Numerical investigation of the two-dimensional neural field equation with delay

Pedro M. Lima
Intitute of Stochastics
Johannes Kepler University
Altenbergerstr. 69, 4040 Linz, Austria
Email: plima@math.ist.utl.pt

Evelyn Buckwar
Intitute of Stochastics
Johannes Kepler University
Altenbergerstr. 69, 4040 Linz, Austria
Email: evelyn.buckwar@jku.at

***Abstract*—Neural Field Equations (NFEs) are integro-differential equations which describe the electric potential field and the interaction between neurons, in certain regions of the brain. They are becoming increasingly important for the interpretation of EEG, fMRi and optical imaging data. In the present article we describe a new efficient algorithm for the numerical simulation of two-dimensional neural fields with delays. The main features of this method are discussed and its performance is illustrated by some numerical examples.**

I. Introduction

The main idea of the neural field models in Mathematical Neuroscience is to treat the cortical space as continuous. Since the number of neurons and synapses is extremely high even in a small piece of cortex, this idea appears naturally as a first approximation to model the neural activity. As remarked by Coombes [3], "neural fields provide a framework for unifying data from dfferent imaging modalities", which maybe explains their increasing role in research. This approach was first developped in 70's by Wilson and Cohen [8], Amari [1], and Nunez [7]; it leads to integro-differential equations (or systems of them), which may be written in the form:

$$c\frac{\partial}{\partial t}V(\bar{x},t) = I(\bar{x},t) - V(\bar{x},t) + \int_\Omega K(|\bar{x}-\bar{y}|)S(V(\bar{y},t))d\bar{y}, \qquad t\in[0,T], \bar{x}\in\Omega\subset\mathbb{R}^2, \tag{1}$$

where the unknown $V(\bar{x},t)$ is a continuous function $V : \Omega\times[0,T]\to\mathbb{R}$, I, K and S are given functions; c is a constant. In this article, by $|\bar{x}-\bar{y}|$ we mean $\|\bar{x}-\bar{y}\|_2$. The physical meaning of $V(\bar{x},t)$ is the membrane potential in point x at time t. The function I represents external sources of excitation and S describes the dependence between the firing rate of the neurons and their membrane potential. It can be either a smooth function (typically of sigmoidal type) or a Heaviside function. The kernel function $K(|\bar{x}-\bar{y}|)$ gives the connectivity between neurons in positions $\bar{x}$ and $\bar{y}$. By writing the arguments of the function in this form we mean that we consider the connectivity homogeneous, that is, it depends only on the distance between neurons, and not on their specific location. We search for a solution V of (1) which satisfies the initial condition

$$V(\bar{x},0) = V_0(\bar{x}), \qquad \bar{x}\in\Omega. \tag{2}$$

According to many authors (see for example [4]), realistic models of neural fields must take into account that the propagation speed of neuronal interactions is finite, which leads to NFE with delays of the form

$$c\frac{\partial}{\partial t}V(\bar{x},t) = I(\bar{x},t) - V(\bar{x},t) + \int_\Omega K(|\bar{x}-\bar{y}|)S(V(y,t-\tau(\bar{x},\bar{y}))d\bar{y}, \qquad t\in[0,T], \quad \bar{x}\in\Omega\subset\mathbb{R}^2, \tag{3}$$

where $\tau(\bar{x},\bar{y}) > 0$ is a delay, depending on the spatial variables. In the last case, the initial condition has the form

$$V(\bar{x},t) = V_0(\bar{x},t), \qquad \bar{x}\in\Omega, \quad t\in[-\tau_{max},0], \tag{4}$$

where $\tau_{max} = \max_{\bar{x},\bar{y}\in\Omega}\tau(\bar{x},\bar{y})$.

In the two-dimensional case, the required computational effort to solve equations (1) and (3) grows very fast as the discretization step is reduced, and therefore special attention has to be paid to the creation of effective methods. This can be achieved by means of low-rank methods, as those discussed in [9], when the kernel is approximated by polynomial interpolation, which enables a significant reduction of the dimensions of the matrices. In [2], the authors use an iterative method to solve linear systems of equations which takes into account the special form of the matrix to introduce parallel computation.

Concerning equation (3), besides the existence and stability of solution, numerical approximations were obtained in [4]. The computational method applies quadrature rule in space to reduce the problem to a system of delay differential equations, which is then solved by a standard algorithm for this kind of equations. A more efficient approach was recently proposed in [5], where the authors introduce a new approach to deal with the convolution kernel of the equation and use Fast Fourier Transforms to reduce significantly the computational effort required by numerical integration.

II. Numerical method for the two-dimensional equation without delay

A. Time Discretization

We begin by rewriting equation (1) in the form

$$c\frac{\partial}{\partial t}V(\bar{x},t) = I(\bar{x},t) - V(\bar{x},t) + \kappa(V(\bar{x},t)) \qquad t\in[0,T], \bar{x}\in\Omega\subset\mathbb{R}^2, \tag{5}$$

where κ denotes the nonlinear integral operator defined by

$$\kappa(V(\bar{x},t)) = \int_\Omega K(|\bar{x}-\bar{y}|)S(V(\bar{y},t))d\bar{y}. \tag{6}$$

We shall first deal with the time discretization in equation (5), therefore we introduce the stepsize $h_t > 0$ and define

$$t_i = ih_t, \quad i = 0, ..., M, \quad T = h_t M.$$

Moreover, let

$$V_i(\bar{x}) = V(t_i, \bar{x}), \quad \forall x \in \Omega, \quad i = 0, ..., M.$$

We shall approximate the partial derivative in time by the backward difference

$$\frac{\partial}{\partial t} V(\bar{x}, t_i) \approx \frac{3V_i(\bar{x}) - 4V_{i-1}(\bar{x}) + V_{i-2}(\bar{x})}{2h_t}, \qquad (7)$$

which gives a discretization error of the order $O(h_t^2)$, for sufficiently smooth V. By substituting (7) into (5) we obtain the implicit scheme

$$c\frac{3U_i - 4U_{i-1} + U_{i-2}}{2h_t} = I_i - U_i + \kappa(U_i), \quad i = 2, ..., M, \quad (8)$$

where U_i approximates the solution of (5).

To start this scheme we need to know U_0, which is defined by the initial condition V_0, and U_1, which can be obtained by a one-step method, for example, the explicit Euler method. It can be easily shown that his scheme is zero-stable, since its characteristic roots are not greater than one ant the root one is not multiple.

Our next step is to investigate under which conditions equation (8) has a unique solution, so that each step of the iterative process is well defined. With this purpose we write this equation in the form

$$U_i(\bar{x}) - \frac{1}{1 + \frac{3c}{2h_t}} \kappa(U_i) = f_i(\bar{x}), \qquad \bar{x} \in \Omega \qquad (9)$$

where

$$f_i(\bar{x}) = \left(1 + \frac{2h_t}{3c}\right)^{-1} \left(I_i + \frac{c}{h_t} 2U_{i-1}(\bar{x}) - \frac{c}{2h_t} U_{i-2}(\bar{x})\right), \qquad (10)$$

$\bar{x} \in \Omega$. In order to prove the solvability of (9), (10), we define the iterative process:

$$U_i^{(\nu)}(\bar{x}) = \lambda\kappa\left(U_i^{(\nu-1)}(\bar{x})\right) + f_i(\bar{x}) = G\left(U_i^{(\nu-1)}(\bar{x})\right), \quad (11)$$

$\bar{x} \in \Omega, \nu = 1, 2, ...,$ where

$$\lambda = \frac{1}{1 + \frac{3c}{2h_t}} = \frac{2h_t}{2h_t + 3c}. \qquad (12)$$

It can be shown that for a sufficiently small step size h_t the function G is contractive in a certain closed set $X \subset F$, such that $G(X) \subset X$, therefore, by the Banach fixed point theorem equation (9) has a unique solution in X and the sequence $U_i^{(n)}$, defined by (11), converges to this solution, for any initial guess $U_i^{(0)} \in X$. In our case, the solution is by construction the iterate U_i, so it should be close to U_{i-1} and U_{i-2}. Therefore it makes sense to assume that X is a certain set containing U_{i-1} and U_{i-2} and to choose $U_i^{(0)} = U_{i-1}$.

The above construction not only shows that the equation (9) has a unique solution in a certain set X, but it also suggests that the iterative process (11), starting with $U_i^{(0)} = U_{i-1}$, can be effectively used to approximate this solution. Actually, the convergence of the process will be faster and faster as h_t tends to zero.

B. Space Discretization

Since the equation (9) in general cannot be solved analytically, we need a computational method to compute a numerical approximation of its solution. By other words, we need a space discretization, which will be the subject of this subsection.

For the sake of simplicity, assume that Ω is a rectangle: $\Omega = [-1, 1] \times [-1, 1]$. We now introduce a uniform grid of points (x_i, x_j), such that $x_i = -1 + ih$, $i = 0, ..., n$, where h is the discretization step in space. In each subinterval $[x_i, x_{i+1}]$ we introduce k Gaussian nodes: $x_{i,s} = x_i + \frac{h}{2}(1 + \xi_s)$, $i = 0, 1, \ldots n-1$, where ξ_s are the roots of the k-th degree Legendre polynomial, $s = 1, ..., k$. We shall denote Ω_h the set of all grid points (x_{is}, x_{jt}), $i, j = 0, ..., n-1, s, t = 1, ..., k$. A Gaussian quadrature formula to evaluate the integral $\int_\Omega f(u, v)dudv$ will have the form

$$Q(f) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \sum_{s=1}^{k} \sum_{t=1}^{k} \tilde{w}_s \tilde{w}_t f(x_{is}, x_{jt}), \qquad (13)$$

with $\tilde{w}_s = \frac{h}{2} w_s$, where w_s are the standard weights of a Gaussian quadrature formula with k nodes on $[-1, 1]$, $s = 1, ..., k$. As it is well-known, a quadrature formula of this type has degree $2k - 1$ and therefore, assuming that f has at least $2k$ continuous derivatives on Ω, the integration error of (13) is of the order of h^{2k}. Note that the total number of nodes in the space discretization is $k^2 n^2$.

When we introduce the quadrature formula (13) to compute $\kappa(U)$ we define a finite-dimensional approximation of the operator κ. Let us denote U^h a vector with N^2 entries, where $N = n\,k$, such that

$$(U^h)_{is,jt} \approx U(x_{is}, x_{jt});$$

then the finite-dimensional approximation of $\kappa(U)$ may be given by

$$(\kappa^h(U^h))_{mu,lv} = \sum_{i=0}^{n_1} \sum_{j=0}^{n_2} \sum_{s=1}^{k} \sum_{t=1}^{k} \tilde{w}_s \tilde{w}_t \times K(\|(x_{mu}, x_{lv}) - (y_{is}, y_{jt})\|_2) S((U^h)_{is,jt}). \qquad (14)$$

By replacing κ with κ_h in equation (9) we obtain the following system of nonlinear equations:

$$U^h - \frac{1}{1 + \frac{2h_t}{3c}} \kappa^h(U^h) = f^h, \qquad (15)$$

where $\kappa^h(U^h)$ is defined by (14) and

$$(f^h)_{is,jt} = f(x_{is}, x_{jt}),$$

with f defined by (10); in (15), for the sake of simplicity, we have omitted the index i of U_i^h. Note that for the the computation of f^h we have to evaluate the iterates U_{i-1} and U_{i-2} at all the points of Ω_h. We denote the vectors resulting from this evaluation by U_{i-1}^h and U_{i-2}^h, respectively. We conclude that at each time step of our numerical scheme we must solve (15), which is a system of N^2 nonlinear equations.

We can investigate the solvability of (15) in the same way as we have studied the Fredholm integral equation (9). More precisely, we can introduce the iterative procedure

$$U^{h,(\nu)} = \lambda\kappa^h(U^{h,(\nu-1)}) + f^h = G^h(U^{h,(\nu-1)}), \qquad (16)$$

$\nu = 1, 2, \ldots$. As a starting point for this iterative process, we take

$$U^{h,(0)} = U^h_{i-1}$$

(similar to the case of the iteration (11) for the Fredholm equation). In this case, the convergence of the iterative procedure (16) depends on the contractivity of the nonlinear function G^h. Again it can be shown that under a certain restriction on h_t the iterative procedure (16) converges to the solution of (15).

We have also investigated the convergence of U^h_i to U_i, as $h \to 0$. Knowing the properties of Gaussian quadratures, and assuming that the functions K and S in (6) are sufficiently smooth, one can show that

$$\|\kappa(U_i) - \kappa^h(U_i))\|_\infty \leq Mh^{2k}, \tag{17}$$

where M is some constant independent from h. Based on (17) and using again the fact that K is bounded and S is continuously differentiable, we obtain that

$$\|U_i - U^h\|_\infty = O(h^{2k}), \qquad \text{as } h \to 0. \tag{18}$$

The proof of this error estimate can be found in [6].

C. Computational Implementation

The above numerical algorithm for the approximate solution of the neural field equation in the two-dimensional case was implemented by means of a MATLAB code.

The code has the following structure. After introducing the input data (step size in time and in space, initial condition U_0, error tolerance for the inner cycle, required number of steps in time) , there is an outer cycle that computes each vector U^h , given U^h_{i-1} and U^h_{i-2}, according to the multistep method (8). In order to initialize this cycle, besides U_0, we need U^h_1, which is obtained by the explicit Euler method. More precisely, we compute

$$U^h_1 = U_0 + \frac{h_t}{c}(I_0 - U_0 + \kappa^h(U_0)). \tag{19}$$

We recall that at each step in time we must solve the nonlinear system of equations (9), which as suggested above is obtained by means of the fixed point method, that is, we iterate the scheme (16), until the iterates satisfy

$$\|U^{h,(\nu)} - U^{h,(\nu-1)}\|_\infty < \epsilon,$$

for some given ϵ. This is the inner cycle of our scheme. Typically, in all the examples we have computed the number of iterations in the inner cycle is not very high (3-4, in general), confirming that the fixed point method is an efficient way of solving the system (15). To start the inner cycle we use an initial guess which is obtained from U^h_{i-1} using again the Euler method:

$$U^{h,(0)} = U_{i-1} + \frac{h_t}{c}(I_i - U^h_{i-1} + \kappa^h(U^h_{i-1})). \tag{20}$$

Note that at each step of the inner cycle it is necessary to compute the function κ_h at all the grid points. From the computational point of view, this means that we must evaluate N^2 times a quadrature rule of the form (13) (with N^2 nodes). Of course, this requires a high computational effort and the greatest part of the computing time of our algorithm is spent in this process. Therefore, we pay special attention to reducing the computational cost at this stage. In order to improve the efficiency of the numerical method, we apply the following technique, proposed in [9] for the solution of two-dimensional Fredholm equations. Assuming that the function V is sufficiently smooth, we can approximate it by an interpolating polynomial of a certain degree. As it is known from the theory of approximation, the best approximation of a smooth function by an interpolating polynomial of degree m is obtained if the interpolating points are the roots of the Chebyshev polynomial of degree m:

$$p^m_i = cos\left(\frac{(2i-1)\pi}{m}\right), \qquad i = 1, ..., m \tag{21}$$

Our approach for reducing the matrices rank in our method consists in replacing the solution V_i by its interpolating polynomial at the Chebyshev nodes in Ω. If V_i is sufficiently smooth, this produces a very small error and yields a very significant reduction of computational cost. Actually, when computing the vector $\tilde{\kappa}(U_i)$ (see formula (6)) we have only to compute m^2 components, one for each Chebyshev node on $[-1, 1]\times[-1, 1]$. Choosing m much smaller than n, we thus obtain a significant computational advantage.

The procedure at each iteration is as follows. We compute the matrix M such that

$$M_{i,j} = Q(V(p^m_i, p^m_j, t)), \qquad i = 1, ..., m, j = 1, ..., m,$$

where Q is the approximation of the integral κ, obtained by means of the quadrature (13), p^m_i are the Chebyshev nodes, defined by (21). Then we have to perform the matrix multiplication

$$\Lambda = CMC^T, \tag{22}$$

where C is the matrix defined by

$$C_{ij} = c_{i-1}(p^m_j), \qquad i = 1, ..., m, \quad j = 1, ..., m;$$

here c_k represents the scaled Chebyshev polynomial of degree k,

$$c_k(x) = \delta_k \cos(k \arccos(x)), \quad k = 0, 1, ..$$

with $\delta_0 = 1/\sqrt{n}$, $\delta_k = \sqrt{2}\delta_0$, $k = 1, ..., m-1$. The matrix Λ contains the coefficients of the interpolating polynomial of the solution (expanded in terms of scaled Chebyshev polynomials). Finally, in order to obtain the interpolated values of the solution at the Gaussian nodes, we have to compute

$$T = P^T \Lambda P, \tag{23}$$

where P is the transformation matrix, given by

$$P_{ij} = c_{i-1}(x_{(j)}), \qquad i = 1, ..., m, j = 1, ..., N.$$

Here $x_{(j)}$ represents each Gaussian node: $x_{(j)} = x_{i,s}$, if $j = ik + s$. Finally, the vector U_i for the next time step (of size N^2) is obtained by copying T, row by row (note that T is a matrix of dimension $N \times N$).

D. Complexity Analysis

As remarked before, it is important to analyse the complexity of the computations, since the computational effort can be signifficantly reduced by the application of adequate techniques. In the previous section, we have described an algorithm for computing each iterate of the fixed point method,

which requires m^2 applications of the quadrature formula (13). Since this quadrature implies N^2 evaluations of the integrand function, we have a total of m^2N^2 function evaluations. Note that if no polynomial interpolation would be applied, N^4 evaluations of the integrand function would be required at each iteration. It is easy to conclude that the number of arithmetic operations required to apply the quadrature is also proportional to m^2N^2.

Then, according to the described algorithm, we must perform the matrix multiplication (22). Since the involved matrices have dimension $m \times m$, the total number of arithmetic operations is $O(m^3)$. Since, by construction, $m << N$, the complexity of this part of the computations is much less than the previous one.

Finally, we have the matrix product (23). Here the transformation matrix P has dimensions $m \times N$, as the resulting matrix T has dimensions $N \times N$ The resulting complexity is therefore $O(mN^2)$.

In conclusion, the number of evaluations of the integrand function in each iterate of the fixed point method is N^2m^2 and the complexity of each iteration is $O(N^2m^2)$. Note that the number of iterations of the fixed point method at each time step is typically $2 - 4$.

E. Error Analysis

We start by analysing the error resulting from the time discretization. Assuming that the partial derivatives $\frac{\partial^i V(x,t)}{\partial^i t}$, $i = 1, 2, 3$, are continuous on a certain domain $\Omega \times [0, T]$, the local discretization error of the approximation, given by (7), has the order of $O(h_t^2)$.

Concerning the space discretization, the error has two components: one resulting from the application of the discretization scheme (15) , and the other resulting from the polynomial interpolation. In both cases, the order of the approximation depends on the smoothness of the solution. Therefore we must choose the degree of the Gaussian quadrature according to the smoothness of U_i.

The first component was analysed in Sec. 2.1.2. If the functions S , K and U_i satisfy certain smoothness conditions, the discretization scheme (15) has order $2k$ in space, where k is the number of Gaussian nodes at each subinterval.

To analyse the interpolation error, we refer to Lemma 3 in [9]. According to this Lemma, if the partial derivatives $\frac{\partial^i f(y_1,y_2)}{\partial^i y_j}$ of a certain function f are continuous, with $j = 1, 2$, $i = 1, 2, .., s$ then

$$\|f - C_m f\| = O(m^{-s} \log^2 m), \tag{24}$$

where $C_m f$ represents the m-th degree interpolating polynomial of f in the Chebyshev nodes.

In order to obtain an optimal precision of the method, we require that the components of the error, given by (24) and (18), are of the same order.

If s is large (which in our case means that the functions K and S must have a high degree of smoothness), then the error given by (24) can be as small as the error of the space discretization, even if m is much smaller than N. This means that **we can drastically reduce the dimension of matrices without loss of accuracy**.

In conclusion, if the input functions K and S are sufficently smooth, the proposed method has second order convergence on h_t and order $2k$ on h. A more detailed error analysis can be found in [6].

III. Delay Equation

We now focus our attention on equation (3), where the argument of the solution inside the integral has a delay $\tau(\bar{x}, \bar{y})$. This delay takes into account the fact that the propagation speed of signals between neurons is finite and therefore the post-synaptic potential generated at location $\bar{x}$ in instant t by action potentials arriving from connected neurons at location $\bar{y}$ actually depends on the potential of these neurons at instant $t - \tau(\bar{x}, \bar{y})$, where $\tau(\bar{x}, \bar{y})$ is the time taken by the signal to come from $\bar{y}$ to $\bar{x}$. Since we assume that the propagation speed v is constant and uniform in space, we have

$$\tau(\bar{x}, \bar{y}) = \frac{|\bar{y} - \bar{x}|}{v}.$$

Hence, the delay integro-differential equation that we must solve has the form

$$c\frac{\partial}{\partial t}V(\bar{x}, t) = I(\bar{x}, t) - V(\bar{x}, t) + \int_\Omega K(|\bar{x} - \bar{y}|)S(V(\bar{y}, t - \frac{|\bar{y} - \bar{x}|}{v}))d\bar{y}. \tag{25}$$

Note that in this case the initial conditions satisfied by the solution of our problem have the form (4), where

$$\tau_{max} = \max_{\bar{x},\bar{y}\in\Omega} \frac{|\bar{y} - \bar{x}|}{v}.$$

The numerical algorithm used to solve equation (25) is essentially the same as described in the previous sections. The main difference results from the fact that when computing the integral on the right-hand side of (25) at instant t_i we must use not only the approximate solution at instants t_{i-1} and t_{i-2}, but at all instants t_{i-k}, $k = 1, ..., k_{max}$, where k_{max} is the integer part of τ_{max}/h_t. Note also that the argument $t_i - \frac{|\bar{y}-\bar{x}|}{v}$ may not be a multiple of h_t. In general let j and δ_t be the integer and the fractional part of $\frac{|\bar{y}-\bar{x}|}{vh_t}$. In this case, we have

$$t_{i-j-1} \leq t_i - \frac{|\bar{y} - \bar{x}|}{v} \leq t_{i-j}$$

and

$$h_t\delta_t = \left(t_i - \frac{|\bar{y} - \bar{x}|}{v}\right) - t_{i-j-1}.$$

The needed value of the solution $V(\bar{y}, t_i - \frac{|\bar{y}-\bar{x}|}{v})$ is then approximated by linear interpolation:

$$V\left(\bar{y}, t_i - \frac{|\bar{y} - \bar{x}|}{v}\right) \approx \delta_t U_{i-j} + (1 - \delta_t)U_{i-j-1}. \tag{26}$$

Note that the error analysis that we have carried out in the previous subsection may not apply to the delay equation. What we can say in this case, assuming that V is a smooth function of t, is that the error introduced each time we use the approximation formula (26) has the order of $O(h_t^2)$ (the same as the error resulting from the time discretization). However, the overall effect of this error in the computations requires a more detailed analysis, which is left for a future work.

Concerning complexity, in the case of the delay equation, each time we compute the integrand function, we must compute the delay $\tau(\bar{x}, \bar{y})$. As discussed above, this delay is obtained dividing the distance $\|\bar{y} - \bar{x}\|_2$ by v. Since this distance is also required to compute the kernel connectivity $K(\|\bar{y}-\bar{x}\|_2)$, for an effective computation this quantity should be evaluated only once and then kept in memory.

IV. Numerical Results

Here we present the results of some numerical tests we have carried out, in order to check the convergence properties of the described method (in the case where no delay is considered). In this subsection our main purpose is to test experimentally the convergence of the method and measure the error; therefore we have chosen some cases where the exact solution is known and do not arise from applications. However the form of the connectivity kernels and firing rate functions in these examples are close to the ones of neuroscience problems. We first check the convergence order in time. With this purpose, we consider the following example.

Example 1. In this example,

$$K(|\bar{x} - \bar{y}|) = \exp\left(-\lambda(x_1 - y_1)^2 - \lambda(x_2 - y_2)^2\right),$$

where $\lambda \in \mathbb{R}^+$; $S(x) = \tanh(\sigma x)$, $\sigma \in \mathbb{R}^+$. We set

$$I(x, y, t) = -\tanh\left(\sigma \exp\left(-\frac{t}{c}\right)\right) b(\lambda, x, y),$$

where

$$b(\lambda, x_1, x_2) = \int_{-1}^{1}\int_{-1}^{1} K(x_1, x_2, y_1, y_2)dy_1dy_2 = \\ = \frac{\pi}{4\lambda}\left(\text{Erf}(\sqrt{\lambda}(1 - x_1)) + \text{Erf}(\sqrt{\lambda}(1 + x_1))\right) \times \\ \left(\text{Erf}(\sqrt{\lambda}(1 - x_2)) + \text{Erf}(\sqrt{\lambda}(1 + x_2))\right),$$

where Erf represents the Gaussian error function.

In this case, it is easy to check that the exact solution is

$$V(\bar{x}, t) = \exp(-\frac{t}{c}).$$

The initial condition is $V_0(\bar{x}) \equiv 1$.

For the space discretisation, we have used $k = 4$, that is, 4 Gaussian nodes in each subinterval. Since the discretisation error in space must be $O(h^8)$, we consider it negligible, compared with the discretisation error in time.

With the following tests, we want to check that the discretisation error in time satisfies the condition

$$e_i = \|V_i - U_i\| = O(h_t^2).$$

The results are displayed in Table I. We have used two different time steps, $h_t = 0.02$ and $h_t = 0.01$, and we have approximated the solution over the time interval $[0, 0.1]$. For the space discretisation, we have considered $N = 24$, $m = 12$. The equation parameters are $\lambda = \sigma = c = 1$.

The discretisation errors $e_i(h_t) = \|V_i - U_i\|$ are displayed at different moments t_i, for different stepsizes h_t. We also present the ratios $e_i(2h_t)/e_i(h_t)$, which allow us check the convergence order. The ratios are close to 4, which confirms the second order convergence.

t	$e_i(0.01)$	$e_i(0.02)$	$e_i(0.02)/e_i(0.01)$
0.02	$6.66E-5$		
0.03	$7.24E-5$		
0.04	$7.46E-5$	$2.66E-4$	3.57
0.05	$7.56E-5$		
0.06	$7.61E-5$	$2.91E-4$	3.82
0.07	$7.65E-5$		
0.08	$7.69E-5$	$3.01E-4$	3.91
0.09	$7.72E-5$		
0.10	$7.76E-5$	$3.06E-4$	3.94

TABLE I. Numerical results for Example 1

m	$N = 12$	$N = 24$	e_{12}/e_{24}	$N = 48$	e_{24}/e_{48}
12	$3.11E-10$	$1.11E-12$	280	$3.997E-15$	278
24		$1.03E-12$		$4.413E-15$	234

TABLE II. Numerical results for Example 2, with $\lambda = 1, \sigma = 1$.

Now, in order to check the convergence of the space discretisation, we choose an example, where the time discretisation is exact (does not produce any error).

Example 2. In this example, the functions K and S are the same as in example 1. We set

$$I(x, y, t) = c + t - \tanh(\sigma t)b(\lambda, x, y).$$

As in example 1, $c = 1$. In this case, it is easy to check that the exact solution is

$$V(\bar{x}, t) = t.$$

The initial condition is $V_0(\bar{x}) \equiv 0$. The difference operator (7) is exact for linear functions of t, and this is why the scheme in this case does not have discretisation error in time. Therefore, the observed errors result from the space discretisation. In this case, we are only considering the norm of the error at $t = 0.1$. To check the dependence of the error on λ and σ, we consider 3 different cases: $\lambda = 1, \sigma = 1$; $\lambda = 1, \sigma = 5$; and $\lambda = 5, \sigma = 5$, which are described in tables II, III and IV, respectively.

Since we are using 4 Gaussian points in each subinterval, we expect that the error of the space discretisation is $O(h^8)$. Therefore, when we duplicate the number N of gridpoints, the error should decrease by a factor of approximately $2^8 = 256$.

In order to check the influence of interpolation error, for each N, we consider a set of different values of m (interpolation polynomial degree).

When m increases from 12 to 24, the difference in accuracy is not significant. This means that for values of N up to 96 it is enough to consider $m = 12$. When λ or σ increase we observe that the errors (for the same discretisation step) also increase. This could be expected, since the discretisation error in space depends on the derivatives $\frac{\partial^i K(\bar{x}, y_1, y_2)}{\partial^i y_j}$ and $\frac{\partial^i S(V)}{\partial^i V}$, which increase with λ and σ, respectively.

m	$N=24$	$N=48$	e_{24}/e_{48}	$N=96$	e_{48}/e_{96}
12	$1.62E-10$	$5.52E-13$	293	$2.36E-15$	234
24	$1.69E-10$	$5.33E-13$	317	$2.22E-15$	240

TABLE III. NUMERICAL RESULTS FOR EXAMPLE 2, WITH $\lambda = 5, \sigma = 1$.

m	$N=24$	$N=48$	e_{24}/e_{48}	$N=96$	e_{48}/e_{96}
12	$7.31E-10$	$2.48E-12$	295	$9.38E-15$	264
24	$7.65E-10$	$2.40E-12$	319	$8.94E-15$	268

TABLE IV. NUMERICAL RESULTS FOR EXAMPLE 2, WITH $\lambda = 5, \sigma = 5$.

Example 3. We now test the performance of our algorithm, when applied to a neural field described in [4], which includes the neural field equation with delay (3) (the numerical algorithm for this case is described in sec. III). As mentioned above, equations of this form often arise in Neuroscience applications.

In this example the firing rate function has the form

$$S(x) = \frac{2}{1+e^{-\mu x}},$$

where $\mu \in \mathbb{R}^+$, and the connectivity function is given by

$$K(r) = \frac{1}{\sqrt{2\pi\xi_1^2}} \exp\left(-\frac{r^2}{2\pi\xi_1^2}\right) - \frac{A}{\sqrt{2\pi\xi_2^2}} \exp\left(-\frac{r^2}{2\pi\xi_2^2}\right),$$

where $r = \|x-y\|_2 = \sqrt{(x_1-y_1)^2+(x_2-y_2)^2}$ and $\xi_1, \xi_2, A \in \mathbb{R}^+$. In this example we consider $c = 1$ and in all the simulations we use the initial condition $V_0(x) \equiv 0.01$.

It is known [4] that when there is no external input ($I(x,y,t) = 0$), the stability of the trivial solution of this equation depends on the value of μ, on the parameters of the connectivity function and on the propagation speed. In particular, for each set of values of these parameters, there exists a bifurcation value μ_{bif}, such that if $\mu < \mu_{bif}$ the zero solution is asymptotically stable, and otherwise it isn't.

Let x_1 be a point in the boundary of Ω and x_2 be a point close to the center of the domain. In Fig. 1 (resp. Fig. 2) the graph of $V(x_1,t)$ (resp. $V(x_2,t)$) is displayed, as a function of time, in the case $\xi_1 = 0.4, \xi_2 = 0.2, A = 1$. The behaviour of the solution may be quite different in the two points and it also depends strongly on μ. For $\mu = 10$, for example, we see that after a certain time the solution becomes decreasing, both in x_1 and x_2. But for $\mu = 15$, if t is sufficiently high, the solution increases in both points. This suggests that for some value of μ, between 10 and 15, there should be a bifurcation (the zero solution becomes unstable).

The graphs displayed in Fig.3 correspond to the case $\xi_1 = 0.1, \xi_2 = 0.2, A = 1$. In this figure the graphs on the left-hand side are the surface plots of the solution at equidistant moments in time ($t = 30, 45, 60, 75$), while the graphs on the left-hand side are the corresponding contour plots. As time tends to infinity, we observe that the solution tends to a certain nontrivial stationary state.

All the numerical examples discussed so far are for the case $\tau(x,y) = 0$ (when no delay is considered). It is also interesting

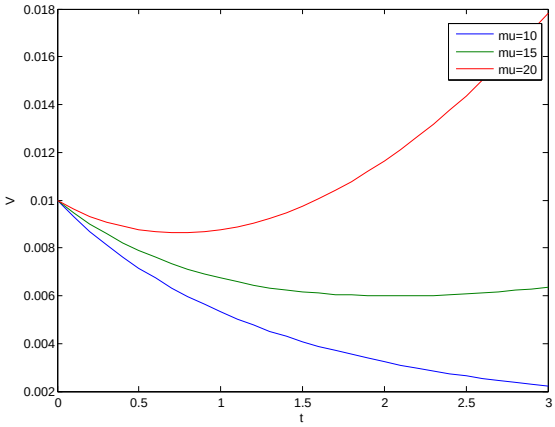

Fig. 1. Evolution of the solution at the boundary of the domain, for different values of μ.

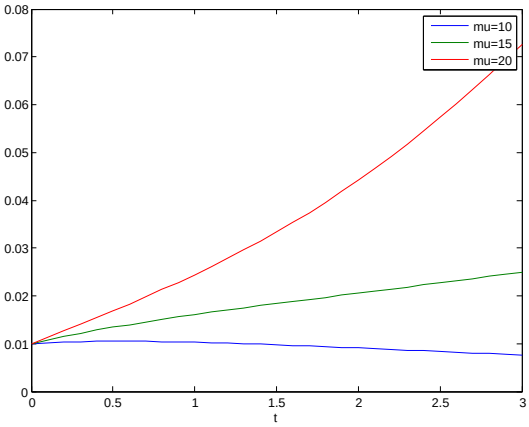

Fig. 2. Evolution of the solution at the boundary of the domain, for different values of μ.

to observe the effect of delay on the behaviour of the solutions. We will now analyse a numerical example which illustrates this effect. Let us consider the case $A = 1$, $\xi_1 = 0.5$, $\xi_2 = 0.4$, $\mu = 27.5$. We simulate this case over the time interval $[0,3]$, with stepsize in time $h_t = 0.1$. First we consider $\tau(x,y) = 0$ (no delay). In this case, the graphs in Fig. 4 show that in the middle of the domain there is an initial period in which the solution grows and then it tends to zero. Knowing that the solution attains its maximum at the middle, this indicates that the zero solution is asymptotically stable.

In order to analyse the effect of delay in the behaviour of solutions we have carried out computations for the same case, but with delay, when the propagation speed is $v = 1$. The numerical results are displayed in Fig. 5. In this case, in the middle of the domain the period during which the solution increases (before attaining its maximum) is much longer, which is obviously an effect of the propagation delay.

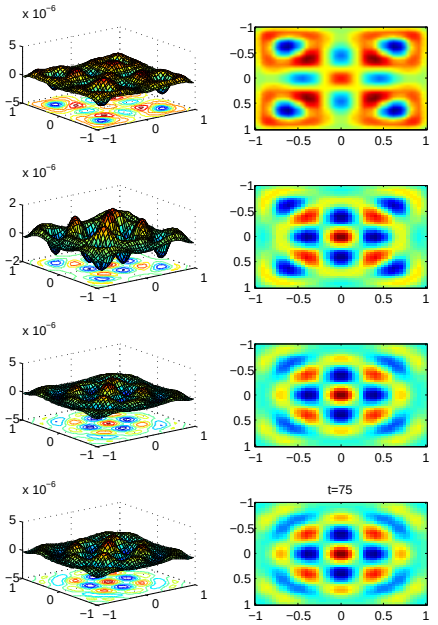

Fig. 3. 3D-graphs (left) and contour plots (right) of the solution at $t = 30, 45, 60, 75$, in the case 1, with $\mu = 45$, with no delay.

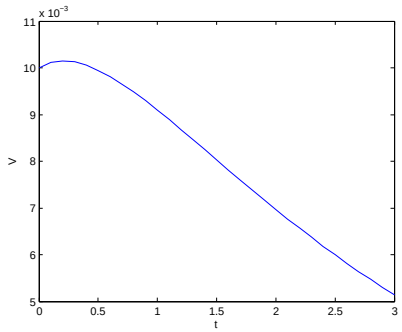

Fig. 4. Graph of $V(x_2, t)$ in the case $A = 1$, $\xi_1 = 0.5$, $\xi_2 = 0.4$, $\mu = 27.5$, without delay.

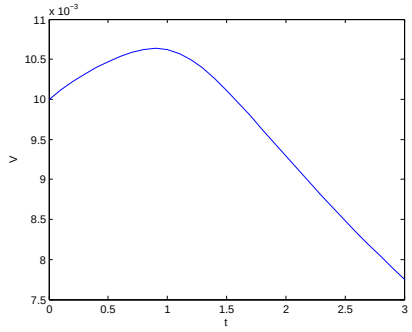

Fig. 5. Graph of $V(x_2, t)$ in the case $A = 1$, $\xi_1 = 0.5$, $\xi_2 = 0.4$, $\mu = 27.5$, with $v = 1$.

V. Conclusion

As other integro-differential equations in several dimensions, the Neural Field Equation represents quite a challenge for numerical computation. In the present paper we have described a new numerical approach to this equation in the two-dimensional case, including a space-dependent delay. A remarkable feature of our method is that we use use a implicit second order scheme for the time discretisation, which improves its accuracy and stability, when compared with the available algorithms. Moreover, to reduce the computational complexity of our method and improve its efficiency we have used an interpolation procedure which allows a drastic reduction of matrix dimensions, without a significant loss of accuracy. In the last section, we have presented three numerical examples. In the first two of them we consider model cases (where the exact solution is known) and we use them to obtain numerical estimates of the convergence rate. These estimates are in agreement with the theoretical results about the method. In the last example we analyse a case which was previously described by other authors [4]. Here we use our algorithm to analyse how the behaviour of the solutions depends on the equation parameters. Our numerical results are in agreement with the expected behaviour of the solutions.

Acknowledgment

This research was supported by a Marie Curie Intra European Fellowship within the 7th European Community Framework Programme (PIEF-GA-2013-629496).

References

[1] S. L. Amari, “Dynamics of pattern formation in lateral-inhibition type neural fields,” *Biol. Cybern.*, vol. 27(2), pp. 77–82, 1977.

[2] A. Cardone, E. Messina, and E. Russo,“ A fast iterative method for discretized Volterra-Fredholm integral equations,” *J. Comput. Applied Math.*, vol. 189, pp. 568–579, 2006.

[3] S. Coombes, “Large scale neural dynamics: Simple and Complex”, *Neuroimage*, vol. 52, pp. 731–739, 2010.

[4] G. Faye and O. Faugeras, “Some theoretical and numerical results for delayed neural field equations”, *Physica D*, vol. 239, pp. 561–578, 2010.

[5] A. Hutt and N. Rougier, “Activity spread and breathers induced by finite transmission speeds in two-dimensional neuronal fields”, *Physical Review E*, vol. 82, pp. 055701, 2010.

[6] P. M. Lima and E. Buckwar, “Numerical solution of the neural field equation in the two-dimensional case ”, submitted.

[7] P. I. Nunez, “The brain wave equation: a model for the EEG,” *Math. Biosci.*, vol. 21, pp. 279–297, 1974.

[8] H. R. Wilson and J. D. Cowan, “Excitatory and inhibitory interactions in localized populations of model neurons,”*Biophys. J.*, vol. 12 , pp. 1–24, 1972.

[9] Weng-Jing Xie and Fu-Rong Lin, “A fast numerical solution method for two dimensional Fredholm integral equations of the second kind,” *Appl. Num. Math.*, vol. 59, pp. 1709–1719, 2009.