Numerical Investigation of One-Dimensional Stochastic Neural Field Equations with Delay

Madalena França Martins Rolão Nabais
madalena.nabais@ist.utl.pt

Instituto Superior Técnico, Lisboa, Portugal

December 2019

Abstract

Neural field equations (NFEs) are a particular type of integro-differential equations that model the spatiotemporal evolution of variables like synaptic activity in populations of neurons, which enables a detailed analysis of the dynamic behaviour of this type of populations. They play an important role in fields such as Robotics, notably in the creation of autonomous robots that interact with other agents in order to solve a mutual task and Neuroscience, where they serve as a method of interpreting experimental data, including information obtained from EEG, fMRI and optical imaging.
This paper analyses deterministic and stochastic one-dimensional NFEs, taking into account the transmission speed, noise and external stimulus, and presents a numerical method to solve them, which is properly tested for its convergence and complexity. The numerical method belongs to the class of Galerkin-type spectral approximations, relies on the trapezoidal rule to solve the integrals and in Euler-Maruyama's method to solve the system of stochastic differential equations. The originally proposed programming language MATLAB is compared with the up-and-coming Julia with respect to computational speed. Additionally, we compare the proposed algorithm with another one, that uses the Fast Fourier Transform. Finally, we present several simulations that illustrate the performance of the algorithm and analyse the important differences in behaviour induced by finite transmission speed, noise and when both factors co-exist.

Keywords: Stochastic neural field equations; Finite propagation speed; Galerkin method; Euler-Maruyama; Julia.

1. Introduction

1.1. Basic elements of Neuronal Systems

The nervous system is composed by the central nervous system (CNS) and the peripheral one (PNS) and contains two main types of cells: neurons and glial cells. The basic unit of the nervous system is the neuron, while glial cells are supporter cells that are required for energy supply, structural stabilization of brain tissue and protection of neurons. As glial cells do not affect directly the processing of information, we will focus only on neurons. Neurons are cells that use electrical and chemical signs to carry information throughout the human body and they can be decomposed in three parts: the **soma** (cell body) is responsible for processing information, **dendrites** are responsible for carrying information from other neurons to the soma (input) and lastly the **axon**, that carries the information from the soma to other neurons (output). To carry the information, neurons communicate between themselves and other cells by sending small electrical pulses, called action potentials, using specific connections called synapses, that are usually not immediate. The action potentials trigger the release of other neurotransmitters, which means that neurons communicate with each other by firing and that the firing of a single neuron can trigger the firing of the connected neurons, generating an intricate pattern of connectivity.

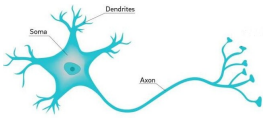

Figure 1: Single neuron drawing, adapted from [5]

1.2. Mathematical Modelling of Deterministic and Stochastic Neural Field Equations

The desire to mathematically model the behaviour of our brain and the dynamic populations of neurons within it, has lead to a growing interest in neural field models. As the brain is composed by populations of neurons, it is important to understand the connections between them (synapses) and how they are triggered (action potentials).

In 1952, A.H. Hodgkin and A.F.Huxley, see [11], authored a paper where it is described how action potentials in neurons are initiated and propagated, using the language of electric circuits and translating them into a system of four ordinary differential equations. Further investigation of the propagation of nervous stimulus lead in 1962 to the FitzHugh-Nagumo equations, [8], [9], [16], where the Hodgkin-Huxley system is reduced to two equations.

The high density of neurons and the number of synapses that occur in the cortex, suggest the creation of a continuum approximation model of neural activity. In 1956, a first attempt was made by Beurle, see [4]. In this model, it is considered a continuously distributed population of excitatory neurons, with a fixed threshold to send signals to each other. Moreover, by focusing on the fraction of neurons becoming active per unit of time, Beurle was able to analyse the triggering and propagation of large scale brain activity. In the 1970s, Wilson and Cowan [20] extended the theory to include both inhibitory and excitatory neurons, as well as refractoriness. In 1977, Amari [3] developed further work, particularly on the pattern formation under natural assumptions of connectivity and action potentials, thus providing the formulation of neural field models as we know them today.

We start by presenting the simplest model, the deterministic NFE without delay:

$$\frac{\partial U(x,t)}{\partial t} = I(x,t) - \alpha U(x,t) + \int_\Omega F(\mid x-y \mid) S(U(y,t))dy \quad (1)$$

where $t \in [0,T]$, $x \in \Omega \subset \mathbb{R}$.

On a neurological context, $U(x,t)$ is the membrane potential in point x at time t; I represents the external sources of excitation; $F(\mid x-y \mid)$ represents the connectivity strength between the different neurons x and y and can be positive or negative, depending on whether the connectivity is excitatory or inhibitory; S is the dependency between the firing rate of the neurons and their membrane potentials; α is a constant related to the decay rate of the potential. On a mathematical context, $U : \Omega \times [0,T] \longrightarrow \mathbb{R}$ is the unknown continuous function; I, F and S are given functions.

According to equation (1), the propagation speed of neuronal interactions is infinite. However, in reality, action potentials and synaptic processing can have delays in the order of milliseconds. In our context, the delay τ is defined as $\tau = \mid \frac{x-y}{v} \mid$ (the time spent by the signal to travel between x and y), $\tau_{max} = \frac{|\Omega|}{v}$ is the maximum value of the delay, where $\mid \Omega \mid$ is the cardinality of Ω and v is the propagation speed.

As we are always searching for the most consistent and realistic model that translates the phenomena that happens in our brain, we should also consider the effects of random processes in the system. We introduce the ideas used by Kühn and Riedler [14], to present the notion of noise and model a stochastic neural field equation (SNFE). The authors cite [6], [10] and [19] as motivation for the approach taken, since it is well known that intra and interneuron dynamics are subject to fluctuations [6] and many meso or macroscale continuum models have stochastic perturbations due to finite size effects [10], [19]. We consider the stochastic neural field equation (SNFE) with delay:

$$dU(x,t) = \epsilon dW(x,t) + [I(x,t) - \alpha U(x,t) + \int_\Omega F(\mid x-y \mid) S(U(y,t-\tau))dy]dt \quad (2)$$

where $W(x,t)$ is a Q-Wiener process and ϵ is a constant that describes the level of noise.

We will study different variations of these equations:

- NFE without delay (equation 1) with initial condition $U(x,0) = U_0(x)$
- NFE with delay (equation 2) with ϵ=0 and initial condition $U(x,t)$=$U_0(x,t)$, $t \in [-\tau_{max}, 0]$, $x \in \Omega$
- SNFE without delay (equation 2) with τ=0 and initial condition $U(x,0) = U_0(x)$
- SNFE with delay (equation 2) with initial condition $U(x,t)$=$U_0(x,t)$, $t \in [-\tau_{max}, 0]$, $x \in \Omega$

Notice that when we have a deterministic NFE, U_0 represents a given function whilst for a SNFE, U_0 represents a given stochastic process.

2. Numerical Method

2.1. Code Structure

The code implemented in this work is based on the algorithm developed by Kulikov and his co-authors (described in [1] and [2]), to solve the different equations presented at the end of **1.2**.

We start by using the Karhunen-Loéve formula to expand the solution $U(x,t)$ into the form:

$$U(x,t) = \sum_{k=0}^{\infty} u_k(t) v_k(x) \quad (3)$$

where $v_k(x)$ are the eigenfunctions of the covariance operator of the noise term in equation (2). These eigenfunctions establish an orthonormal basis in the Hilbert space associated to the problem under consideration. Knowing this, we need a formula to calculate the coefficient functions $u_k(t)$, which are random distributions. It is important to observe that expanding our solution into the form (3) using the Karhunen-Loéve formula, allows the integrations in time and in space to be fulfilled separately.

In order to derive a formula for the coefficient functions, we take the inner product of equation (2) with the basis functions $v_k(x)$:

$$< dU(x,t), v_k(x) >= [< I(x,t), v_k(x) > - \alpha < U(x,t), v_k(x) > + (\int_\Omega < F(\mid x-y \mid) S(U(y,t-d(x,y)))dy, v_k(x) >]dt + \epsilon < dW(x,t), v_k(x) > \quad (4)$$

The delay τ in equation (2) is space-dependent, thus, here and below we represent τ as $d(x,y)$.
We follow Kühn and Riedler's approach [14] and define the inner product of any two functions in the integral form as:

$$< f(x), g(x) >= \int_\Omega f(x)g(x)dx \quad (5)$$

Furthermore, as $W(x,t)$ is a Q-Wiener process, we can expand $dW(x,t)$ as:

$$\mathrm{d}W(x,t) = \sum_{k=0}^{\infty} v_k(x)\lambda_k d\beta_k(t) \quad (6)$$

where $\beta_k(t)$ are a sequence of independent scalar Wiener processes and λ_k are the eigenvalues of the covariance operator Q, which defines the Q-Wiener process.
To simplify the algorithm, we deal with a particular case, described by [14] p.7, where the correlation function obeys the rule:

$$E\{W(x,t)W(y,s)\} = min(t,s)\frac{1}{2\xi}\exp(\frac{-\pi}{4}\frac{\mid x-y \mid^2}{\xi^2})$$

for any space points and $x, y \in \Omega$ and time instants $t, s > 0$. The $E\{\cdot\}$ refers to the expectation operator and the fixed parameter ξ determines the spatial correlation length.
Following [18], we choose $\xi << 2L$ and derive the eigenvalues λ_k of the covariance operator Q in the SNFE (2), which satisfy the formula:

$$\lambda_k^2 = \exp(-\frac{\xi^2 k^2}{4\pi})$$

Taking into account the orthonormality of the basis functions $v_k(x)$, the inner product in the form (5) and expansion (6), substituting it in equation (4) yields the formula for describing the evolution of the coefficient functions $u_k(t)$ as follows:

$$du_k(t) = [< I(x,t), v_k(x) > -\alpha u_k(t) + FS_k(U(y,t-d(x,y)))]dt + \epsilon\lambda_k d\beta_k(t) \quad (7)$$

where $FS_k(U(y,t-d(x,y)))$ denotes the delay dependent non-linear term in the stochastic differential equation. This term also links all SDEs (7) with different k's into a unified system that is required to be solved simultaneously. Using the expansion of the solution presented in (3) and the inner product definition (5) the term $FS_k(U(y,t-d(x,y)))$ is evaluated as:

$$FS_k(U(y,t-d(x,y))) = \int_\Omega v_k(x)[\int_\Omega F(\mid x-y \mid) \times S(\sum_{l=0}^{\infty} u_l(t-d(x,y))v_l(y))dy]dx \quad (8)$$

When using the Karhunen-Loéve formula, we obtain an infinite expansion of the solution $U(x,t)$. In order to truncate this infinite series we use a Galerkin-type approximation, i.e we choose a sufficiently large positive number K and replace the exact solution $U(x,t)$ to equation (2) with a finite approximation:

$$U(x,t) = \sum_{k=0}^{K} u_k(t)v_k(x) \quad (9)$$

In equation (9), we obtain a truncated series of the infinite expansion, however, this equation does not account for the continuous-time properties of the eigenfunctions $v_k(x)$ and the coefficient functions $u_k(t)$.
In this paper we adapt the eigenfunctions considered by Kühn and Riedler [[14], Example 2.1] and assume:

$$v_k(x) = \begin{cases} \frac{1}{\sqrt{2L}}, & \text{if } k=0 \\ \frac{1}{\sqrt{L}}\cos(\frac{k\pi x}{L}), & \text{if } k \geq 1 \end{cases} \qquad k = 0, 1, \cdots, K$$

which have been accommodated to deal with the delay condition. This orthonormal basis is of the conventional Fourier type, where all the sinusoidal eigenfunctions of the Fourier series are absent, because if the functions I and F are even, the exact solution to the SNFE is also an even function. At the same time, the coefficient functions $u_k(t)$ obey SDEs (7). Subsequently, we need to adapt the rest of our results to the approximation (9). To do this, we have to discretize the eigenfunctions $v_k(x)$ in space and derive a discretized version of equations (7). To implement these discretization steps, we start by introducing an equidistant space mesh:

$$\mathbf{x} := \{x_i = x_0 + ih_x, \quad i = 0, 1, \cdots, N\}$$

where x_0=-L, x_N=L, $h_x = \frac{2L}{N}$ (step size in space) and where N is a chosen number of the prefixed discretization steps in the space interval [-L,L]. This subdivision $\mathbf{x}$ of the space interval is demanded to satisfy the condition $N >> K$, that is, the number of space-discretization steps utilized must be much larger than the number of basis functions in the Galerkin approximation (9). In this way, we

significantly improve the accuracy of the numerical integration.
Having applied the Galerkin approximation and, then, replaced all the space-dependent functions in formulas (7) and (8) with their values calculated at the mesh nodes $\mathbf{x}$, we obtain the space-discretized SDEs of the form:

$$du_k(t) = [< I(\mathbf{x},t), \mathbf{v}_k > -\alpha u_k(t) + FS_k(U_k(\mathbf{x}, t-d(\mathbf{x},\mathbf{x})))]dt + \epsilon\lambda_k d\beta_k(t) \quad (10)$$

where $\mathbf{x}$ is the $(N+1)$-dimensional vector in which $\mathbf{x}(i) = x_0 + ih_x$ and $\mathbf{v}_k := v_k(\mathbf{x})$. Here and below, we take $\mathbf{v}_k$ as the $(N+1)$-dimensional vector with its entries $v_k(x_i)$, i=0,1,$\cdots$,N .
To complete the discretization, we replace all integrals with quadrature rules, in our case, the trapezoidal rule. Applying the quadrature rule to $< I(\mathbf{x},t), \mathbf{v}_k >$ and $FS_k(U_k(\mathbf{x}, t-d(\mathbf{x},\mathbf{x})))$ yields:

$$< I(\mathbf{x},t), \mathbf{v}_k >= h_x \sum_{i=0}^{N-1} I(x_i,t)v_k(x_i) \quad (11)$$

$$FS_k(U_k(\mathbf{x}, t-d(\mathbf{x},\mathbf{x}))) = h_x^2 \sum_{i=0}^{N-1} v_k(x_i) \times [\sum_{j=0}^{N-1} F(\mid x_j - x_i \mid) S(\sum_{l=0}^{K} u_l(t-d(x_i,x_j))v_l(x_j))] \quad (12)$$

where (11) and (12) are utilized in the system (10) for any subscript k=0,1,$\cdots$,K.
Notice that it is taken into account that all the integrands are even functions and therefore, as the values at the initial and last points of the introduced mesh coincide, we double the corresponding values at the initial mesh node and reduce our summation index ranges by one in the mentioned integral approximations. The system in (10) can be represented in a vectorized form, for each k=0,1,$\cdots$,K as:

$$d\mathbf{u}(t) = [h_x\mathbf{V} \times I(\mathbf{x},t) - \alpha\mathbf{u}(t) + \mathbf{FS}(U_k(\mathbf{x}, t-d(\mathbf{x},\mathbf{x})))]dt + \epsilon d\mathbf{W} \quad (13)$$

where $\mathbf{u}(t) = (u_0(t), u_1(t), \cdots, u_K(t))^T$, $\mathbf{V}$ is the rectangular matrix of size $(K+1)\times N$ whose (k,i)-entry is computed as $v_k(x_i)$ with k=0,1,$\cdots$,K and i=0,1,$\cdots$,N-1. $\mathbf{FS}(U_k(\mathbf{x}, t-d(\mathbf{x},\mathbf{x})))$ represents the $(K+1)$-dimensional column-vector whose kth entry is calculated by formula (12) and finally d$\mathbf{W}$ is the zero-mean white Gaussian process with the diagonal covariance matrix Λdt in which Λ is a diagonal matrix where $\Lambda = diag\{\lambda_0^2, \lambda_1^2, \cdots, \lambda_K^2\}$.
It is important to notice that the last entries of the vectors v_k are excluded from matrix $\mathbf{V}$ because of the explanation given above about the trapezoidal rule.
Now, to solve system (13) we can simply use a commonly known method for solving SDEs. We choose the explicit Euler-Maruyama method which is applied to an equidistant mesh with sufficiently small step size h_t, prefixed in the time interval [0,T] of SNFE (2).

2.2. Some notes on implementation:

2.2.1 Delay-free case

We are simply considering equations (13) with $d(x,y)$=0 for all $x, y \in \Omega$, since we are in the delay-free case. We can compute the non-linear term $\mathbf{FS}(U_k(\mathbf{x},t)$ efficiently by:

$$\mathbf{FS}(U_k(\mathbf{x},t)) = h_x^2\mathbf{V} \times \mathbf{F} \times \mathbf{s} \quad (14)$$

where $\mathbf{F}$ is the symmetric matrix of dimension $(N \times N)$ whose (i,j)-entry means $F(\mid x_j - x_i \mid)$ with i,j=0,1,$\cdots$,N-1 and $\mathbf{s}$ is a time-variant column-vector of size N and its ith entry is evaluated by the formula $S(\mathbf{u}^T(t)\mathbf{V}_i)$ at each time instant t used, where $\mathbf{V}_i$ stands for the ith column of matrix $\mathbf{V}$.
Observe that the biggest computational effort is the calculation of the term $h_x^2\mathbf{V} \times \mathbf{F}$ which is time-invariant and thus, it can be pre-computed before starting solving the system of SDEs by the Euler-Maruyama scheme. In conclusion, at each time step in our time integration of SDEs (13) with $d(x,y)$=0, we need only to evaluate vector $\mathbf{s}$ and multiply it by the already computed $h_x^2\mathbf{V} \times \mathbf{F}$, which saves a significant amount of computational work.

2.2.2 Delay-dependent case

When performing simulations with this algorithm, in the case of infinite transmission speed, one can set the time step h_t and space step h_x independently. In contrast, when delay is considered, we have to ensure that the value $(t-d(x_i,x_j))$ belongs to the time discretization mesh accepted in the numerical simulation. By setting the time step size by the formula

$$h_t = \frac{h_x}{v}$$

we guarantee this condition, i.e for each pair of nodes (x,y) belonging to the mesh, there exists an integer a, such that $d(x,y)$=ah_x=avh_t.
When there is delay, it may happen that $(t - d(x,y)) < 0$. In this case, the value of $U(x, t_j - d(x,y))$ will depend on the initial condition of the deterministic or stochastic equation, which is also discretized with the predefined step size h_x and h_t satisfying the required relation in the space-time domain $[-L, L] \times [-\tau_{max}, 0]$.

In this case, we consider equations (13). The evaluation formula of $\mathbf{FS}(U_k(\mathbf{x}, t - d(\mathbf{x}, \mathbf{x})))$ is given by:

$$\mathbf{FS}(U_k(\mathbf{x}, t - d(\mathbf{x}, \mathbf{x}))) = h_x^2 \mathbf{V} \times (\mathbf{F} \circ \mathbf{S}) \qquad (15)$$

where matrix $\mathbf{V}$ and matrix $\mathbf{F}$ have already been defined. $\mathbf{S}$ is the time-variant square matrix $(N \times N)$ whose (i, j)-entry is the value of $S(U_k(x_j, t - d(x_i, x_j)))$ already computed in past time $t - d(x_i, x_j)$. In formula (15), $\circ$ refers a specifically defined multiplication of two matrices. This product returns a column-vector whose ith entry is obtained by the inner product of the ith row of the first matrix with the ith column of the second one.
Here and below, we consider N as the chosen number of the prefixed discretization steps in the space interval [-L,L]; K is the number of basis functions; n is the chosen number of the prefixed discretization steps in the time interval [0,T]; ns is the number of paths/trajectories.

3. Choosing the programming language

It is of utter importance for the programming language chosen to be as efficient as possible, since the algorithm presented contains very expensive computations, such as operations involving high dimensional matrices, summations and a complex chain of iterative loops to obtain the final solution.
When solving numerical problems, interpreted languages such as MATLAB are preferred as they are simple to program in, offer easy ways to plot data and do complex mathematics. However, with these programs, the speed is usually not very high. Recently, a high-level programming language called Julia was developed, which provides the ease and expressiveness of high level numerical computing (in the same way as languages as MATLAB), but also supports general programming, having the ability to be as efficient as lower level languages such as C/C#. As Julia promises faster computations, using a high syntax language and having a graphical output (still being built) but very similar to MATLAB, for the particular uses we need in this text, the results will be presented coded in Julia. Nonetheless, we will perform a comparison between the two programming languages to support our analysis.
To test the differences between the computing time of both languages, we used the functions described in the introduction of Section **5**, with n=200 for the delay-free case, N and K varying. For the case with delay, to satisfy the formula $h_t = \frac{h_x}{v}$, n also varies. The tests were only performed for the cases without noise because each iteration is made independently, thus, in order to obtain the performance of the cases with noise, we simply need to multiply the elapsed time for one trajectory by the number of trajectories computed.

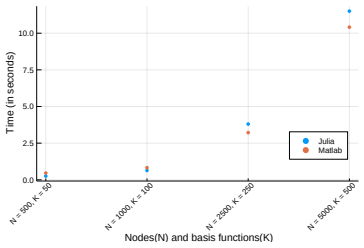

Figure 2: Comparing performance of programming languages: Time performance of Julia (Blue) and MATLAB (Red) in the case without delay

By Figure 2, we observe that for the first pair of (N,K) the Julia implementation is slightly quicker, but with a larger number of nodes MATLAB becomes faster. In spite of this, the difference is around 0.5 to 1 second, hence we could use either MATLAB or Julia for our implementation in the delay-free case. As for the computing time in the delay case (Table 1), Julia proves slightly faster for bigger values of (N,K).

$(N,\ K)$	n	Performing time (in seconds) for Julia	Performing time (in seconds) for MATLAB
(5000,500)	2000	4725.89	4818.22
(2500,250)	1000	662.44	722.35
(1000,100)	400	54.69	52.35
(500,50)	200	5.97	4.59

Table 1: Comparing performance of programming languages: Time performance of Julia and MATLAB in the case with delay

4. Comparing algorithms

In order to evaluate the efficiency of our method, we will compare it with another algorithm which was proposed by Hutt and Rougier ([12], [13]). The authors suggest a numerical method that is based on the Fast Fourier Transform (FFT) and solves deterministic NFEs with delays in two dimensions. Due to this, the code was adapted to one dimension to be in accordance with this text (this was done by Master's Student Tiago Sequeira [17]). Henceforth, Algorithm 1 represents the algorithm implemented in this text and Algorithm 2 represents the one-dimensional version of the algorithm proposed by Hutt and Rougier, where the number of basis functions K is equal to N, thus we compare the algorithms by using the same values of N. As input parameters we use the same as in **3**.

For Algorithm 2, in the delay-free case, the computation time increases slowly with N, in contrast with Algorithm 1 in which the computational time increases faster with (N,K), illustrated by Figure 3. In the delay-dependent case, the performance of Algorithm 2 proves much faster, see Table 2.

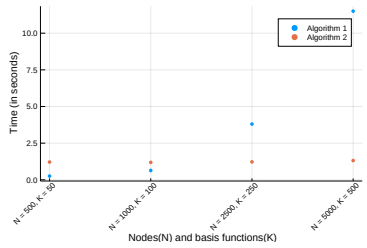

Figure 3: Comparing algorithms: Time performance of Algorithm 1 (Blue) and Algorithm 2 (Red) in the delay-free case

$(N,\ K)$	n	Performing time (in seconds) for Algorithm 1	Performing time (in seconds) for Algorithm 2
(5000,500)	2000	4725.89	953.91
(2500,250)	1000	662.44	125.88
(1000,100)	400	54.69	10.11
(500,50)	200	5.97	2.35

Table 2: Comparing algorithms: Time performance of Algorithm 1 and Algorithm 2 in the delay-dependent case

Although we can not account for the accuracy of this method, it seems that using the FFT improves considerably the time needed to find a solution for neural field equations.

5. Complexity and Convergence

To analyse the complexity and convergence of the proposed algorithm, we will apply the input functions presented in [15]. The firing rate function $S(x)$ is the Heaviside function; the connectivity kernel is given by $F(x) = 2\exp(-0.08x)(0.08\sin(\frac{\pi x}{10}) + \cos(\frac{\pi x}{10}))$ and the external input has the form $I(x,t) = -3.39967 + 8\exp(\frac{-x^2}{18})$. The functions will be evaluated in the time interval [0,4] and in the space interval $\Omega = [-50, 50]$, with α=1 and ξ=0.1.

5.1. Complexity

It is important to understand for which parameter we are going to test the complexity. We could consider the number of points of discretization in space (N) and the number of basis functions (K), yet it suffices to test the complexity for N. Since N has to be much greater than K in order to improve the accuracy of the Galerkin method, we choose to have (always) N=10K. By equation (13), at each step in time, K inner products need to be computed and for each of them, N sums need to be calculated. Therefore, an order of $\mathcal{O}(K \times N)$=$\mathcal{O}(\frac{N^2}{10})$ =$\mathcal{O}(N^2)$. Suppose that $t(N)$ is the time needed to finish one simulation, where the parameter of discretization is N. We assume that

$$t(N) = CN^w \quad \text{and} \quad t(2N) = C(2N)^w$$

and want to find w which will be the complexity of our algorithm, then:

$$\frac{t(2N)}{t(N)} = \frac{(2N)^w}{N^w} = 2^w \Leftrightarrow w = \log_2 \frac{t(2N)}{t(N)} \qquad (16)$$

Although we can expect an order of $\mathcal{O}(N^2)$, $t(N)$ does not depend solely on the number of operations and on equation (13).
We will measure the time that a simulation takes to run, fixing n=200 for the delay-free case, ns=1 and changing (N,K) values to be (1000,100), (2000,200) and (4000,400).

$(N,\ K)$	Performing time (in seconds)
(1000,100)	0.6844
(2000,200)	2.1067
(4000,400)	8.1207

$(N,\ K)$	n	Performing time (in seconds)
(1000,100)	400	54.69
(2000,200)	800	324.13
(4000,400)	1600	2484.23

Table 3: Complexity Analysis: Time performance of the algorithm in the case without delay (Left); with delay (Right)

Considering Table 3, we can now obtain the order of our method for the deterministic delay-free case, which was as expected $w \approx 2$ (Table 4).

N	**2N**	$\boldsymbol{w = \log_2 \frac{t(2N)}{t(N)}}$
1000	2000	1.6221
2000	4000	1.9467

Table 4: Complexity Analysis: case without delay

As discussed above, in the delay-dependent case, we use the formula $h_t = \frac{h_x}{v}$; therefore, as a rule, when we change the value of N and K we need to change the value of n as well. As we are trying to compute the complexity with respect to N, n should remain constant, which is incompatible with the rule explained above.

5.2. Convergence

As we do not have an exact solution to compare our results with, we estimate the convergence order p, using the following formula $p \approx \frac{\log \frac{|u_h - u_{\frac{h}{2}}|}{|u_{\frac{h}{2}} - u_{\frac{h}{4}}|}}{\log 2}$.
We assume that, in the deterministic, delay-free case the algorithm will have order of $\mathcal{O}(h_x^2)+\mathcal{O}(h_t)$, due to the trapezoidal rule used to approximate the integrals and the Euler-Maruyama method.
Convergence with respect to h_x
Fixing a point in time $t_n = 4$, we will analyse the different results obtained in particular space points. We consider n=10000, N=500, 1000 and 2000. The specific points studied are x_k=-20, 0, 40.
Notice that this order of convergence is not $p \approx 2$

x_k	N = 500	N = 1000	N = 2000
-20	-0.8749	-0.8837	-0.8794
0	16.1566	16.1445	16.1496
40	-2.8404	-2.8421	-2.8412

x_k	p
-20	1.03
0	1.25
40	0.91

Table 5: (Left) Numerical values of $U(x_k, 4)$; (Right) Convergence with respect to h_x; using a Heaviside type firing rate function

or higher (Table 5), as it was expected since we are using the trapezoidal rule. This is due to the fact that the integrand functions contains the Heaviside function, which is not continuous. Hence, we can not guarantee that in this case, the trapezoidal rule will converge at the optimal rate. To test how the smoothness of the integrand function can influence the convergence order, we have carried an additional numerical experiment with a different firing rate function $S(x) = \frac{1}{1+\exp(-10(x-1))}$, which is twice continuously differentiable on the interval $[-50, 50]$, that is $S \in C^2([-50, 50])$. Using this type of firing rate function, we obtain even a greater order than $\mathcal{O}(h_x^2)$, see Table 6.

x_k	N = 500	N = 1000	N = 2000
-20	-0.848632	-0.84903	-0.84899
0	16.07923	16.0770	16.07691
40	-2.83501	-2.835044	-2.835040

x_k	p
-20	3.31
0	4.63
40	3.1

Table 6: (Left) Numerical values of $U(x_k, 4)$; (Right) Convergence with respect to h_x; for the deterministic case without delay, using a continuous type firing rate function

Convergence with respect to h_t

Fixing values in space, we will now analyse the different results obtained in particular time instants. We will fix x_k=0 with N=5000 and gather the results when n=500, 1000 and 2000.

t_n	n = 500	n = 1000	n = 2000
2	14.2290	14.2241	14.2217
4	16.1474	16.1456	16.1447

t_n	p
2	1.02
4	1

Table 7: (Left) Numerical values of $U(0, t_n)$; (Right) Convergence with respect to h_t; for the deterministic case without delay

This was as expected $\mathcal{O}(h_t)$.

6. Simulations

In this section, we will consider similar functions to the ones described in Section 5, adapted from [1] and [2]. The firing rate $S(x)$, the connectivity kernel $F(x)$, α and ξ are the same, but the external input has now the form

$$I(x,t) = \begin{cases} -3.39967 + 8\exp(-\frac{x^2}{2\sigma^2}), & \text{if} \quad t \in [0,5] \\ -2.89967, & \text{if} \quad t \in (5,10] \end{cases}$$

which was inspired by the example considered by Ferreira, see [[7], p. 37]. With the given time interval, the number of bumps of the solution, as well as their form, in most cases stabilizes. However, it is not excluded that the position of the bumps may be shifted with time.

We assume that $I(x,t)$ has a gaussian form in the interval $t \in [0,5]$, hence σ is interpreted as the standard deviation. This problem is closely related with the simulation of working memory in the cortex, where $I(x,t)$ represents a transient external stimulus (e.g. a light signal) which is turned off after a certain amount of time. The stationary solution results from the neuronal activity and it contains information about the stimulus, which is kept in the cortex. For the cases with delay, we will always consider the finite transmission speed v=10. For the stochastic cases we perform ns=100 independent Monte Carlo numerical simulations and we will consider the external stimulus $I(x,t)$ with σ=3 or σ=13. Furthermore, we will use as input parameters: N=2000, K=200, n=1000, Ω=[-100,100] and homogeneous initial conditions, that is, $U_0(x) \equiv 0$ for the delay-free case and $U_0(x,t) \equiv 0$ if $t \in [\tau_{max}, 0]$ in the delay-dependent case.

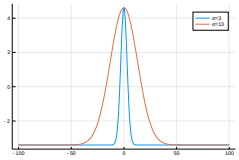

Figure 4: Dispersion of the external stimulus inputs, for σ=3 and 13

6.1. Deterministic Case

6.1.1 Delay-free

There are many papers that deal with the deterministic NFE and produce results on this case, e.g. [15], [1], [2]. Our attempt to recreate these results is illustrated by Figure 5.

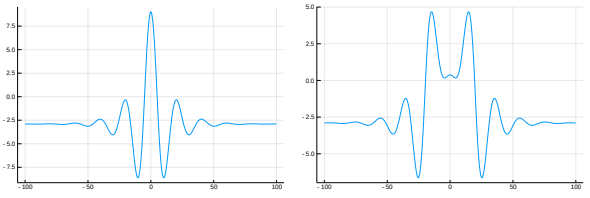

Figure 5: Numerical solution profiles of the deterministic, delay-free NFE with σ=3 (Left) and σ=13 (Right) at t=10

6.1.2 Delay-dependent

In Figure 6, for an average dispersion of the signal, σ=3, the solution profiles for the delay-free and delay-dependent cases overlap and by studying the

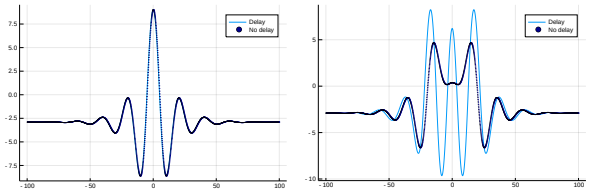

Figure 6: Numerical solution profiles of the deterministic, delay-free and delay-dependent NFE with σ=3 (Left); σ=13 (Right) at t=10

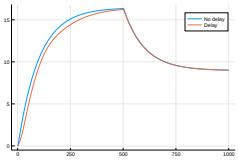

Figure 7: Evolution of the solution maximum for the deterministic, delay-free (Blue) and delay-dependent (Red) NFE with σ=3

graphics of the delay-free and delay-dependent solutions at t=10 do not overlap. In this case, we have a scattered input signal and consequently a scattered response to that signal, which leads to solutions of higher complexity that respond differently to delay.

6.2. Stochastic Case

In this section we will present the plots of the functions $\underline{\text{mean}}(x,t)$, $\underline{\max}(x,t)$ and $\underline{\min}(x,t)$ defined as follows:

$$\texttt{mean} := \text{mean}(x,t) = \frac{1}{100}\sum_{s=1}^{100} U_s(x,t)$$

$$\texttt{max} := \max(x,t) = \max_{s\in\{1,\cdots,100\}} U_s(x,t)$$

$$\texttt{min} := \min(x,t) = \min_{s\in\{1,\cdots,100\}} U_s(x,t)$$

Note that $\underline{\text{mean}}$ is the average of all the paths (which approximates the mathematical expectation of the random process), while $\underline{\max}$ and $\underline{\min}$ give the maximal and minimal values (respectively) among the given set of paths. We will study the influence of weak (ϵ=0.05) and strong (ϵ=0.5) noise.

6.2.1 Delay-free

$\underline{\text{Weak noise, } \epsilon=0.05}$

When considering the weak noise case, it is noticeable that the $\underline{\text{mean}}(x,t)$ of all trajectories (Figure 8) remains similar to the solution of the corresponding deterministic delay-free NFE and thus, it appears that stochastic neural fields inherit the behaviour of the underlying deterministic neural field model. In spite of this, the solutions pattern is again sensitive to noise when σ=13. If the dispersion signal

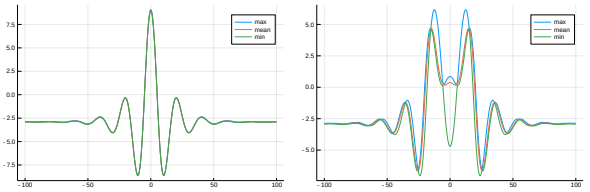

Figure 8: Graphics of `max`, `mean` and `min` of the delay-free stochastic NFE for σ=3 (Left); σ=13 (Right) and *weak noise* (ϵ=0.05), at t=10

is average, σ=3, then the influence of weak noise is completely negligible. However, when the external stimulus is applied to a sufficiently large domain of neurons (σ=13), even weak noise may produce some changes in the solutions behaviour, which is visible in the right plot in Figure 8, where after analysing the different paths, we concluded that there were 94 one-bump paths (similar to the deterministic solution) and 6 two-bump paths.

$\underline{\text{Strong noise, } \epsilon=0.5}$

In the case of *strong noise*, the results are very

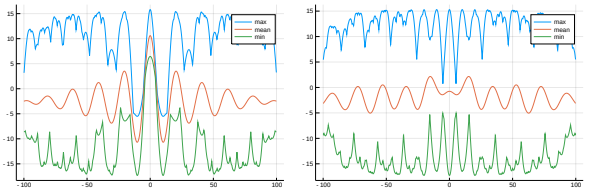

Figure 9: Graphics of `max`, `mean` and `min` of the stochastic, delay-free NFE for σ=3 (Left); σ=13 (Right) and *strong noise* (ϵ=0.5), at t=10

different from the ones computed for the deterministic, delay-free NFE. In this case, regardless of the value of σ, the formation of multi-bump solutions occurs. The precise number of all SNFE paths with a particular number of bumps is showed in Table 8 for t=10.

$\boldsymbol{\sigma}$	**Number of bumps observed**										
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
3	61	0	11	0	1	0	9	0	14	0	4
13	7	19	15	3	20	2	3	15	4	10	2

Table 8: Number of paths with a certain number of bumps, for the strong noise delay-free case, at t=10 and σ=3 and 13, out of 100 paths

6.2.2 Delay-dependent

$\underline{\text{Weak noise, } \epsilon=0.05}$

In the weak noise case, by analysing the results for σ=3 (left plot in Figure 10), we observe that the solution profile is equal to the one presented in the deterministic NFE (left plot in Figure 5), hence the effect of delay and additive noise is negligible. When considering σ=13 (right plot in Figure

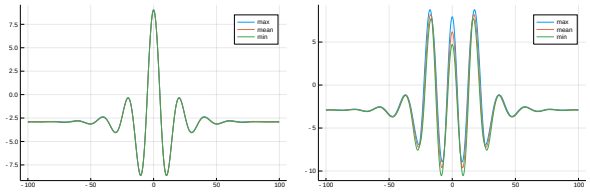

Figure 10: Graphics of `max`, `mean` and `min` of the stochastic, delay-dependent NFE for σ=3 (Left); σ=13 (Right) and *weak noise* (ϵ=0.05), at t=10

10), the solution does not resemble the deterministic delay-free NFE, instead, it resembles the solution obtained when we have delay but not noise (right plot Figure 6).

Strong noise, ϵ=0.5

For strong noise, we can see by Figure 11 that the

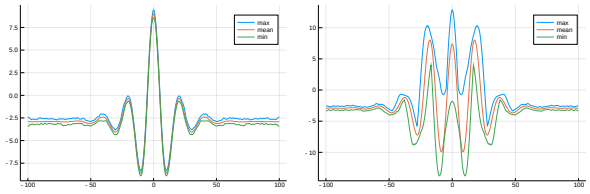

Figure 11: Graphics of `max`, `mean` and `min` of the delay-dependent stochastic NFE for σ=3 (Left); σ=13 (Right) and *strong noise* (ϵ=0.5)

paths are more varied then the ones in Figure 10, regardless of the dispersion of the signal. Additionally, we can clearly differentiate the results when considering noise without delay (Table 8) and with delay (Table 9), in particular in the formation of multi-bump solutions. Although for σ=13, we do not get a pattern similar to the delay-free NFE, we presume this is due to the delay and not the noise, since we already had the formation of 3-bump solutions when only accounting for delay.

$\boldsymbol{\sigma}$	Number of bumps observed		
	1	**2**	**3**
3	100	0	0
13	0	1	99

Table 9: Number of paths with a certain number of bumps for the strong noise, delay-dependent case, at t=10 and σ=3 and 13, out of 100 paths

7. Conclusions

After a brief comparison between the programming languages MATLAB and Julia, the latter was chosen to implement the proposed method based on the promise of faster computations (using just-in-time compiling) while still using a high syntax language and a graphical output very similar to MATLAB. We were able to show that the complexity of the algorithm with respect to the number of points in the space discretization N was, as theorized for the deterministic case without delay, $\mathcal{O}(N^2)$ and that the method converges with the total order of $\mathcal{O}(h_t) + \mathcal{O}(h_x)$. This was contrary to what was expected and it is explained by the use of the Heaviside function as the firing rate, which is not continuous and therefore the trapezoidal rule may not converge at the optimal rate. Still regarding the algorithm, we were able to recognise the advantages of using the FFT when solving NFEs, since it reduces considerably the time needed to find a solution. However, in this work, we have used an approach without the FFT, because this allows us to use a number of nodes N much higher than the number of basis functions K, which reduces drastically the error of the numerical integration. We emphasize that in spite of the complexity of the algorithm, in the one-dimensional case it is possible (without using the FFT) to obtain accurate numerical approximations within a reasonable computing time.

In the cases studied in this text, we determined that when delay is introduced in the equation, the solution takes a longer time to respond to the external input and as a consequence, the stationary state is reached slower than in the deterministic case. The results with delay are also influenced by the spatial effect of the dispersion of the external stimulus, controlled by σ. When considering a medium dispersion, the solution is the same at the final instant for the delay-free and delay-dependent cases, the only difference is the time it takes to reach the stationary state. On the other hand, when we have a larger dispersion of the signal (σ=13), this leads to solutions with higher complexity and different from the ones of the delay-free case. We have also analysed the influence of additive noise (weak and strong) in the equation. For the weak noise (ϵ=0.05), the influence is negligible since in both cases the mean of all paths produces a solution profile identical to its underlying deterministic counter part. It is important to notice that although the mean of all paths is identical to the deterministic solution without delay, the solutions pattern is again sensitive to a wider dispersion signal (σ=13) and therefore we can obtain in some paths of the stochastic solution, different patterns from the ones of the deterministic, delay-free case. This contrasts with the results obtained with a strong additive noise (ϵ=0.5), where the solutions are very different from the deterministic case without delay, producing a diverse range of paths with different number of bumps, independently of the value of σ. In conclusion, choosing a sufficiently strong additive noise may change significantly the profile of the SNFE model.

Finally we analyse how delay and noise together interfere with the neural field model. When considering weak noise, we notice again that the effect is

negligible with both of the dispersion values. Although with σ=13 we do not obtain solutions with an equal profile to the deterministic case, this can be attributed solely to the delay. When we have a strong influence of noise, on the contrary to the delay-free case, we do not have the formation of multi-bump solutions. Instead, we see results very similar to the ones presented in the delay-dependent case without noise. In conclusion, the results seem to imply that delay soothes the random effects resulting from the additive noise.
All computations were made on a PC with processor Intel(R) Core(TM) i5-5200U CPU 2.20GHz×4 and with 4 GB of installed memory (RAM).

References

[1] Kulikov, G.Y. and Kulikova, M.V. and Lima,P.M. Numerical simulation of neural fields with finite transmission speed and random disturbance. *Proceedings of 23. International Conference on System Theory, Control and Computing (ICSTCC)*, pages 644–649, 2019.

[2] Kulikov, G.Y. and Kulikova, M.V. and Lima,P.M. Numerical solution of the neural field equation in the presence of random disturbance. *Journal of Computational and Applied Mathematics*, 112563, 2019.

[3] Amari, S. Dynamics of pattern formation in lateral-inhibition type neural fields. *Biol. Cybern.*, 27(77), february (1977). https://doi.org/10.1007/BF00337259.

[4] Beurle, R.L. Properties of a mass of cells capable of regenerating pulses. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 240, 08 1956.

[5] Coombes, S. Neural fields. http://www.scholarpedia.org/article/Neural_fields, 2006. [Online; accessed 11-September-2019].

[6] Destexhe, A. and Rudolph-Lilith, M. *Neuronal Noise*. Springer Series in Computational Neuroscience. Springer US, 2012.

[7] Ferreira, Flora. *Multi-bump solutions in dynamic neural fields: analysis and applications*. PhD thesis, Universidade do Minho, Portugal, 05 2014.

[8] FitzHugh, Richard. Impulses and physiological states in theoretical models of nerve membrane. *Biophysical Journal*, 1(6):445 – 466, 1961.

[9] Fitzhugh, Richard. Computation of impulse initiation and saltatory conduction in a myelinated nerve fiber. *Biophysical Journal*, 2(1):11 – 21, 1962.

[10] Ginzburg, Iris and Sompolinsky, Haim. Theory of correlations in stochastic neural networks. *Phys. Rev. E*, 50:3171–3191, Oct 1994.

[11] Hodgkin, A.L. and Huxley, A.F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Bulletin of Mathematical Biology*, 52(1):25 – 71, 1990.

[12] Hutt, Axel and Rougier, Nicolas P. Activity spread and breathers induced by finite transmission speeds in two-dimensional neural fields. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 82:055701, 11 2010.

[13] Hutt, Axel and Rougier, Nicolas P. Numerical simulation scheme of one- and two- dimensional neural fields involving space-dependent delays. *Neural Fields: Theory and Applications*, 01 2013.

[14] Kühn, Christian and Riedler, Martin G. Large deviations for nonlocal stochastic neural fields. *Journal of mathematical neuroscience*, 4:1, 04 2014.

[15] Lima, P.M. *Numerical Investigation of Stochastic Neural Field Equations, in Advances in Mathematical Methods and High Performance Computing: V.K. Sing , D. Gao, A. Fischer*. Springer International Publishing, 2019.

[16] Nagumo, J. and Arimoto, S. and Yoshizawa, S. An active pulse transmission line simulating nerve axon. *Proceedings of the IRE*, 50(10):2061–2070, Oct 1962.

[17] Sequeira, Tiago. Numerical investigation of two-dimensional stochastic neural field equations with delay. Master's thesis, Instituto Superior Técnico, 2019. (*in preparation*).

[18] Shardlow, Tony. Numerical simulation of stochastic pdes for excitable media. *J. Comput. Appl. Math.*, 175(2):429–446, Mar. 2005.

[19] Soula, Hédi and Chow, Carson C. Stochastic dynamics of a finite-size spiking neural network. *Neural Comput.*, 19(12):3262–3292, Dec. 2007.

[20] Wilson, Hugh and Cowan, Jack. Excitatory and inhibitory interactions in localized populations of model neurons. *Biophysical journal*, 12:1–24, 02 (1972).