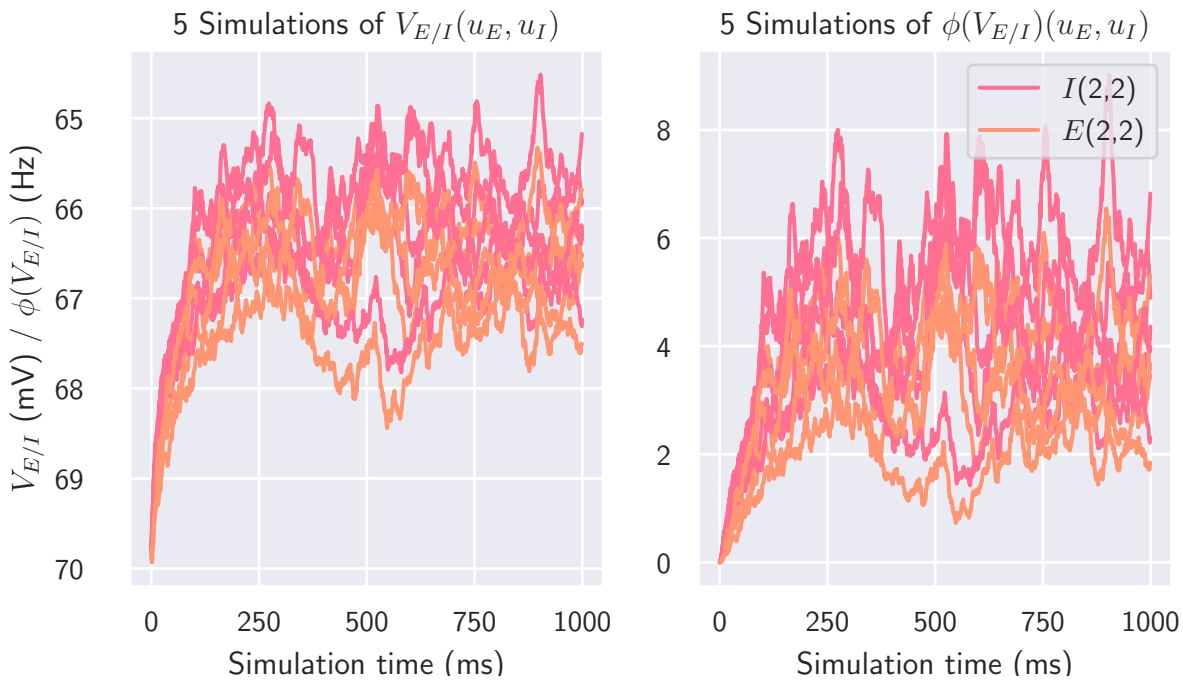

Figure 3: Repeated simulations with different random seeds under the same modle parameters ($u_E = u_I = 2$)

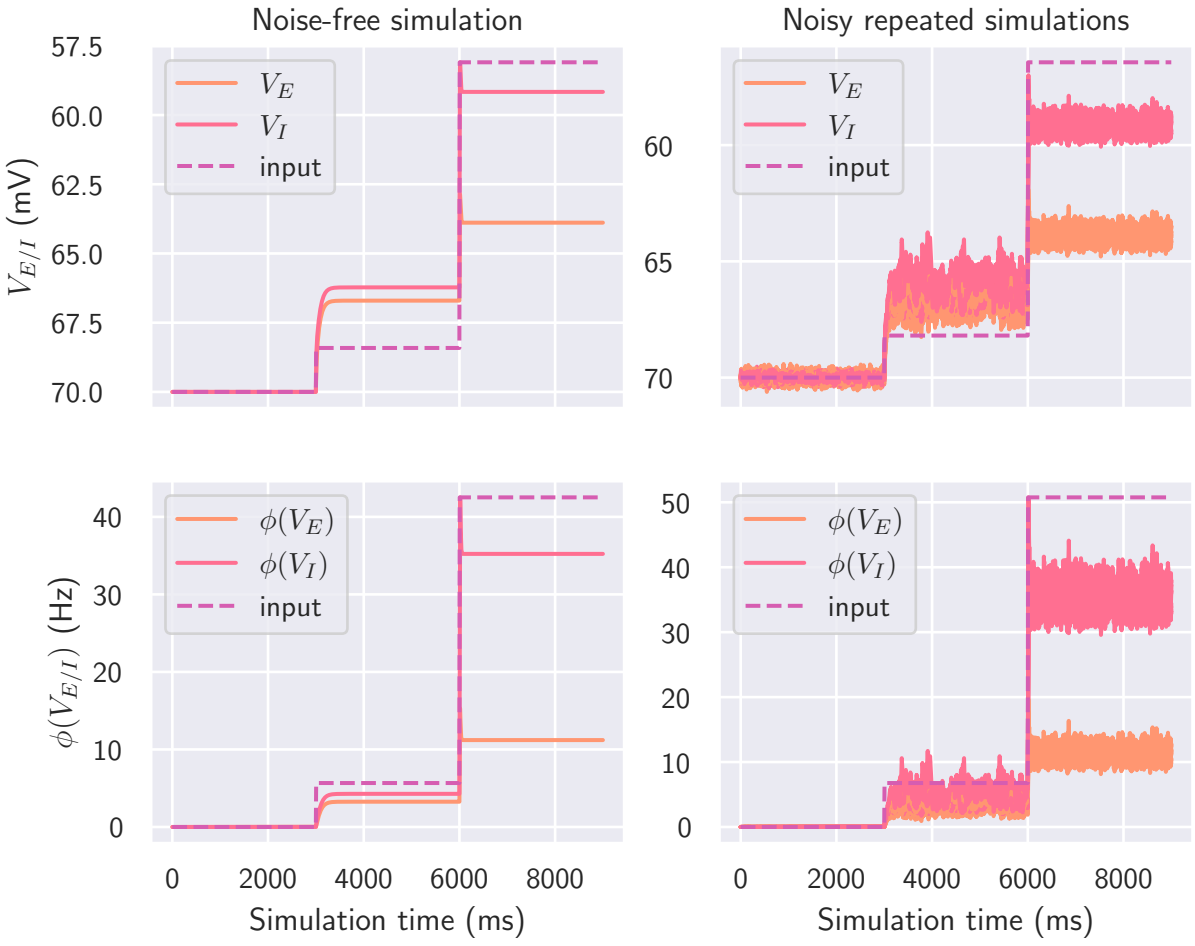

Figure 4: Simulations with varying $\mu_{E/I}$ under noise-free and noisy (5 trials) conditions respectively, and the purple dashed lines indicate the transition of $\mu_E = \mu_I = 0 \to 2 \to 15$.

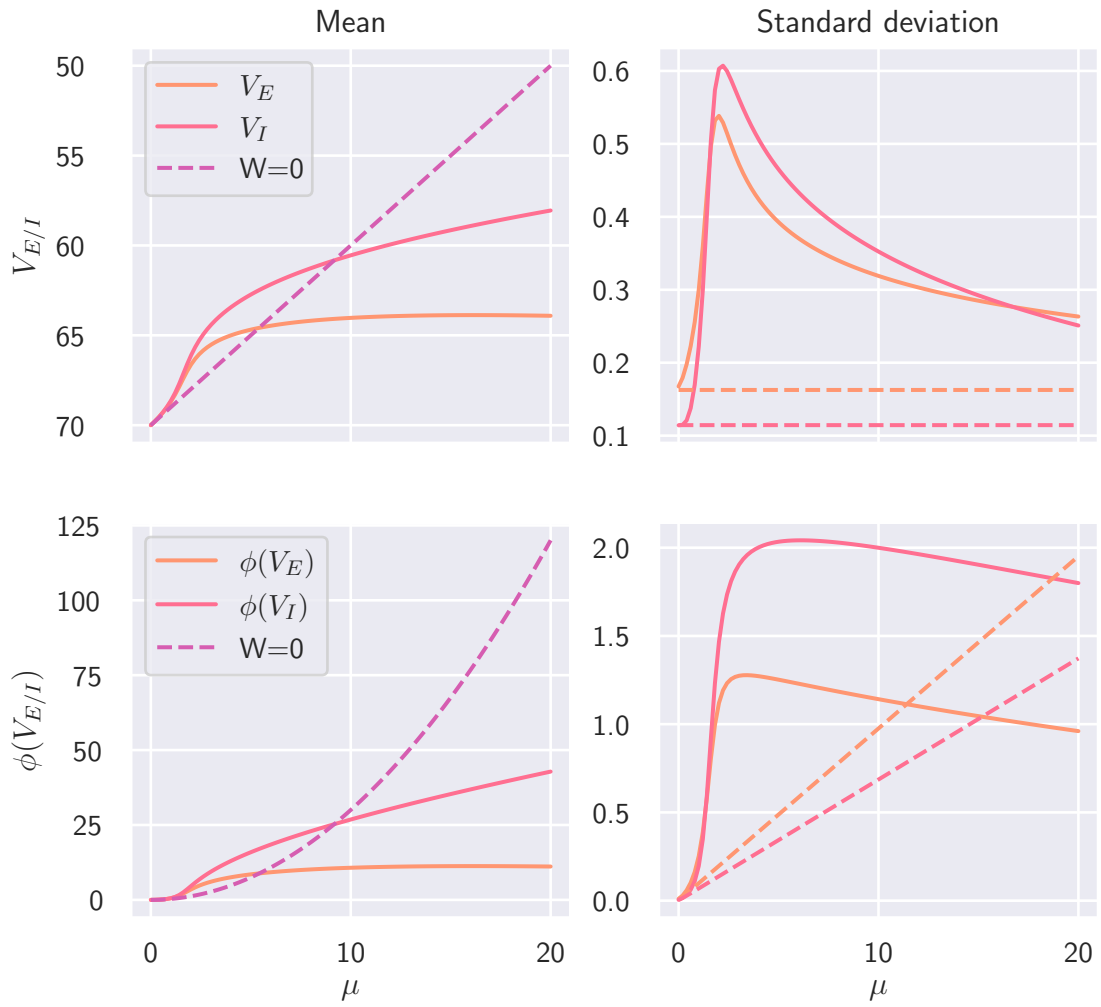

Figure 5: Dependence of statistics on stimulus strength $\mu_{E/I}$. The comparison with a purely feedforward network $W_{EE} = W_{EI} = W_{IE} = W_{II} = 0$ receiving the same external input is shown in dashed lines. When calculate the statistical indices, we truncate the first 500 simulation data (actually a smaller choice like 100 is also enough) so that we eliminate the data during growing stage when the external input is introduced initially.

Figure (5) shows the temporal pattern of mean and standard deviation (variability) of respond strength (activities) varying over a range of external input powers. The mean of V_m increased monotonically with growing μ, linearly or supralinearly (this result could be easily obtained by comparing the solid lines with the dashed line, which indicates the linear condition with absence of W) in the early stage (input is weak).

Then it turns sublinearly (smaller slope compared with dashed lines) when inputs grow large, as well as the fact that the mean of V_E or $\phi(V_E)$ now is significantly larger than the mean of V_I or $\phi(V_I)$, and the slope of the former one (indicating growth speed) is also larger than the latter one.

On the other hand, the standard deviation (variability) for $V_{E/I}$ increases when external input is still small. It grows supralinearly first and then sublinearly until the peak, and then goes down gradually with the growth of input. Also note that when $\mu \approx 16$, there is an intersection, indicating a lower variability of V_I in the future. As for the condition of $\phi(V_{E/I})$, the total trend or pattern is roughly similar with the case of $V_{E/I}$, the main difference may be that the decreasing speed is much lower. When $W = 0$, no inhibitory stabilization exists, and the variability of $V_{E/I}$ keeps constant while the variability of $\phi(V_{E/I})$ grows linearly with μ.

Hennequin et al. (2018) also noted that, input modulation of variability required recurrent network interactions. This was revealed by comparing our network to a purely feedforward circuit that exhibited qualitatively different behavior (the dashed lines in Figure (5)). In the feedforward circuit, the mean of $V_{E/I}$ remained linear in μ, so that mean rates rose quadratically with $V_{E/I}$ or μ (reflecting the input/output nonlinearity in Figure (1)), and fluctuations in $V_{E/I}$ no longer depended on the input strength.

(7)

In what way can the above results be interpreted as a model for the phenomenon of stimulus-quenching of neural variability:

Spontaneous activity: Found by Hennequin et al. (2018) that the firing rates at the peak of variability are typically low (2.5 Hz on average over a thousand randomly parameterized stable networks and below 6 Hz for 90% of them). In this case, we could regard the 2 mV condition in Figure (4) as spontaneous activity status. This is because in the buttom right plot, The values of $\phi(V_{E/I})$ under 2 mV (3001 to 6000 ms period) are exactly fluctuating around 4 to 5 Hz, which is consistent with this research result. At the same time, we could connect this result with Figure (5), in which we could find the standard deviation (variability) computed by simulations. For a 2 mV "input" (since we could regard the spontaneous activity as a "low input" during simulation, as the corresponding rates are comparable to cortical spontaneous firing rates), both the standard deviations (variabilities) of V_E and V_I are almost at the peak, around 0.5 to 0.6.

Variability suppression by strong visual stimulus: As 2 mV conditions are comparable to cortical spontaneous firing rates, whose variability is almost at the peak, we could intuitively assume that a significant increased sensory drive should generally result in variability quenching. Now we check this from Figure (4) and (5). To adopt the result of Figure (4) to analyse, we are assuming that there is an external input which leads the nerouns to showing a activity pattern under 15 mV. So now we see what happens under 15 mV condition. In upper right plot of Figure (4), visually, we notice a decrease of standard deviation (variability) under 15 mV, compared with the case under 2 mV. And in the upper right plot of Figure (5), the standard deviations of both V_E and V_I are within 0.25 to 0.3, which is significantly lower than the values of variability around 0.5 to 0.6 under 2 mV case, discussed previously, showing a apparent phenomenon of so-called variability suppression by strong visual stimulus.

(8)

What properties of the SSN model might give rise to this stimulus-quenching of variability, and in particular, how might this effect depend on the form of transfer function and the interactions between the excitatory and inhibitory populations:

SSN properties: The property of **inhibitory stabilization of strong excitatory feedback** and **supralinearity of input/output functions (transfer function) in single neurons** of SSN model leads to the stimulus-quenching of variability.

Form of transfer function: In detail, from Equation (4) and Figure (1), given $k = 0.3$, $n = 2$ and $V_0 = V_{rest}$, the transfer function here is in a quadratic form (the analysis would be similar to any transfer functions with $n > 1$, but to be clear, here we directly discuss the effect under $n = 2$). With $\phi(V) = 0.3(V - V_{rest})^2$ when $V > V_{rest}$, let $V^* = V - V_{rest}$, we could obtain derivative $\phi(V_{rest} + V^*)' = 1$ when $V^* \approx 1.67$ (for other transfer functions with $n > 1$, there would always be a $V^* > 0$ with $\phi(V_{rest} + V^*)' = 1$). So for all $V > V_{rest}$: (i) when $V - V_{rest} < V^*$, $\phi(V)$ grows sublinearly (and function value is close to 0); (ii) when $V - V_{rest} = V^*$, $\phi(V)$ grows linearly (always only for a moment); (iii) when $V - V_{rest} > V^*$, $\phi(V)$ grows supralinearly (causing high level of E-I interactions). Such mathematical property leads to the fact that, under a low input or spontaneous activity, the transfer function affect the dynamics of networks weakly; only when the external input is significant then the effect of transfer function will dominates. Such effect is related to E-I interaction, as discussed below.

E-I interactions: We have already seen that with the input increasing, V_I grows faster than V_E (from the upper left plot of Figure (5)), so the inhibition effect (due to the minus symbols in front of coefficients of W_{EI} and W_{II}) dominates the dynamics of networks. This feature is not obvious when input is low or under spontaneous activity status, because as we discussed above, without a significant large stimulus, the value of transfer function is small, and even grows sublinearly (slowly), so even we have W_{EI} and W_{II} in Equations (1) and (2), the product with $\phi(V_I)$ would be small, causing a weak effect in the interactions. Lack of the effect of $W\phi(V)$s, the networks are not dominated by inhibition stabilization. On the other hand, under large stimulus, as $\phi(V_I)$ grows supralinearly and V_I grows faster than V_E, the inhibitory stabilization property now dominates the dynamics of networks. And under such a significant inhibition effect, the random noise within large external input now can only drive the networks change slightly (weakly), with a smaller variability.

(9)

What advantages are there to taking the analytical approach for the response statistics of the SSN as opposed to the numerical approach you have taken for the above questions:

Computational efficiency and precise analytical solution: With the analytical approach of mean responses in the stabilized supralinear regime, Hennequin et al. (2018) found that the firing rates at the peak of variability are typically low (2.5 Hz on average over a thousand randomly parameterized stable networks and below 6 Hz for 90% of them. If we want to obtain the similar result, we need various of repeated simulations under a large parameter space, which is both complex with high computational cost and not precise (since it is just numerical solution rather than analytical solution). As for the numerical approach we have taken, if we need to implement more steps of simulations, or more parameters combinations, the work complexity will grow fast, and large number of repeatition is needed for estimators' convergency.

Generalization property, without specific parametrization effect: When comparing the differences of conclusions with other researches, Hennequin et al. (2018) found that such differences were caused by special, non-generic parametrization of the model of others. That means, under the numerical approaches, we always need a specific parameter setting to implement, while this operation may lead to a extreme (not common) conclusion or result. Even for our result in Figure (5), we still set various of fixed parameters, which are assumed no harm to our result. So this is also a limitation of numerical approach. On the contrary, Hennequin et al. (2018) explores more general scenarios with cases for instance, $n > 2$ for transfer function, with the analytical approach of response statistics of the SSN.

(10)

Speculate on how this stimulus-quenching of variability might influence neural coding for visual stimuli:

Stimulus-quenching of variability is usually represented by Fano factor (like the coefficient of variation, is a measure of the dispersion of a probability distribution of a Fano noise). And based on the research of Hennequin et al. (2018), neuronal recordings in visual areas have shown that Fano factors drop (stimulus-quenching of variability) at the onset of the stimulus (drifting gratings or plaids) in almost every neuron, which was well accounted for by the randomly connected network we studied above. However, in the experiments, variability did not drop uniformly across cells, but exhibited systematic dependencies on stimulus tuning. This could not be explained by randomly connected architectures, so the previous model should be extended to include tuning dependence in connectivity and input noise correlations, as the discussion below.

How to extend the model considered so far in order to address questions regarding coding for simple stimulus features such as orientation:

By introducing the architecture in which the preferred stimulus of E/I neuron pairs varied systematically around a "ring" representing an angular stimulus variable (stimulus orientation in V1), now we need to analyse the cases with different responds to different orientations (θ). we should introduce the varying external inputs u to represent different directions. First we consider the mean term. Under such a "ring model", following the structure discussed by Hennequin et al. (2018), each unit will receive the same constant mean input μ. The modified mean input to neuron i is represented in Equation (7).

$$\mu_i(\theta_{stim}) = b + c \cdot A_{max} \cdot \exp\left(\frac{\cos(\theta_i - \theta_{stim}) - 1}{\ell^2_{stim}}\right) \tag{7}$$

In which $b = 2mV$ is a constant that drives spontaneous activity (like an intercept term in linear function forms), as we discussed before. Then θ_{stim} is the orientation in the visual field, ℓ_{stim} is "half width" centered around θ_{stim} and scaled by a factor $c \in (0, 1]$ (increasing c indicates increasing stimulus contrast) times a maximum amplitude A_{max}. And assume for simplicity that E and I cells are driven equally strongly by the stimulus, though this could be relaxed. Note that the second term of the right is in a form of circular-Gaussian input bump.

Now we consider the associated noise term σ in a ring model. The noise has spatial structure, with correlations among neurons decreasing with the difference in their preferred directions following a circular-Gaussian, written in Equation (8).

$$\Sigma^{noise}_{ij} = \sigma_{\alpha(i)}\sigma_{\alpha(j)} \exp\left(\frac{\cos(\theta_i - \theta_j) - 1}{\ell^2_{noise}}\right) \tag{8}$$

Here, θ_i and θ_j are the preferred orientations of neurons i and j (E or I), and ℓ_{noise} is the correlation length.

The corresponding connectivity coeficients will also follow a proportional relationship with orientation θ as Equation (9).

$$W_{ij} \propto \sigma_{\alpha(i)}\sigma_{\alpha(j)} \exp\left(\frac{\cos(\theta_i - \theta_j) - 1}{\ell^2_{syn}}\right) \tag{9}$$

In equation, ℓ_{syn} sets the length scale over which synaptic weights decay.

Since both mean term and noise term of external input u as well as connectivity parameters W are modified in the form with orientation θ, when the visual networks receive stimulus of different orientations, different input will lead to different respond. Under such a orientational framework, we could analyse further (e.g., surround suppression and Cross-Orientation suppression).

(11)

How to distinguish the SSN model for stimulus-quenching of variability against these alternative hypotheses via new experiments or novel analyses of experimental data:

As introduced by Hennequin et al. (2018), the fundamentally different mechanisms responsible for variability modulation in the SSN, the multi-attractor, and the chaotic dynamical regimes should be revealed in the dynamics of **variability suppression at stimulus onset** and of **variability recovery at stimulus offset**. In this case, the experiment should measure the across-trial variability (averaged across neurons) following the onset and offset of a step stimulus in each model (parametrically varied the amplitude of the stimulus and therefore the degree of variability suppression). In detail, draw and analyse the timescale of variability suppression (time to reach half of the total suppression) or recovery timescales (time to recover half of the total suppression) as a function of the percentage of variance suppression in the three models, extracted from their corresponding variability trajectories.

As for potential novel analyses of experimental data, maybe we could also compute and record the Fano Factors (single total mean or even time window series) in the experiments above, to detect whether there is any difference on the factors between different models.

Reference

Ahmadian, Y., Rubin, D.B., and Miller, K. D. (2013). Analysis of the stabilized supralinear network. Neural Computation.

Hansell, D. and Van Vreeswijk, C. (2002). How Noise Contributes to Contrast Invariance of Orientation Tuning in Cat Visual Cortex. Journal of Neuroscience.

Hennequin, G., Ahmadian, Y., Rubin, D. B., Lengyel, M., and Miller, K. D. (2018). The dynamical regime of sensory cortex: Stable dynamics around a single stimulus-tuned attractor account for patterns of noise variability. Neuron, 98(4):846–860.e5.

Priebe, N.J., and Ferster, D. (2008). Inhibition, spike threshold, and stimulus selectivity in primary visual cortex. Neuron 57, 482–497.

Miller, K. D. and Troyer, T. W. (2002). Neural noise can explain expansive, power-law nonlinearities in neural response functions. Journal of Neurophysiology.

Rubin, D. B., VanHooser, S. D., and Miller, K. D. (2015). The stabilized supralinear network: A unifying circuit motif underlying multi-input integration in sensory cortex. Neuron.

Sanzeni, A., Akitake, B., Goldbach, H. C., Leedy, C. E., Brunel, N., and Histed, M. H. (2020). Inhibition stabilization is a widespread property of cortical networks. Elife, 9, e54875.

Appendix

Figure (1)

```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns

sns.set()
# mpl.use('pgf') # for latex

def phi(k, V, V0, n):
    VV = V - V0
    phi_v = (VV>0) * k * np.power(VV, n)
    return phi_v

k = 0.3
V0 = -70
n = 2
V = np.linspace(-75, -50, 100)

phi_V = phi(k=k, V=V, V0=V0, n=n)

fig, ax = plt.subplots(1, 1, figsize=(5,3.7))
ax.plot(V, phi_V, color='black')
ax.set_ylabel('$\phi(V)$ (Hz)')
ax.set_xlabel('$V$ (mV)')
ax.set_title('$\phi(V)$–$V$ curve')
plt.axvline(x = -70, ymin = -10, ymax = 130, linestyle = "dashed", color='black')
plt.text(-67,90,"$V_0=-70$mV",fontdict={'color':'black'})
plt.show()
# plt.savefig('1.pgf', format='pgf')
```

Figure (2)

```
def EIND(N_t, dt, tau_EI, V_rest, V_EI_init, W_EI, u_EI, V0, k, n):
    V_EI = np.zeros([N_t+1,2])
    V_EI[0,] = V_EI_init
    phi_V_EI = np.zeros([N_t,2])
    for t in range(1,N_t+1):
        phi_V_EI[t-1,] = phi(k, V_EI[t-1,], V0, n)
        dV_EI = -(V_EI[t-1,] - V_rest) + np.dot(W_EI, phi_V_EI[t-1,]) + u_EI
        V_EI[t,] = V_EI[t-1,] + np.divide(1,tau_EI) * dt * dV_EI
    return V_EI, phi_V_EI

N_t = 1000
dt = 1
tau_EI = np.array([20,10])
V_rest = -70
```

```
V_EI_init = np.array([V_rest,V_rest])
W_EI = np.array([[1.25,-0.65],[1.2,-0.5]])
u_EI1 = np.array([2,2])
u_EI2 = np.array([2,-2])
u_EI3 = np.array([-2,2])
u_EI4 = np.array([-2,-2])
V0 = V_rest
k = 0.3
n = 2

V_EI1, phi_V_EI1 = EIND(N_t, dt, tau_EI, V_rest, V_EI_init, W_EI, u_EI1, V0, k, n)
V_EI2, phi_V_EI2 = EIND(N_t, dt, tau_EI, V_rest, V_EI_init, W_EI, u_EI2, V0, k, n)
V_EI3, phi_V_EI3 = EIND(N_t, dt, tau_EI, V_rest, V_EI_init, W_EI, u_EI3, V0, k, n)
V_EI4, phi_V_EI4 = EIND(N_t, dt, tau_EI, V_rest, V_EI_init, W_EI, u_EI4, V0, k, n)

fig, ax =  plt.subplots(1,2, sharex = True, figsize=(7,3.7))
label1 = ['$E$(2,2)','$I$(2,2)']
label2 = ['$E$(2,-2)','$I$(2,-2)']
label3 = ['$E$(-2,2)','$I$(-2,2)']
color1 = ['#d65db1','#845ec2']
color2 = ['#ff9671','#ff6f91']
color3 = ['#f9f871','#ffc75f']
order = np.array([1,0])
for i in order:
    ax[0].plot(np.arange(0, N_t*dt, dt), V_EI2[1:,i], label = label2[i], \
               color = color1[i])
    ax[1].plot(np.arange(0, N_t*dt, dt), phi_V_EI2[:,i], label = label2[i], \
               color = color1[i])
for i in order:
    ax[0].plot(np.arange(0, N_t*dt, dt), V_EI1[1:,i], label = label1[i], \
               color = color2[i])
    ax[1].plot(np.arange(0, N_t*dt, dt), phi_V_EI1[:,i], label = label1[i], \
               color = color2[i])
for i in order:
    ax[0].plot(np.arange(0, N_t*dt, dt), V_EI3[1:,i], label = label3[i], \
               color = color3[i])
    ax[1].plot(np.arange(0, N_t*dt, dt), phi_V_EI3[:,i], label = label3[i], \
               color = color3[i])
ax[0].plot(np.arange(0, N_t*dt, dt), V_EI4[1:,0], label = '$E/I$(-2,-2)', \
           color = '#00c9a7')
ax[1].plot(np.arange(0, N_t*dt, dt), phi_V_EI4[:,0], label = '$E/I$(-2,-2)', \
           color = '#00c9a7')
ax[0].set_title('Simulation of $V_{E/I}(u_E,u_I)$')
ax[1].set_title('Simulation of $\phi(V_{E/I})(u_E,u_I)$')
ax[0].set_ylabel('$V_{E/I}$ (mV) / $\phi(V_{E/I})$ (Hz)')
ax[0].set_xlabel('Simulation time (ms)')
ax[1].set_xlabel('Simulation time (ms)')
ax[1].legend(loc='upper right')
plt.show()
# plt.savefig('2.pgf', format='pgf')
```

Figure (3)

```
def EIND_noise(N_t, dt, tau_EI, V_rest, V_EI_init, W_EI, mu_EI, sigma_EI, V0, k, n, seed):
    V_EI = np.zeros([N_t+1,2])
    V_EI[0,] = V_EI_init
    phi_V_EI = np.zeros([N_t,2])
    # introduce Gaussian white noise
    u_seed = np.random.RandomState(seed=seed)
    u_EI = mu_EI + 1/np.sqrt(dt) * sigma_EI * u_seed.randn(N_t,2)
    for t in range(1,N_t+1):
        phi_V_EI[t-1,] = phi(k, V_EI[t-1,], V0, n)
```

```
        dV_EI = -(V_EI[t-1,] - V_rest) + np.dot(W_EI, phi_V_EI[t-1,]) + u_EI[t-1,]
        V_EI[t,] = V_EI[t-1,] + np.divide(1, tau_EI) * dt * dV_EI
    return V_EI, phi_V_EI

N_t = 1000
dt = 1
tau_EI = np.array([20,10])
V_rest = -70
V_EI_init = np.array([V_rest, V_rest])
W_EI = np.array([[1.25, -0.65],[1.2, -0.5]])
mu_EI1 = np.array([2,2])
mu_EI2 = np.array([2,-2])
mu_EI3 = np.array([-2,2])
mu_EI4 = np.array([-2,-2])
sigma_EI = np.array([1,0.5])
V0 = V_rest
k = 0.3
n = 2
seed = np.array([1,2,3,4,5])

fig, ax =  plt.subplots(1,2, sharex = True, figsize=(7,3.7))
label1 = ['$E$(2,2)','$I$(2,2)']
color2 = ['#ff9671','#ff6f91']
order = np.array([1,0])
s = 4
V_EI_noise1, phi_V_EI_noise1 = \
    EIND_noise(N_t, dt, tau_EI, V_rest, V_EI_init, W_EI, mu_EI1, sigma_EI, \
               V0, k, n, seed[s])
for i in order:
    ax[0].plot(np.arange(0, N_t*dt, dt), V_EI_noise1[1:,i], label = label1[i], \
               color = color2[i])
    ax[1].plot(np.arange(0, N_t*dt, dt), phi_V_EI_noise1[:,i], label = label1[i], \
               color = color2[i])
for s in range(4):
    V_EI_noise1, phi_V_EI_noise1 = \
        EIND_noise(N_t, dt, tau_EI, V_rest, V_EI_init, W_EI, mu_EI1, sigma_EI, \
                   V0, k, n, seed[s])
    for i in order:
        ax[0].plot(np.arange(0, N_t*dt, dt), V_EI_noise1[1:,i], color = color2[i])
        ax[1].plot(np.arange(0, N_t*dt, dt), phi_V_EI_noise1[:,i], color = color2[i])
ax[0].set_title('5 Simulations of $V_{E/I}(u_E,u_I)$')
ax[1].set_title('5 Simulations of $\phi(V_{E/I})(u_E,u_I)$')
ax[0].set_ylabel('$V_{E/I}$ (mV) / $\phi(V_{E/I})$ (Hz)')
ax[0].set_xlabel('Simulation time (ms)')
ax[1].set_xlabel('Simulation time (ms)')
ax[1].legend(loc='upper right')
plt.show()
# plt.savefig('3.pgf', format='pgf')
```

Figure (4)

```
def EIND_mean_noise(N_t, dt, tau_EI, V_rest, V_EI_init, W_EI, mu_EI, sigma_EI, \
                    V0, k, n, seed, interval, noise=False):
    V_EI = np.zeros([N_t+1,2])
    V_EI[0,] = V_EI_init
    phi_V_EI = np.zeros([N_t,2])
    # introduce increasing mean
    u_EI = np.zeros([N_t,2])
    for i in range(mu_EI.shape[0]):
        u_EI[i*interval:(i+1)*interval] = mu_EI[i,]
    # introduce Gaussian white noise when noise = True
    if(noise==True):
```

```
        u_seed = np.random.RandomState(seed=seed)
        u_EI = u_EI + 1/np.sqrt(dt) * sigma_EI * u_seed.randn(N_t,2)
    for t in range(1,N_t+1):
        phi_V_EI[t-1,] = phi(k, V_EI[t-1,], V0, n)
        dV_EI = -(V_EI[t-1,] - V_rest) + np.dot(W_EI,phi_V_EI[t-1,]) + u_EI[t-1,]
        V_EI[t,] = V_EI[t-1,] + np.divide(1,tau_EI) * dt * dV_EI
    return V_EI, phi_V_EI

N_t = 9000
dt = 1
tau_EI = np.array([20,10])
V_rest = -70
V_EI_init = np.array([V_rest,V_rest])
W_EI = np.array([[1.25,-0.65],[1.2,-0.5]])
mu_EI = np.array([[0,0],[2,2],[15,15]])
sigma_EI = np.array([1,0.5])
V0 = V_rest
k = 0.3
n = 2
seed = 2032451
interval = 3000

V_EI_mean_noise_free, phi_V_EI_mean_noise_free = \
EIND_mean_noise(N_t, dt, tau_EI, V_rest, V_EI_init, W_EI, mu_EI, sigma_EI, \
                V0, k, n, seed, interval, noise=False)
V_EI_mean_noise, phi_V_EI_mean_noise = \
EIND_mean_noise(N_t, dt, tau_EI, V_rest, V_EI_init, W_EI, mu_EI, sigma_EI, \
                V0, k, n, seed, interval, noise=True)

fig, ax =  plt.subplots(2,2, sharex = True, figsize=(7,5.5))
label1 = ['$V_E$','$V_I$']
label2 = ['$\phi(V_E)$','$\phi(V_I)$']
u_EI = np.zeros([N_t])
color = ['#ff9671','#ff6f91']
max_mu = np.max(mu_EI[:,0])
max_V = np.max(V_EI_mean_noise[1:,])-np.min(V_EI_mean_noise[1:,])
max_phi = np.max(phi_V_EI_mean_noise[:,])-np.min(phi_V_EI_mean_noise[:,])
for i in range(mu_EI.shape[0]):
        u_EI[i*interval:(i+1)*interval] = mu_EI[i,0]/max_mu
for i in range(2):
    ax[0,0].plot(np.arange(0, N_t*dt, dt), V_EI_mean_noise_free[1:,i], \
                 label = label1[i], color = color[i])
    ax[1,0].plot(np.arange(0, N_t*dt, dt), phi_V_EI_mean_noise_free[:,i], \
                 label = label2[i], color = color[i])
    ax[0,1].plot(np.arange(0, N_t*dt, dt), V_EI_mean_noise[1:,i], \
                 label = label1[i], color = color[i])
    ax[1,1].plot(np.arange(0, N_t*dt, dt), phi_V_EI_mean_noise[:,i], \
                 label = label2[i], color = color[i])
ax[0,0].plot(np.arange(0, N_t*dt, dt), u_EI*(np.max(V_EI_mean_noise_free[1:,])-\
                                             np.min(V_EI_mean_noise_free[1:,]))-70,\
             linestyle = "dashed", color='#d65db1', label = "input")
ax[1,0].plot(np.arange(0, N_t*dt, dt), u_EI*(np.max(phi_V_EI_mean_noise_free[:,])-\
                                             np.min(phi_V_EI_mean_noise_free[:,])),\
             linestyle = "dashed", color='#d65db1', label = "input")
for s in range(9):
    V_EI_mean_noise, phi_V_EI_mean_noise = \
    EIND_mean_noise(N_t, dt, tau_EI, V_rest, V_EI_init, W_EI, mu_EI, sigma_EI, \
                    V0, k, n, s, interval, noise=True)
    max_V = np.max([max_V, np.max(V_EI_mean_noise[1:,])-\
                    np.min(V_EI_mean_noise[1:,])])
    max_phi = np.max([max_phi, np.max(phi_V_EI_mean_noise[:,])-\
                      np.min(phi_V_EI_mean_noise[:,])])
    for i in range(2):
```

```
        ax[0,1].plot(np.arange(0, N_t*dt, dt), V_EI_mean_noise[1:,i], \
                     color = color[i])
        ax[1,1].plot(np.arange(0, N_t*dt, dt), phi_V_EI_mean_noise[:,i], \
                     color = color[i])
ax[0,1].plot(np.arange(0, N_t*dt, dt), u_EI*max_V-70,\
             linestyle = "dashed", color='#d65db1', label = "input")
ax[1,1].plot(np.arange(0, N_t*dt, dt), u_EI*max_phi,\
             linestyle = "dashed", color='#d65db1', label = "input")
ax[0,0].set_title('Noise-free simulation')
ax[0,1].set_title('Noisy repeated simulations')
ax[0,0].set_ylabel('$V_{E/I}$ (mV)')
ax[1,0].set_ylabel('$\phi(V_{E/I})$ (Hz)')
ax[1,0].set_xlabel('Simulation time (ms)')
ax[1,1].set_xlabel('Simulation time (ms)')
ax[0,0].legend(loc='upper left')
ax[1,0].legend(loc='upper left')
ax[0,1].legend(loc='upper left')
ax[1,1].legend(loc='upper left')
plt.show()
# plt.savefig('4.pgf', format='pgf')
```

Figure (5)

```
def EIND_stat(N_t, dt, tau_EI, V_rest, V_EI_init, W_EI, mu_EI_minmax, sigma_EI, \
              V0, k, n, seed, grid, truncate):
    mean_V_EI = np.zeros([grid,2])
    mean_phi_V_EI = np.zeros([grid,2])
    std_V_EI = np.zeros([grid,2])
    std_phi_V_EI = np.zeros([grid,2])
    mu_EI = np.linspace(mu_EI_minmax[0,], mu_EI_minmax[1,], grid)
    for i in range(grid):
        V_EI = np.zeros([N_t+1,2])
        V_EI[0,] = V_EI_init
        phi_V_EI = np.zeros([N_t,2])
        # introduce Gaussian white noise
        u_seed = np.random.RandomState(seed=seed)
        u_EI = mu_EI[i,] + 1/np.sqrt(dt) * sigma_EI * u_seed.randn(N_t,2)
        for t in range(1,N_t+1):
            phi_V_EI[t-1,] = phi(k, V_EI[t-1,], V0, n)
            dV_EI = -(V_EI[t-1,] - V_rest) + np.dot(W_EI,phi_V_EI[t-1,]) + u_EI[t-1,]
            V_EI[t,] = V_EI[t-1,] + np.divide(1,tau_EI) * dt * dV_EI
        mean_V_EI[i,] = np.mean(V_EI[truncate:,],axis=0)
        mean_phi_V_EI[i,] = np.mean(phi_V_EI[truncate:,],axis=0)
        std_V_EI[i,] = np.std(V_EI[truncate:,],axis=0)
        std_phi_V_EI[i,] = np.std(phi_V_EI[truncate:,],axis=0)
    return mean_V_EI, mean_phi_V_EI, std_V_EI, std_phi_V_EI

N_t = 10000
dt = 1
tau_EI = np.array([20,10])
V_rest = -70
V_EI_init = np.array([V_rest,V_rest])
W_EI = np.array([[1.25,-0.65],[1.2,-0.5]])
W_EI_0 = np.array([[0.,0.],[0.,0.]])
mu_EI_minmax = np.array([[0,0],[20,20]])
sigma_EI = np.array([1,0.5])
V0 = V_rest
k = 0.3
n = 2
seed = 2032451
grid = 100
truncate = 500
```

```
mean_V_EI, mean_phi_V_EI, std_V_EI, std_phi_V_EI = \
    EIND_stat(N_t, dt, tau_EI, V_rest, V_EI_init, W_EI, mu_EI_minmax, sigma_EI, \
              V0, k, n, seed, grid, truncate)
mean_V_EI_W0, mean_phi_V_EI_W0, std_V_EI_W0, std_phi_V_EI_W0 = \
    EIND_stat(N_t, dt, tau_EI, V_rest, V_EI_init, W_EI_0, mu_EI_minmax, sigma_EI, \
              V0, k, n, seed, grid, truncate)

fig, ax =  plt.subplots(2,2, sharex = True, figsize=(7,6))
label1 = ['$V_E$','$V_I$']
label2 = ['$\phi(V_E)$','$\phi(V_I)$']
label21 = ['$V_E$ (W=0)','$V_I$ (W=0)']
label22 = ['$\phi(V_E)$ (W=0)','$\phi(V_I)$ (W=0)']
color = ['#ff9671','#ff6f91']
for i in range(2):
    ax[0,0].plot(np.linspace(mu_EI_minmax[0,0], mu_EI_minmax[1,0], grid), \
                 mean_V_EI[:,i], label = label1[i], color=color[i])
    ax[1,0].plot(np.linspace(mu_EI_minmax[0,1], mu_EI_minmax[1,1], grid), \
                 mean_phi_V_EI[:,i], label = label2[i], color=color[i])
    ax[0,1].plot(np.linspace(mu_EI_minmax[0,0], mu_EI_minmax[1,0], grid), \
                 std_V_EI[:,i], label = label1[i], color=color[i])
    ax[1,1].plot(np.linspace(mu_EI_minmax[0,1], mu_EI_minmax[1,1], grid), \
                 std_phi_V_EI[:,i], label = label2[i], color=color[i])
for i in range(2):
    ax[0,1].plot(np.linspace(mu_EI_minmax[0,0], mu_EI_minmax[1,0], grid), \
                 std_V_EI_W0[:,i], label = label21[i], \
                 linestyle = "dashed", color=color[i])
    ax[1,1].plot(np.linspace(mu_EI_minmax[0,1], mu_EI_minmax[1,1], grid), \
                 std_phi_V_EI_W0[:,i], label = label22[i], \
                 linestyle = "dashed", color=color[i])
ax[0,0].plot(np.linspace(mu_EI_minmax[0,0], mu_EI_minmax[1,0], grid), \
             mean_V_EI_W0[:,0], label = "W=0", \
             linestyle = "dashed", color='#d65db1')
ax[1,0].plot(np.linspace(mu_EI_minmax[0,1], mu_EI_minmax[1,1], grid), \
             mean_phi_V_EI_W0[:,0], label = "W=0", \
             linestyle = "dashed", color='#d65db1')
ax[0,0].set_title('Mean')
ax[0,1].set_title('Standard deviation')
ax[0,0].set_ylabel('$V_{E/I}$')
ax[1,0].set_ylabel('$\phi(V_{E/I})$')
ax[1,0].set_xlabel('$\mu$')
ax[1,1].set_xlabel('$\mu$')
ax[0,0].legend(loc='upper left')
ax[1,0].legend(loc='upper left')
plt.show()
# plt.savefig('6.pgf', format='pgf') # plot 5 eliminated
```