Modeling of propagating waves in primary visual cortex

Alain Destexhe's Laboratory for Computational Neuroscience

M2 Internship Report

July-August 2020

Axel Roques

Abstract

Propagating waves of neuronal activity have been identified in various brain structures, and in particular were found in the primary visual cortex of monkey performing visual tasks. It was shown that these waves improve the discrimination of ambiguous stimuli. It is therefore interesting to integrate these waves in computational models of the visual cortex. In this project, we developed a 2-dimensional mean field model of V1 to simulate propagating waves.

We will study the genesis of propagating waves from such networks, and how to adjust the model to better describe the propagating waves of experimental measurements. The model will be tested for a single stimulus, two stimuli (colliding waves) and more complex patterns of visual input.

1 Introduction

Voltage-sensitive dye imaging (VSDI) is an important recording tool for studying spatial localization of neocortical activity and dynamics [1]. This technique uses dye molecules that bind to the external surface of cell membranes and transform changes in membrane potential into optical signals. Thanks to its high temporal ($1 - 10ms$) and spatial ($< 50\mu m$) resolution over a large field-of-view ($1 - 2cm^2$), VSDI allows to measure neuronal activity at the mesoscopic scale [2].

One interesting application of this technique is the study of propagating waves of neuronal activity, where VSDI appears particularly suitable for the study of the spatiotemporal structure of activity in mesoscopic cortical regions. *Muller et. al.* showed that the stimulus-evoked population response in primary visual cortex of the awake monkey performing visual tasks propagates as a travelling wave, with consistent dynamics across trials [3]. *Chemla et. al.* further showed that these propagating waves are suppressive which leads to an increase in the acuity of the visual system when presented with ambiguous stimuli [4].

The aim of this study is to develop a theoretical model that accounts for these phenomena. While a detailed model of a neocortical column has been published [5], its computational cost does not allow its generalization to a $1 - 2cm^2$ spatial scale. The present model uses a mean-field formalism derived from networks of spiking neurons (adaptive exponential and fire AdEx model). This approach, proposed by *El Boustani and Destexhe* [6], thus describes the evolution of the population of neurons under a single 'pixel' of the camera. Previous work in the lab allowed for the construction of 1-dimensional mean-field ring models [7, 8]. Indeed, the ring geometry offers a simple framework to investigate the emergence of spatio-temporal patterns of activity [9]. This work aims to extend these models: populations of neurons will be linked with physiological connectivity profiles in a network with a 2-dimensional geometry, under mean-field formalism.

2 Material and methods

Here, we describe the equations and parameters used for our computational model.

Parameters	Parameter name	Symbol	Value	Unit
Cellular properties	Leak conductance	g_L	10	nS
	Leak reversal potential	E_L	-65	mV
	Membrane capacitance	C_m	200	pF
	AP threshold	V_{thre}	-50	mV
	Refractory period	τ_{refrac}	5	ms
Excitatory cell	Adaptation time constant	τ_w	500	ms
	Adaptation current increment	b	20	pA
	Adaptation conductance	a	4	nS
Inhibitory cell	Adaptation time constant	τ_w	10ˆ9	ms
	Adaptation current increment	b	0	pA
	Adaptation conductance	a	0	nS
Synaptic properties	Excitatory reversal potential	E_e	0	mV
	Inhibitory reversal potential	E_i	-80	mV
	Excitatory quantal conductance	Q_e	1	nS
	Inhibitory quantal conductance	Q_i	5	nS
	Excitatory decay	τ_e	5	ms
	Inhibitory decay	τ_i	5	ms
Network	Cell number	N_{tot}	10000	
	Connectivity probability	ϵ	1.25%	
	Fraction of inhibitory cells	g	20%	
	External drive	ν_e^{drive}	2	Hz
Model	Extent in x	L_{tot}	36	mm
	Extent in z	Z_{tot}	36	mm
	Excitatory connectivity radius	l_{exc}	5	mm
	Inhibitory connectivity radius	l_{inh}	1	mm
	Propagation delay	v_c	300	mm/s

Table 1: Simulation parameters.

2.1 Cellular models

The neuronal model used is the adaptive exponential and fire (AdEx) model. This model describes the dynamics of the membrane potential through an activation term with an exponential voltage dependence. Parameters for this model are identified in table 1 and fall into two categories, according to physiological insights: inhibitory neurons are represented as fast spiking (FS) neurons with no adaptation while excitatory regular spiking (RS) neurons have a lower level of excitability due to the presence of adaptation.

2.2 Network model

Simulations are based on a network of $N_{tot} = 10000$ neurons divided into two populations: 80% of excitatory neurons and 20% of inhibitory neurons. Every neuron is part of a random directed network where the probability of connection between two neurons is $\epsilon = 1.25\%$.

An external drive $\nu_e^{drive} = 2Hz$ is applied to bring the network out of the quiescent state and have spiking activity in the network.

2.3 Mean-field formalism

This model uses the mean-field approach developed by *El Boustani and Destexhe* [6]. Since VSDI techniques reveal structures and correlations over large distances (up to the centimetre scale), a mean-field formalism seems the most appropriate to handle the temporal and spatial scales of such optical imaging. The main hypothesis is to consider a Markovian dynamics for the network, *i.e.* to consider the system memoryless after a time T. We chose $T = 5ms$, which is large enough so that activity can be considered as memoryless and small enough so that each neuron can fire statistically less than once per time interval T.

The main idea behind this formalism is to derive differential equations for the average firing rate of excitatory (inhibitory) population ν_e (ν_i) from a master equation.

2.3.1 Master equation for local population dynamics

Let $\mu = e, i$ be the population activities (excitatory and inhibitory). Thus, the formalism describes the evolution of the instantaneous population firing rate ν_μ after binning in bins of $T = 5ms$. The following equation corresponds to its reduction to first order:

$$T\frac{\partial \nu_\mu}{\partial t} = \mathcal{F}_\mu(\nu_e, \nu_i) - \nu_\mu$$

Where $\mathcal{F}_\mu$ is the transfer function of a single neuron.

2.3.2 Transfer function

Expression. The transfer function of a neuron is the link between the output firing rate of this neuron and its excitatory and inhibitory inputs: $\nu_{out} = \mathcal{F}(\nu_e, \nu_i)$. While essential, this transfer function is hard to compute analytically. *Zerlaut et. al.* performed a semi-analytical derivation of the transfer function, using physiological experiments to deduce the different coefficients of the following formula:

$$\nu_{out} = \frac{1}{2\tau_V} Erfc\left(\frac{V_{thre}^{eff} - \mu_V}{\sqrt{2}\sigma_V}\right)$$

Where V_{thre}^{eff} is a phenomenological threshold that accounts for the single neuron non-linearities which was taken as a second order polynomial:

$$V_{thre}^{eff} = P_0 + \sum_{x\in\{\mu_V,\sigma_V,\tau_V^N\}} P_x\left(\frac{x - x^0}{\delta x^0}\right) + P_{\mu_G}\log\left(\frac{\mu_G}{g_L}\right) + \sum_{x,y\in\{\mu_V,\sigma_V,\tau_V^N\}^2} P_{xy}\left(\frac{x - x^0}{\delta x^0}\right)\left(\frac{y - y^0}{\delta y^0}\right)$$

Where μ_V, σ_V, τ_V are the mean, standard deviation and autocorrelation time constant of the membrane potential fluctuations ; normalization factors $\mu_V^0 = -60mV$, $\delta\mu_V^0 = 10mV$, $\sigma_V^0 = 4mV$, $\delta\sigma_V^0 = 6mV$, $\tau_V^0 = 10ms$ and $\delta\tau_V^0 = 20ms$.

Sub-threshold membrane potential fluctuations. This section details the calculations necessary to translate the neuron inputs into membrane potential fluctuations. The following formulas are based on the work of *Kuhn et. al.* [10].

$$\mu_{Ge}(\nu_e, \nu_i) = \nu_e K_e \tau_e Q_e$$
$$\mu_{Gi}(\nu_e, \nu_i) = \nu_i K_i \tau_i Q_i$$
$$\sigma_{Ge}(\nu_e, \nu_i) = \sqrt{\frac{\nu_e K_e \tau_e}{2}} Q_e$$
$$\sigma_{Gi}(\nu_e, \nu_i) = \sqrt{\frac{\nu_i K_i \tau_i}{2}} Q_i$$

Where $K_e = \epsilon(1 - g)N_{tot}$ the amount of excitatory connections onto a neuron and $K_i = \epsilon g N_{tot}$ the amount of inhibitory connections onto a neuron.

The mean conductances will control the input conductance of the neuron μ_G and therefore its effective membrane time constant τ_m:

$$\mu_G(\nu_e, \nu_i) = \mu_{Ge} + \mu_{Gi} + g_L$$

$$\tau_m(\nu_e, \nu_i) = \frac{Cm}{\mu_G}$$

We obtain the following formula for the mean voltage:

$$\mu_V(\nu_e, \nu_i) = \frac{\mu_{Ge}E_e + \mu_{Gi}E_i + g_L E_L}{\mu_G}$$

Further calculation (see *Zerlaut et. al.* [7]) gives:

$$\sigma_V(\nu_e, \nu_i) = \sqrt{\sum_s K_s \nu_s \frac{(U_s \tau_s)^2}{2(\tau_m^{eff} + \tau_s)}}$$

$$\tau_V(\nu_e, \nu_i) = \frac{\sum_s K_s \nu_s (U_s \tau_s)^2}{\sum_s \frac{K_s \nu_s (U_s \tau_s)^2}{\tau_m^{eff} + \tau_s)}}$$

Where $U_s = \frac{Q_s}{\mu_G}(E_s - \mu_V)$

These sets of equation translate the pre-synaptic frequencies into membrane fluctuations properties μ_V, σ_V, τ_V, which can then be fed into the transfer function to calculate the output firing rate of each neuron.

2.4 Models

We developed a 2-dimensional model to study population dynamics. This general model declines into two different types of geometry depending on their boundary conditions: a sheet and a torus. The sheet model has mirror boundary conditions, while the torus model has circular boundary conditions ; *i.e.* for every quantity f, $f(x + L) = f(x)$ and $f(z + L) = f(z)$, where L is the length of the extent of the ring model along the x or z axis.

Connectivity amongst neighbouring networks was defined using a Gaussian connectivity profile:

$$\mathcal{N}_e(x) = \frac{A_e}{\sqrt{2\pi} l_{exc}} \exp^{\left(-\frac{x}{\sqrt{2} l_{exc}}\right)^2}$$

$$\mathcal{N}_i(x) = \frac{A_i}{\sqrt{2\pi} l_{inh}} \exp^{\left(-\frac{x}{\sqrt{2} l_{inh}}\right)^2}$$

Where l_{exc} and l_{inh} are the excitatory and inhibitory extent of the connectivity profiles respectively. A_e and A_i are normalization factors for neighbouring excitatory and inhibitory networks. Both A_e and A_i are set experimentally - their role will be discussed in a separate section.

Note that a second version of the torus model has been developed, in which, on top of standard neighbouring Gaussian connections, random connection can occur between networks, with a weight that is manually assigned.

Using these 2D models, the neuronal activity in space and time follows:

$$\begin{cases} \nu_e^{input}(x,t) = \nu_e^{drive} + \int_{\mathbb{R}} \mathcal{N}_e(x-y) \nu_e(y, t - \frac{\|y-x\|}{v_c}) \, \mathrm{d}y \\ \nu_i^{input}(x,t) = \int_{\mathbb{R}} \mathcal{N}_i(x-y) \nu_i(y, t - \frac{\|y-x\|}{v_c}) \, \mathrm{d}y \\ T \frac{\partial \nu_e(x,t)}{\partial t} = \mathcal{F}_e \left(\nu_e^{aff}(x,t) + \nu_e^{input}(x,t), \nu_i^{input}(x,t) \right) - \nu_e(x,t) \\ T \frac{\partial \nu_i(x,t)}{\partial t} = \mathcal{F}_i \left(\nu_e^{input}(x,t), \nu_i^{input}(x,t) \right) - \nu_i(x,t) \end{cases}$$

Where ν_e^{drive} is the external drive and ν_e^{aff} the afferent thalamic stimulation. We introduced v_c, the axonal conduction speed, to take into account the delays in propagation due to finite axonal conduction speed.

We will use the variations of a normalized membrane potential quantity:

$$\delta V_N(x,t) = \frac{\mu_V(x,t) - V^{rest}}{V^{rest}}$$

Where V^{rest}, is the mean membrane potential during spontaneous activity in the model.

2.5 Afferent stimulation

We stimulated the 2-dimensional ring model with an external input mimicking thalamic stimulation:

$$\nu_e^{aff}(x,t) = A\exp^{-\left(\frac{x-x_0}{\sqrt{2}l_{stim}}\right)^2} \exp^{-\left(\frac{z-z_0}{\sqrt{2}l_{stim}}\right)^2} \left(\exp^{-\left(\frac{t-t_0}{\sqrt{2}\tau_1}\right)^2} \mathcal{H}(t_0-t) + \exp^{-\left(\frac{t-t_0}{\sqrt{2}\tau_2}\right)^2} \mathcal{H}(t-t_0)\right)$$

Where $\mathcal{H}$ is the Heaviside function.

The waveform profile in space is a 2-dimensional Gaussian curve of extent l_{stim} along the x and z axis. In time, the profile is a piecewise double Gaussian function. Such a waveform is well adapted since we can independently control: the maximum amplitude A of the stimulation, its rising time constant τ_1 and its decay time constant τ_2. It is important to note that this input does not propagate: its maximum is achieved at the same time for all position (x, z).

2.6 Code acceleration

To reduce computation time, Cython [11] was used. Cython is a compiler for Python that allows to write Python code with plain C performance by adding static type declarations to python variables.

3 User manual

This section will act as a "user's guide" for the model.

3.1 Architecture

Description of the different file types contained in the model's folder:

- ReadMe.txt: instructions to build the model
- RunMe.ipynb: Jupyter Notebook document
- .py files: plain python code
- .pyx files: cythonized python code
- figures folder: default output folder for plot functions
- data folder: default output folder for simulation output files. Also contains RS-cell_CONFIG1_fit.npy and FS-cell_CONFIG1_fit.npy which encodes values for the polynomial fit of V_{thre}^{eff}.
- Parameters.xlsx: a log of the different parameters that were tested for the model.

After building the .pyx files with the instruction in the ReadMe.txt file, .c and .pydfiles as well as a build folder will be created. They are files that are automatically generated by Cython for C acceleration

3.2 Detailed file description

We will not detail files that are automatically generated by Cython as we have no control over them. We will also consider .pyx files as regular .py files since they both can be interpreted as basic Python code (.pyx files simply have a static definition for some of the variables).

- cell_library.py: contains cellular parameters for RS and FS cells.
- synapses_connectivity.py: contains default parameters for synapses and connectivity.
- model_params.py: contains 2D-model default parameters.
- model_stim.py: contains default parameters for all sorts of stimulations: dots, lines, moving lines, a smiley,...
- mean_field.py: contains functions necessary for the mean-field approach.
- toolbox.py: contains numerous functions that facilitate computation, or plotting.
- connectivity_matrix_sheet.py, connectivity_matrix_torus.py, connectivity_matrix_torus_random.py: contain the connectivity matrix for the three kinds of models developed. The role of the connectivity matrix will be detailed further.

- sheet_model.pyx, torus_model.pyx, torus_random_model.pyx: actual core of the model, these files are responsible for the simulation.
- setup.py: necessary file for Cython ; dictates which files need to be cythonized. Is not necessary once the C files of the .pyx files are generated.
- RunMe.ipynb: main notebook, from which the stimulation can be launched and results can be plotted.

3.3 General scheme

Every simulation follows the same general scheme:

1. Loading parameters: synapses and cellular parameters are loaded and reformatted ; model parameters are loaded ; connectivity matrix is imported ; transfer functions are loaded.
2. Computing fixed point: mean-field equations are solved using the external drive as an input. This brings the system out of the quiescent state into the stationary state and generates initial conditions for $\nu_{e,0}$, $\nu_{i,0}$ and $\mu_{V,0}$.
3. Simulation start. For every time increment, we loop through every network in the model. The activity rate of the network is calculated by iterating over every one of its excitatory and inhibitory neighbouring networks. Delay due to limited axon conductivity is taken into account. Mean-field equations are then solved using the explicit Euler method.

3.4 Connectivity matrix

In order to decrease the time needed to conduct a simulation, a connectivity matrix was created. This matrix, unique for the sheet and the torus model, is an array of lists of dictionaries. Its size is the same as the size of the model, so that each element of the array corresponds to a single network at the same position in the model. Furthermore, each element of the connectivity matrix is a vector that gives information, in the form of a dictionary, on all the neighbours of this particular network. To be more specific, every dictionary has 4 arguments: the distance to this neighbour, its Gaussian weight according to its distance, and its x and z position in the model.

As an illustration, let's call the connectivity matrix M. Thus $M[1][9][5]$ returns the dictionary of the 5th neighbour from the network at $z = 1$ and $x = 9$. One can then access the informations contained in the dictionary using the dictionary keys 'dist', 'weight', 'pos_x', 'pos_z'.

Every connectivity matrix is constructed using model-specific boundary conditions, and is therefore unique to its model.

A connectivity matrix can be reconstructed from the Jupyter Notebook document. This generates a .npy file in the data folder that the model can easily call during simulations.

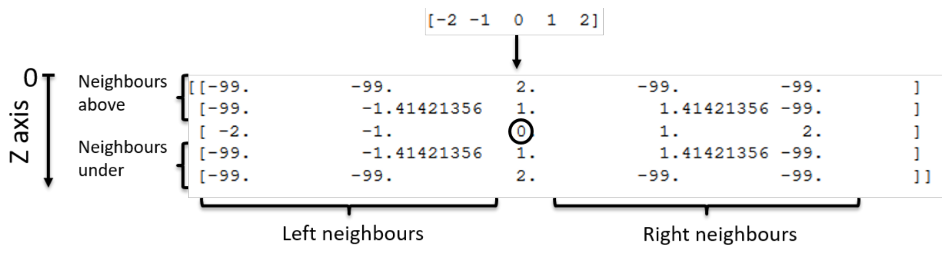
3.5 Main changes from 1D to 2D

Compared to Yann Zerlaut's 1D model [7], the 2D model differs on four main points.

3.5.1 Model parameters

Starting from a single x axis in 1D, a second axis z was added. This second axis was chosen to be the same length as the x axis for simplification sake, but its length can easily be modified.

Thus, the extent of the connectivity between neighbouring excitatory and inhibitory networks had to be made 2D-compatible. Figure 1 shows the upgrade from 1D to 2D.



```
[-2 -1  0  1  2]
```

```
[[-99.          -99.           2.          -99.           -99.        ]
 [-99.           -1.41421356   1.            1.41421356  -99.        ]
 [ -2.           -1.           0             1.             2.        ]
 [-99.           -1.41421356   1.            1.41421356  -99.        ]
 [-99.          -99.           2.          -99.           -99.        ]]
```

Figure 1: Neighbour's matrix for the extent of the inhibitory connections between networks.

Top matrix show the original connectivity between inhibitory networks. The 0 mimics the position of the network of interest and the surrounding non-zero number correspond to other inhibitory networks located at a distance equal to their absolute value (in pixels). In the 2D version, neighbours follow the same rule, except that neighbours that are too far away to be considered neighbours are assigned the value -99.

3.5.2 Stimulation initialization

Because of the 2D nature of the model, the initial stimulation had to be expanded along the z axis *via* addition of the term $exp^{-\left(\frac{z-z_0}{\sqrt{2}l_{stim}}\right)^2}$ in front of the 1D formula.

The 2D expansion allows for more interesting stimulations mimicking visual stimuli. In the model_stim.py file, many stimuli are represented in the get_stimulation function:

- 'CENTER': a single dot at the center of the image
- 'DOT': draws a single dot
- 'DOT-LEFT': a single dot slightly offset to the left
- 'DOT-RIGHT': a single dot slightly offset to the right
- 'DOUBLE': two single dots, at the same position as 'DOT-LEFT' and 'DOT-RIGHT', offset in time by 100ms
- 'MOV-HORZ-LINE': moving dots along a horizontal line; left to right - each dot that makes the line is offset in time from the previous one by 10ms
- 'MOV-VERT-LINE': moving dots along a vertical line, top to bottom, slightly offset to the right - each dot that makes the line is offset in time from the previous one by 2.5ms
- 'VERT-LINE': a vertical line, slightly offset to the right
- 'LINE-MOTION': a single dot followed by a vertical line, just like the famous line motion illusion
- 'DOUBLE_LINE': two moving horizontal lines going in opposite directions
- 'SMILEY': draws a smiley
- 'RECTANGLE': draws a rectangle
- 'CIRCLE': draws a circle
- 'RANDOM': draws many single dots at random x and z

3.5.3 Connectivity matrix and Euler method

This section of the algorithm is in charge of computing all the interactions between a network and its neighbours. The implementation of a second dimension greatly increases the computation time since it adds a nested loop to iterate over this new dimension. Furthermore, the position of neighbouring networks in a direction that is diagonal to the network of interest is not trivial to compute and boundary conditions require heavy 'if' statements. These supplementary tasks are costly for a programming language such as Python and when they are done on the spot, they result in simulation time of around 4 hours. That is why we chose a 'connectivity matrix' approach: information on every neighbour of each network is computed and stored in a matrix beforehand. Thus, when solving the activity rates using the Euler method, the algorithm simply has to look for the values of interest in the connectivity matrix.

This method produces great results, and decreases the simulation time from 4 hours to less than 30 minutes.

3.5.4 Numerical parameters

A Excel file 'Parameters.xlsx' can be found in the model folder. This file contains some evaluations of the model for different values of the numerical parameters. Compared to the 1D model, three main changes were made:

Connectivity probability ϵ, which dictates the internal circuitry between inhibitory and excitatory population inside every network, was changed from 5% to 1.25%. Indeed, a value of 5% makes the system unstable (figure 2).

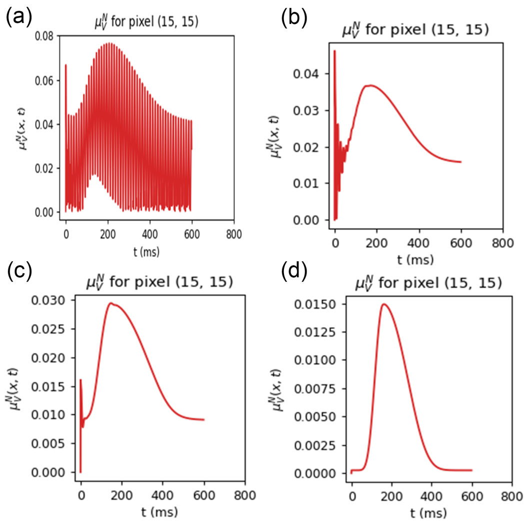


Figure 2: Variations in normalized membrane potential for different values of ϵ for the central pixel (position $x, z = 15$). Simulations realized in the sheet model with a 'CENTER' stimulation peaking at $200ms$. (a) $\epsilon = 10\%$, (b) $\epsilon = 5\%$, (c) $\epsilon = 2.5\%$, (d) $\epsilon = 1.25\%$.

Decreasing ϵ reduces parasitic oscillations until making the system stable for $\epsilon < 1.25\%$.

Drive stimulation ν_{drive} was changed from $2.5 - 4Hz$ down to $2Hz$ (figure 3).

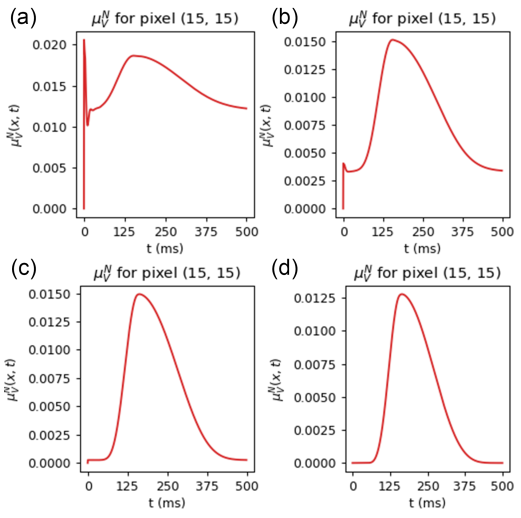


Figure 3: Variations in normalized membrane potential for different values of ν_{drive} for the central pixel (position $x, z = 15$). Simulations realized in the sheet model with a 'CENTER' stimulation peaking at $200ms$. (a) $\nu_{drive} = 5$, (b) $\nu_{drive} = 3$, (c) $\nu_{drive} = 2$, (d) $\nu_{drive} = 1$.

A high drive generates a burst of activity in the inhibitory population at the beginning of the simulation, which provokes these spikes observed in the normalized membrane potential. A drive of $2Hz$ was chosen instead of $1Hz$ because it produces great results and generates a higher response in amplitude for the membrane potential.

Connectivity normalization. This feature is not present in the 1D model. Because of the high number of added interactions in the 2D version, we need to add a pre-factor to the Gaussian connectivity between neighbours(A_e and A_i as described in section 2.4). Otherwise, the system is often unstable and the membrane potential diverges (see Excel file). Numerous pairs of values for A_e and A_i were tested and we retained two (figure 4).

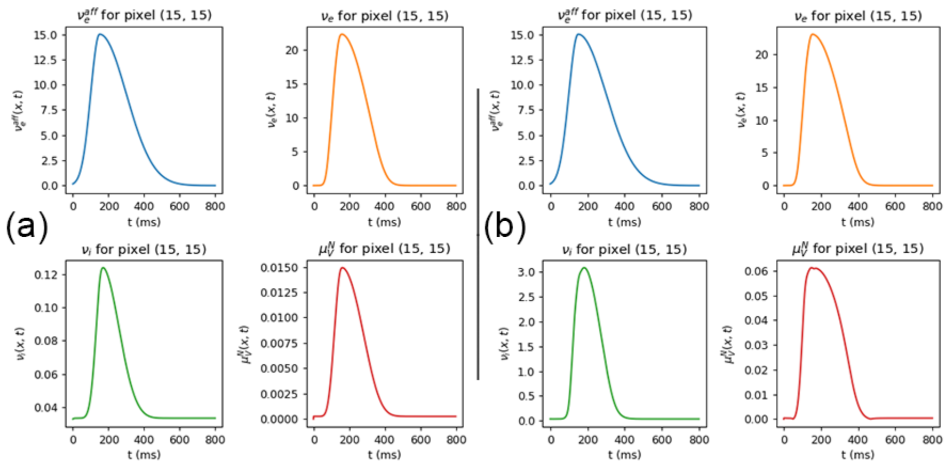


Figure 4: Plots over time of: afferent stimulation (blue curve), excitatory population activity rate (orange curve), inhibitory population activity rate (green curve) and normalized membrane potential (red curve), for different values of ν_{drive} for the central pixel (position $x, z = 15$). Simulations realized in the sheet model with a 'CENTER' stimulation peaking at $200ms$. (a) $A_e = 0,0631$ and $A_i = 0,3077$, (b) $A_e = 0.5$ and $A_i = 0.8$.

The values in (a) were calculated in a way that the sum of every neighbouring network would have a weight that is identical to the sum of the weight of neighbouring networks in the 1D model. While this normalization is effective and leads to sufficient results, the amplitude of ν_i is quite low which leads to a maximum change in membrane potential of 1.5%. In order to increase these amplitudes and reach amplitudes similar to the ones in 1D, *i.e.* around 5% for the normalized membrane potential, another pair of amplitudes were found manually , through trial and error. This new set of pre-factors results in higher amplitudes while keeping the system stable. However, as we will see later on, this set of A_e and A_i can lead to over-expression of the membrane potential which decreases the 'readability' of the model's output.

3.6 Complementary information

Any complementary information needed to use the model should be found in the RunMe.ipynb Jupyter Notebook file. If I forgot anything, do not hesitate to contact me at axel.roques@espci.fr or roquesaxel98@gmail.com, I will be happy to help :)

4 Results

This section will showcase some of the simulations that can be run using this 2D model. Results will be shown in both the sheet model and the torus model using the two pairs of 'connectivity normalization' values that were discussed before. For simplicity sake, let's call the connectivity values derived from the 1D model the 'base normalization', and the other couple of values resulting in an increase in amplitude the 'augmented normalization'.

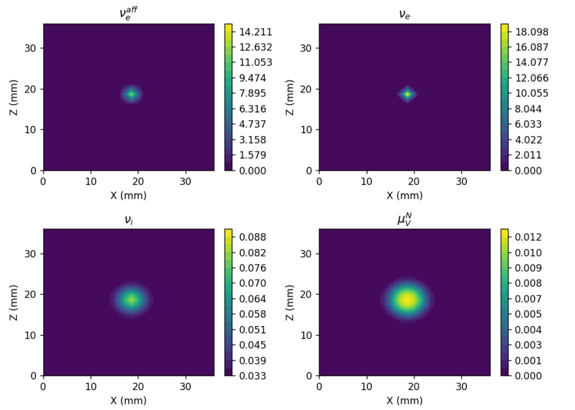
Films detailing the figures can be found in the folder /figures.

4.1 Simple stimulation

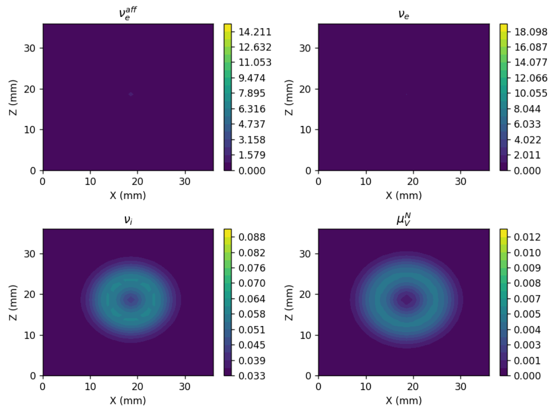
Simple visual stimuli were simulated. Afferent stimulation is described in section 2.5:

$$\nu_e^{aff}(x,t) = A \exp^{-\left(\frac{x-x_0}{\sqrt{2}l_{stim}}\right)^2} \exp^{-\left(\frac{z-z_0}{\sqrt{2}l_{stim}}\right)^2} \left(\exp^{-\left(\frac{t-t_0}{\sqrt{2}\tau_1}\right)^2} \mathcal{H}(t_0 - t) + \exp^{-\left(\frac{t-t_0}{\sqrt{2}\tau_2}\right)^2} \mathcal{H}(t - t_0)\right)$$

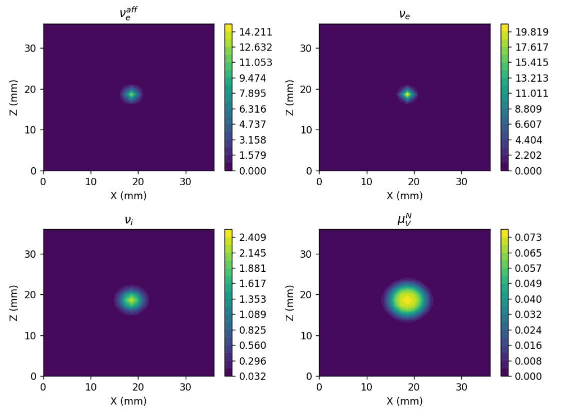
with parameters $l_{stim} = 1.5$ (network units), rising time $\tau_1 = 50ms$, decay time $\tau_2 = 150ms$ and amplitude $A = 15Hz$. This stimulation is centred, starts at $t = 150ms$ and ends at $t = 400ms$. Thus, peak stimulation amplitude is reached for $t = 200ms$ and starts decaying immediately after. Figure 5 and 6 show contour plots of the afferent stimulation ν_e^{aff}, the average activity rate for the excitatory population ν_e, the average activity rate for the inhibitory population ν_i and the normalized mean potential μ_V^N for both normalization factors at different times.



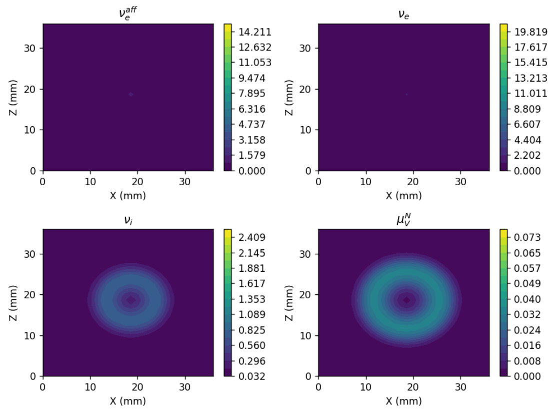
(a) 'base normalization', $t = 250ms$



(b) 'base normalization', $t = 500ms$



(c) 'augmented normalization', $t = 250ms$

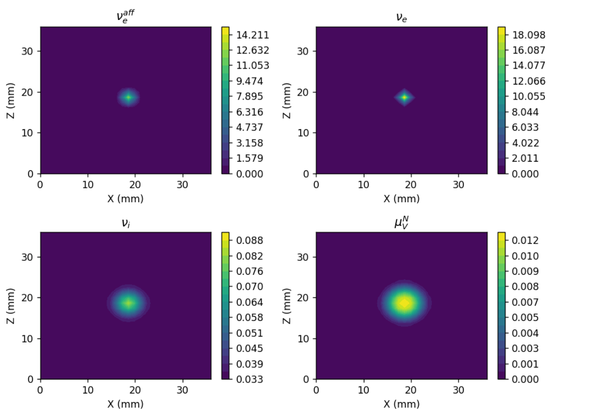


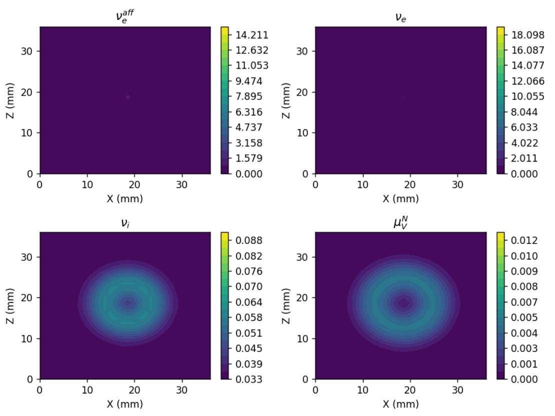
(d) 'augmented normalization', $t = 500ms$

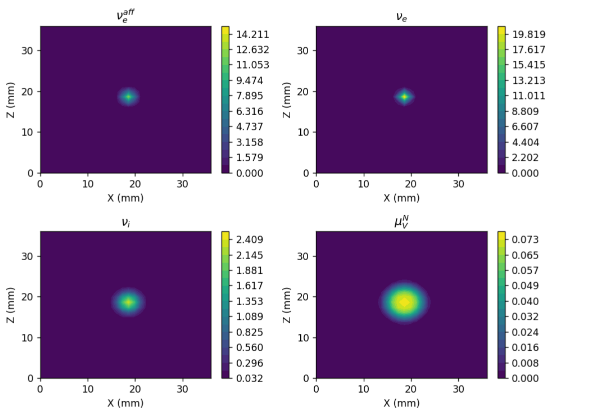
Figure 5: Contour plots of the afferent stimulation ν_e^{aff}, the average activity rate for the excitatory population ν_e, the average activity rate for the inhibitory population ν_i and the normalized mean potential μ_V^N for the sheet model. Top row are contour plots for the 'base normalization', bottom row are are contour plots for the 'augmented normalization'. Left column plots show system values for $t = 200ms$ at max afferent stimulation amplitude. Right column plots show system values for $t = 300ms$. Films for theses simulations can be found in the /figures folder under the names 'Sheet.mp4' and 'Sheet_Norm.mp4'.

We observe that activity rate for the excitatory population closely mimics the spatial extent of the afferent stimulation while the activity rate for the inhibitory population extends far further spatially. Overall, the model clearly shows signs of propagating wave as can be seen from the transition between $t = 200ms$ and $t = 300ms$.

Using the 'augmented normalization' doesn't seem to change the spatial extent of ν_e, ν_i or μ_V^N. However, the amplitude of ν_i and μ_V^N see a 30 fold and 5 fold increase respectively, and reach values that can be compared with the 1D model.


(a) 'base normalization', $t = 250ms$


(b) 'base normalization', $t = 500ms$


(c) 'augmented normalization', $t = 250ms$

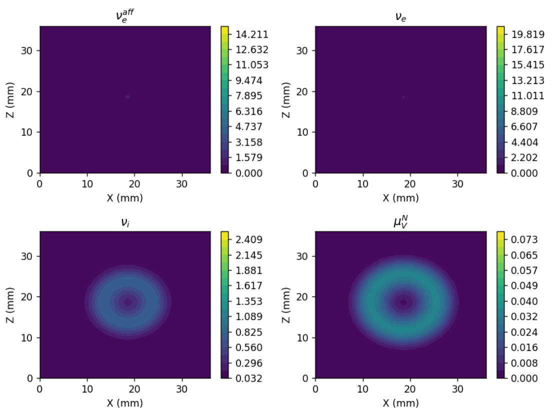

(d) 'augmented normalization', $t = 500ms$

Figure 6: Contour plots of the afferent stimulation ν_e^{aff}, the average activity rate for the excitatory population ν_e, the average activity rate for the inhibitory population ν_i and the normalized mean potential μ_V^N for the torus model. Top row are contour plots for the 'base normalization', bottom row are are contour plots for the 'augmented normalization'. Left column plots show system values for $t = 200ms$ at max afferent stimulation amplitude. Right column plots show system values for $t = 300ms$. Films for theses simulations can be found in the /figures folder under the names 'Torus.mp4' and 'Torus_Norm.mp4'.

The torus model shows similar results. Normalization has the same effect as in the sheet model.

4.2 Complex stimulations

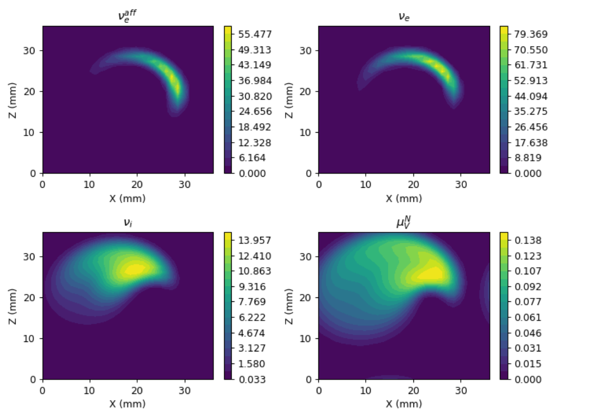
More complex visual stimuli were simulated using the torus model and the 'base normalization'. They consist in a series of 'dots' with the same equation as the afferent stimulation in the previous section. Each dot has the same previous parameters. Dots are drawn along a line to form a shape. Each dot is separated from the previous one by $10ms$, except for the 'smiley' shape where every dot is separated by $50ms$.

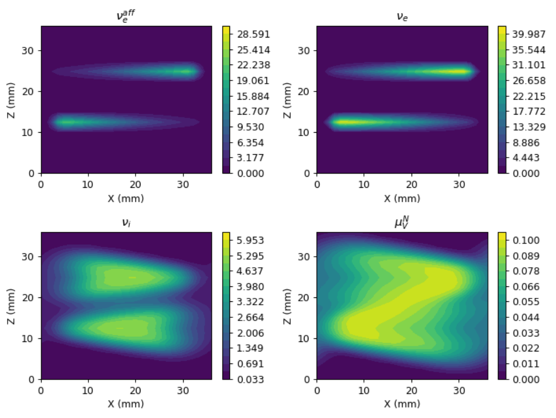
Figure 7 shows the contour plots associated with these complex stimulations.

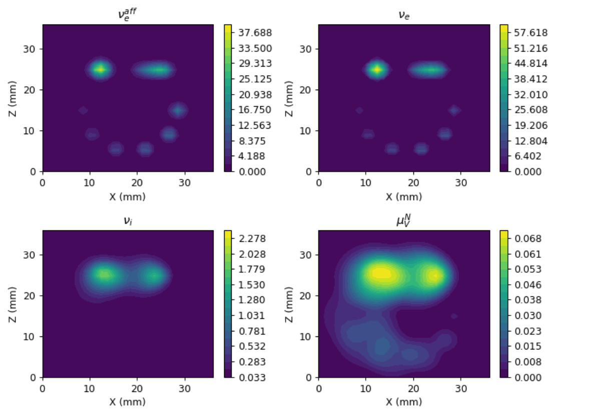
The model is easily capable to reproduce complex stimuli. Each dot from the input stimulation generates a propagating wave that can interact with other propagating waves.

One important thing to note is that the normalization factors chosen correspond to the 'base normalization'. Indeed, the close proximity of these waves adds up and the amplitude of the membrane potential sky rockets. Therefore, the 'augmented normalization' isn't recommended for these types of simulations.

Another remark is that while two propagating waves close to each other will result in a higher amplitude where the waves superimpose, the resulting wave is not the sum of the two initial waves. The


(a) Circle, $t = 600ms$


(b) Parallel lines, $t = 500ms$


(c) Smiley, $t = 400ms$

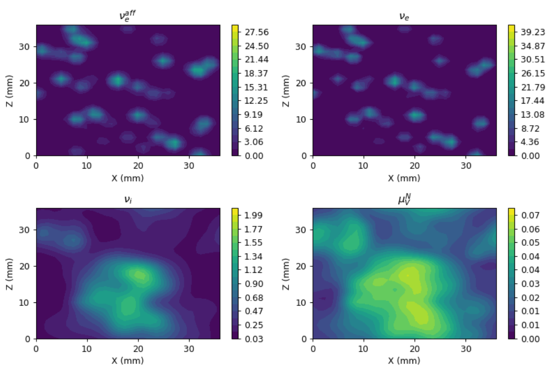

(d) Random dots, $t = 600ms$

Figure 7: Simulation of complex visual stimuli in the torus model with 'base normalization'. Contour plots are represented as described previously. (a) circle shape, imaged at $t = 600ms$; (b) two lines moving in the opposite direction, imaged at $t = 500ms$; (c) a smiley face, made out of an arc for the mouth, a small circle for one eye, and a movinf line for the other eye, imaged at $t = 400ms$; (d) random dots, a new dot appearing every $10ms$, imaged at $t = 600ms$. Films for theses simulations can be found in the /figures folder under the names 'Torus_Circle.mp4', 'Torus_Lines.mp4', 'Torus_Random.mp4' and 'Torus_Smiley.mp4'.

next section will develop this point.

4.3 Colliding waves

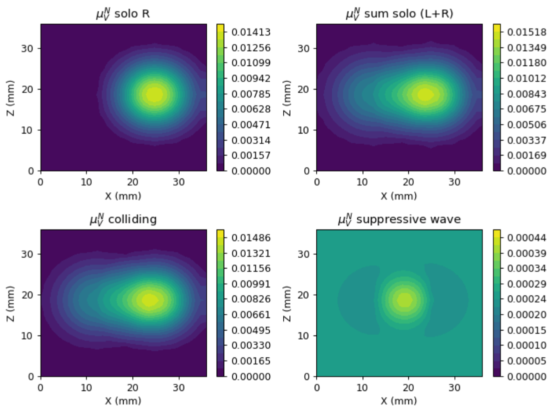
Chemla *et. al.* [4] showed that when two stationary dots are flashed in close spatial and temporal succession in awake monkeys, VSD imaging reveals that the collision of the two corresponding propagating waves generate a suppressive wave, that is also propagating. This suppressive wave is proposed to act as a way to help the visual system when presented with ambiguous stimuli.

In this section, we will try to reproduce these experimental results using the 2D model. Both models and both normalization were used.

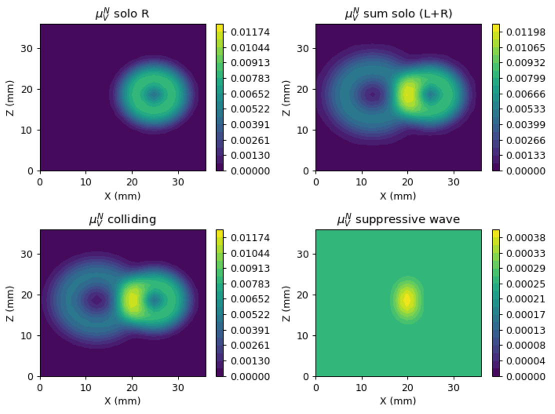
First, two simulations with stimulation from a stationary dot were conducted: one where the dot is offset to the left from the center, and a second where the dot is offset to the right. The latter had its stimulation start $100ms$ later compared to the first one, in order to recreate the close temporal succession mentioned in the article. Then, a simulation with the two previous dots simultaneously was done. The spatial extent and temporal succession between the dots was kept the same as in the individual simulations. Finally, using the data from the normalized membrane potential of these simulations, the suppressive wave was calculated. The suppressive wave simply corresponds to the difference between the mathematical sum of the normalized membrane potential from the simulations with the single dots and the normalized membrane potential resulting from the collision of the two propagating waves:

$$\mu^N_{V_{sum}} = \mu^N_{V_{soloR}} + \mu^N_{V_{soloL}}$$
$$\mu^N_{V_{suppressive}} = \mu^N_{V,sum} - \mu^N_{V_{colliding}}$$

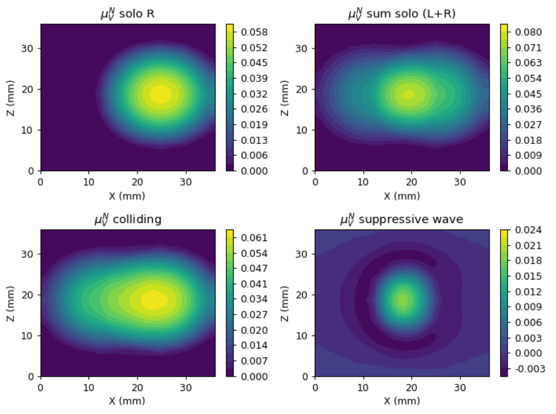
Figure 8 shows the contour plots of the normalized membrane potential for the different conditions.



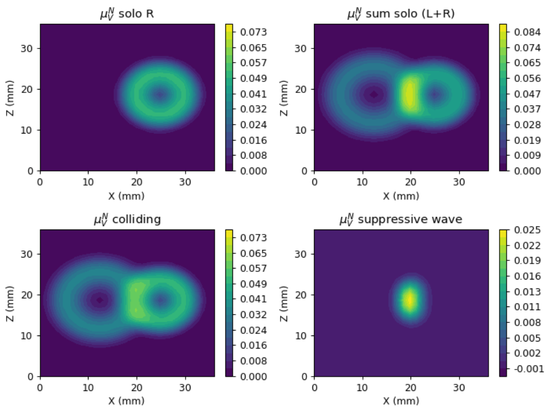
(a) Sheet model, $t = 300ms$, 'base normalization'



(b) Torus model, $t = 500ms$, 'base normalization'



(c) Sheet model, $t = 300ms$, 'augmented normalization'



(d) Torus model, $t = 500ms$, 'augmented normalization'

Figure 8: Contour plots of the normalized membrane potential for the sheet and torus models. Top row shows the behaviour of the models at $t = 300ms$ and $t = 500ms$ using the 'base normalization'. Bottom row shows the behaviour of the models at $t = 300ms$ for the sheet model and at $t = 500ms$ for the torus model using the 'augmented normalization'. For each model and each normalization is represented: top left = normalized membrane potential from the single dot stimulation (with offset to the right) ; top right = sum of the normalized membrane potential from the individual single dot simulations ; bottom left = normalized membrane potential from the colliding wave simulation ; bottom right = representation of the suppressive wave. Films for theses simulations can be found in the /figures folder under the names 'Sheet_Colliding.mp4', 'Sheet_Colliding_Norm.mp4', 'Torus_Colliding.mp4' and 'Torus_Colliding_Norm.mp4'.

We can clearly see that both the sheet and torus model exhibit a suppressive wave when two propagating waves are colliding. The shape of the suppressive wave resembles that of a propagating wave.

Using the 'base normalization', however, the amplitude of the suppressive wave is quite low and tends to blend with the background noise. This problem does not appear with the 'augmented normalization' in which the suppressive wave has a significantly improved signal to noise ratio. We suggest using the 'augmented normalization' for such experiments.

It appears that the model can reproduce the suppressive wave observed experimentally when two stationary dots are flashed in close spatial and temporal succession.

4.4 Line-motion illusion

Jancke *et. al.* studied the line-motion illusion [12]. It is a visual illusion where non-moving stimuli are perceived in motion. This particular illusion involves a stationary square briefly preceding a long stationary bar. They found that presenting a stationary square 60–100ms before a bar induced a dynamic activity patterns resembling that of fast movement. Using VSD imaging, they were able to visualize directly the cortical correlates of illusory line motion. In contrast to the conditions in which either the square or the bar was flashed alone, when flashing the square shortly before the bar regions of high optical activity did not remain stable but were gradually drawn out towards the end of the cortical bar representation.

Using the 2D sheet model, we were able to reproduce these experimental results. First, simulations where a dot or a bar was flashed alone were realized. Then, simulation with a moving bar was done (10ms between each dot that draws the line). Finally, a simulation where a dot was flashed followed 100ms later by a stationary bar was conducted. Simulations were conducted using both normalization.

See folder /figures/'Sheet_LineMotion_Norm.mp4' to see a movie of this simulation.

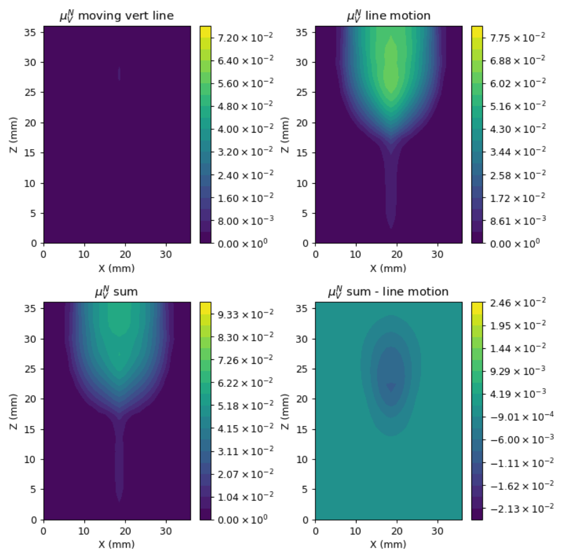
The sheet model's behavior is in accordance with the results from the paper. Compared to the propagating waves produced with either the single dot or the single bar, flashing the dot closely before the bar will make the bar appear to be drawn from top to bottom, like in the moving bar simulation.

The line motion illusion is a subject that has been extensively researched by various authors, notably Jancke [13] and Rangan [14]. It is interesting to note that in every model, the line motion illusion effect can be simulated without additional modeling of higher brain areas. However, both their models and ours have major differences. Rangan used a large-scale integrate-and-fire model of 10^6 neurons. This model is conductance-based: there are three types of channel conductances (NMDA, GABA, AMPA) and includes orientation specific long range connections. On the other hand, Jancke used a neural field model specifically designed to fit the line motion data and is limited to being an "abstract functional description of *in vivo* recorded VSD dynamics" [13]. The originality of our 2D model is to take into account the different gain between inhibition and excitation. These features are biologically relevant and are, for instance, the main elements determining wave suppression (such as in the experiments with the colliding waves).

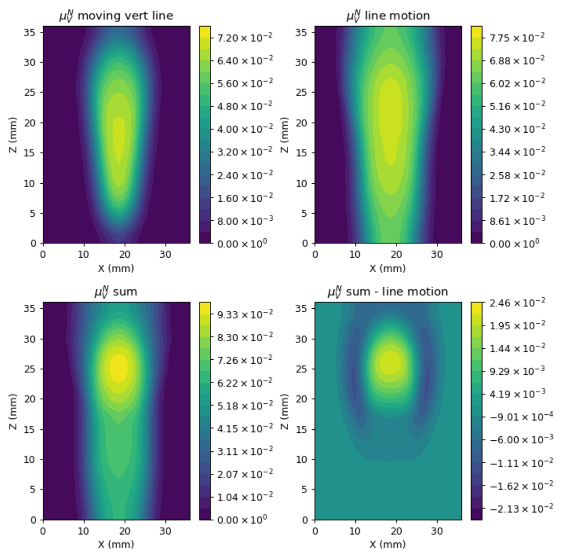
Moreover, there seems to be mechanistic differences that would explain the line motion effect in these models. Rangan insists on the importance of the long range NMDA-type cortical connections for the line motion illusion phenomenon in their model. Jancke noticed the apparition of non-linear 'facilitatory' and 'suppressive' intervals, more particularly a facilitatory effect in the beginning of the response. In our model it seems like a similar mechanism to what has been described in the *Jancke et al.* appears. Indeed, superposition of the responses to the single square and the bar alone differed from the response to their combined presentation in the line motion experiment 9.

Much like with Jancke's findings, no difference between the superposition and the line motion responses was found in the before the vertical line stimulus. Then, when the line starts to appear, a facilitatory effect is visible (blue shades in the contour plots). Following this facilitatory interval, a suppressive effect appears (yellow shades in the contour plots).

Therefore, our model seems to share similar non-linear dynamics with Jancke's model. An important advantage of our model compared to Jancke's is that our model has a much broader usage potential since it was not simply developed to reproduce line motion illusion data - notably, the genesis of suppressive waves from the collision of propagating waves can be simulated.



(a) $t = 470ms$



(b) $t = 550ms$

Figure 9: Contour plots of the normalized membrane potential for the sheet model under 'augmented normalization'. Top left = response of the model to the stimulation with a moving vertical line ; Top right = line motion experiment: a dot is flashed and $100ms$ later a vertical line appears ; Bottom left = superposition of the responses to the single square and the bar alone ; Bottom right = difference between the superposition and the line motion illusion responses. A movie of this figure can be found in the /figures folder under the name 'Sheet_LineMotion_Interactions.mp4'.

5 Discussion

In this project, a 2D mean field model of propagating waves was developed. This model uses a mean field description of two populations of cortical excitatory and inhibitory neurons. This local description of the population constitutes a network. The full model was obtained through spatial arrangement of such networks, in either a planar (sheet) geometry or in a torus shape. The model was able to correctly reproduce experimental VSD imaging data, for simple or complex inputs. To be more specific, localized inputs led to propagating-wave activity in a fashion that is similar to experiments. The model was able to recreate famous visual illusions where non-moving stimuli are perceived as moving, such as the line-motion experiment or the apparition of a suppressive wave when two stationary dots are flashed in close spatial and temporal succession.

Several extensions to the present model could be considered. First, this model is heavily based on Yann Zerlaut's 1D model [7]. Therefore, recent ameliorations made by Matteo di Volo [8] were not implemented - *i.e.* the current model does not take into account the presence of strong nonlinear effects such as conductance-based synaptic interactions, and spike-frequency adaptation. Such an implementation would make the model more 'biologically realistic' and thus more suitable to build very large scale models involving multiple brain visual areas.

Another important aspect is the time cost of this model. Computation time in this 2D model is relatively long (approximately 15 minutes for a 1s simulation). Cython has been used to speed up calculations, however the model can be 'Cythonized' further. Currently, only a handful of variables have been converted to C code. These are the variables that intervene in the core loop for the explicit Euler method. Further conversion of Python code into C code would make the model much faster. This would be a challenging work since it would require a good knowledge in the way Cython works, in order to convert only the most profitable parts of the code - but I believe that such a conversion could result in computation times decreasing to just a few minutes.

In the model's folder, one can find the code for an alternative version of the torus model which includes random connections between networks. While fully operational, this model has not been used a lot during this project. The potential of this model is yet to be found, perhaps it can be used to model other interesting phenomena.

Finally, a remarkable implementation would be to take into account the structure of V1, as an arrangement of hyper-columns composed of a set of orientation columns. While challenging, this would allow for a much more realistic model to describe higher mammal's V1. An interesting approach to this idea can be found in Lerchner *et. al.* [15].

References

[1] A. Grinvald *et al.*, "Vsdi: a new era in functional imaging of cortical dynamics," *Nature Reviews Neuroscience*, 2004.

[2] S. Chemla *et al.*, "Improving voltage-sensitive dye imaging: with a little help from computational approaches," *Neurophotonics*, 2017.

[3] L. Muller *et al.*, "The stimulus-evoked population response in visual cortex of awake monkey is a propagating wave," *Nature Communications*, 2014.

[4] S. Chemla *et al.*, "Suppressive traveling waves shape representations of illusory motion in primary visual cortex of awake primate," *The Journal of Neuroscience*, 2019.

[5] H. Markram *et al.*, "Reconstruction and simulation of neocortical microcircuitry," *Cell*, 2015.

[6] E. Boustani and Destexhe, "A master equation formalism for macroscopic modeling of asynchronous irregular activity states," *Neural Computation*, 2009.

[7] Y. Zerlaut *et al.*, "Modeling mesoscopic cortical dynamics using a mean-field model of conductance-based networks of adaptive exponential integrate-and-fire neuron," *Journal of Computational Neuroscience*, 2017.

[8] M. di Volo *et al.*, "Biologically realistic mean-field models of conductance based networks of spiking neurons with adaptation," *Neural Computation*, 2019.

[9] Hansel and Sompolinsky, "Chaos and synchrony in a model of a hypercolumn in visual cortex," *Journal of Computational Neuroscience*, 1995.

[10] A. Kuhn *et al.*, "Neuronal integration of synaptic input in the fluctuation-driven regime," *Journal of Neuroscience*, 2004.

[11] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith, "Cython: The best of both worlds," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 31–39, 2011.

[12] D. Jancke *et al.*, "Imaging cortical correlates of illusion in early visual cortex," *Letters to Nature*, 2004.

[13] D. Jancke *et al.*, "A dynamic neural field model of mesoscopic cortical activity captured with voltage-sensitive dye imaging," *PLoS Comput Biol*, 2010.

[14] A. Rangan *et al.*, "Modeling the spatiotemporal cortical activity associated with the line-motion illusion in primary visual cortex," *Proc Natl Acad Sci USA*, 2005.

[15] A. Lerchner *et al.*, "Mean field theory for a balanced hypercolumn model of orientation selectivity in primary visual cortex," *Network: Computation in Neural Systems*, 2005.