The Journal of Open Source Software

NeuralFieldEq.jl: A flexible solver to compute Neural Field Equations in several scenarios

Tiago Sequeira[*1]

1 Instituto Superior Técnico - University of Lisbon

DOI: 10.21105/joss.03974

Software

- Review
- Repository
- Archive

Editor: Jarvist Moore Frost

Reviewers:

- @rougier
- @gdetor

Submitted: 02 November 2021
Published: 03 December 2021

Summary

Since Hodgkin & Huxley (1952) that mathematical modelling proved to be a prominent neuroscience research area. One of its main challenges is to describe phenomena like memory, perception, etc. These neurobiological processes occur in the cortex, a layer of the brain that contains approximately 10^{10} neurons distributed along $2500\,cm^2$ of surface area with $2.8\,mm$ of depth. Driven by such complex structure, Wilson & Cowan (1972) and Amari (1977) derived a model called Neural Field Equation (NFE), that treats the cerebral cortex as a continuum space being able to deal with these large scale dynamical neuronal patterns. The equation has the following form

$$\alpha \frac{\partial V}{\partial t}(\mathbf{x},t) = I(\mathbf{x},t) - V(\mathbf{x},t) + \int_{\Omega} K(||\mathbf{x}-\mathbf{y}||_2)\, S\big[V(\mathbf{y},t)\big]\, d^2\mathbf{y}, \tag{1}$$

where $\Omega = [-\frac{L}{2}, \frac{L}{2}]^d$ with $d = 1, 2$; $V(\mathbf{x},t)$ is the membrane potential at point $\mathbf{x} \in \Omega$ at time t; $I(\mathbf{x},t)$ is the external input applied to the neural field; $K(||\mathbf{x}-\mathbf{y}||_2)$ is the average strength of connectivity between neurons located at points $\mathbf{x}$ and $\mathbf{y}$, when the coupling is positive (negative) the synapses are excitatory (inhibitory); $S(V)$ is the firing rate function, which converts the potential to the respective firing rate result; And α is the constant decay rate.

Since then there have been endeavours to improve NFEs. For example, depending on the cortical region the axonal speed can vary between $100\,m/s$ to $1\,m/s$, so considering the time spent by the stimulus to travel from neurons at y to the ones at x leads us to formulate more accurate models. Or the noisy neuronal interactions, generated by the stochastic nature of the neuron or from extrinsic noise sources that can arise from inputs of other neuronal networks.

`NeuralFieldEq.jl` proposes to efficiently solve NFEs in the mentioned scenarios as well as the combination of stochastic neural fields with finite transmission speed.

Statement of need

The classical quadrature methods to numerically approximate the integral present in Equation 1 have a complexity of $\mathcal{O}^2$ or $\mathcal{O}^4$ in 1D or 2D domains, respectively, making these methods unsuitable for efficient numerical approximations of NFEs. Although the this integral can be seen as a convolution, when considering finite signal velocities is no longer the case. Hutt & P Rougier (2013) proposed a novel numerical scheme that addresses the delayed version of Equation 1 that rewrites the integral into a convolutional form in order to apply a Fourier Transform to it, implying a substantially speed-up when computing delayed NFEs.

*first author

The Journal of Open Source Software

The numerical method that `NeuralFieldEq.jl` implements was developed within the scope of the master's thesis Tiago Sequeira (2021) and arose from the combination of the key idea developed by Hutt and Rougier for delayed neural fields in the stochastic scenario presented by Kuehn & G Riedler (2014), where the authors proved the convergence of spectral methods applied to stochastic NFEs with additive white noise spatially correlated.

The performance of `Julia` (Bezanson et al., 2015) code, when well written, designed and profiled, can be close to `C` or `Fortran` without sacrificing the usual features present in high-level languages. Also, the package makes use of the the multiple dispatch concept, allowing it to be flexible enough to handle NFEs in three different scenarios, 1D or 2D domains, non-delayed or delayed equations and deterministic or stochastic neural fields. These advantages were the trigger needed to develop a new user friendly and fast NFE solver, improving the `Python` solver written by Nichols & Hutt (2015).

These features enabled the numerical results achieved in T. Sequeira & Lima (2021), where stochastic two-dimensional neural fields with low finite axonal speeds were simulated 100 times to better understand the role of the noise in the neural field.

Package usage

The solver is divided into three steps: - Introduce the parameters and functions using the structures `Input1D` or `Input2D`, depending on the domain dimension; - **Remark 1**. Function `I`, depending on the dimensionality of the domain, has to have `x,t` or `x,y,t` as its arguments. Function `K` has `x` or `x,y`. And function `S` with `V`. - **Remark 2**. Currently, to work with the non-delayed problem, the velocity to insert must satisfy the condition: $v > \frac{L}{\sqrt{2}\Delta t}$ in 2D and $v > \frac{L}{2\Delta t}$ in 1D. - Pre-process the NFE using the function `probNFE`; - Solve the equation using the function `solveNFE` at time instants chosen by the user, with or without noise.

Once the solution is computed, we can access it at the previously selected instants. Considering `t=[ti,tj,tk]`, to get the solution at `tj`: `V(tj)` or `V(2)`. In the stochastic case, `Vsto(tj)` stands for the mean solution at `tj`, while for the trajectory `p` is `Vsto(tj,p)`. Also, the user can obtain a specific point in space and time, let `x=[x1,x2,...,xN]` be the discretised space vector, `V(x2,tj)` is the solution's value at `(x2,tj)`.

To illustrate the code usage we will show an example taken from Kulikov et al. (2019).

```
using NeuralFieldEq, Plots
I(x,t) = -2.89967 + 8.0*exp(-x^2/(2.0*3^2)) - 0.5
K(x) = 2*exp(-0.08*sqrt(x^2))*(0.08*sin(pi*sqrt(x^2)/10)+cos(pi*sqrt(x^2)/10))
S(V) = V<=0.0 ? 0.0 : 1.0 # Heaviside function
a  = 1.0  # Constant decay
v  = 20.0 # Finite axonal speed
V0 = 0.0  # Initial condition
L  = 100  # Domain length
N  = 512  # Number of nodes to discretise space
T  = 20.0 # Time span
n  = 200  # Number of nodes to discretise time

nf_1d = Input1D(a,v,V0,L,N,T,n,I,K,S); # 1st step
prob  = probNFE(nf_1d) # 2nd step
tj = [5.0,10.0,20.0]   # Choose instants where the sol is saved
V  = solveNFE(prob,tj) # 3rd step, compute solution
```

If we want to address the stochastic case we simply need to add extra arguments to `solveNFE`.

```
# Solve the stochastic equation 100 times
# Noise magnitude: eps = 0.05. Correlation coefficient: xi = 0.1
Vsto  = solveNFE(prob,tj,0.05,100)      # xi default value 0.1
Vsto2 = solveNFE(prob,tj,0.05,100,0.15) # xi = 0.15

# Handling the solutions
V(10.0)     # Returns the deterministic solution at t=10.0
Vsto(20.0)  # Returns the mean stochastic solution at t=20.0
Vsto(5.0,4) # Returns the 4th trajectory at t=5.0

x = V.x # Returns the spatial vector
plot(x,[V(1),Vsto(1),Vsto(1,4)])
```

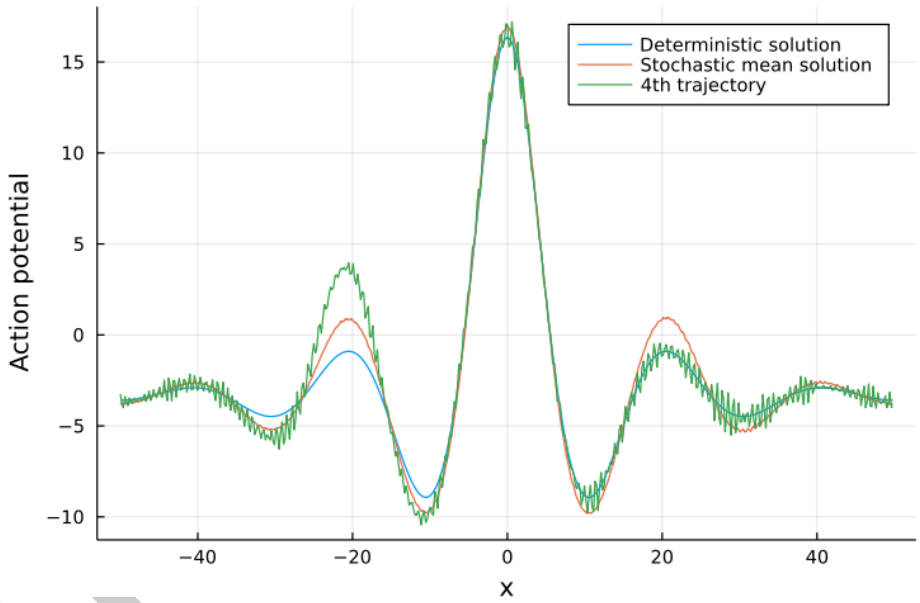

Figure 1: Caption for example figure.

Acknowledgements

A deep thank you to my professor Pedro Lima that kindly reviewed this article.

References

Amari, S. (1977). Dynamic of pattern formation in lateral-inhibition type neural fields. *Biological Cybernetics*, *27*, 77–87. https://doi.org/10.1007/BF00337259

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. (2015). *Julia: A fresh approach to numerical computing*. https://doi.org/10.1137/141000671

Hodgkin, A. L., & Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, *117*(4), 500–544. https://doi.org/10.1113/jphysiol.1952.sp004764

Hutt, A., & P Rougier, N. (2013). Numerical simulation scheme of one- and two dimensional neural fields involving space-dependent delays. *Neural Fields: Theory and Applications*. https://doi.org/10.1007/978-3-642-54593-1_6

Kuehn, C., & G Riedler, M. (2014). Large deviations for nonlocal stochastic neural fields. *Journal of Mathematical Neuroscience*, *4*, 1. https://doi.org/10.1186/2190-8567-4-1

Kulikov, G. Yu., Kulikova, M. V., & Lima, P. M. (2019). Numerical simulation of neural fields with finite transmission speed and random disturbance. *2019 23rd International Conference on System Theory, Control and Computing (ICSTCC)*, 644–649. https://doi.org/10.1109/ICSTCC.2019.8885972

Nichols, E., & Hutt, A. (2015). Neural field simulator: Two-dimensional spatio-temporal dynamics involving finite transmission speed. *Frontiers in Neuroinformatics*, *9*, 25. https://doi.org/10.3389/fninf.2015.00025

Sequeira, Tiago. (2021). *Numerical simulations of one- and two-dimensional stochastic neural field equations with delay* [Master's thesis]. Instituto Superior Técnico - Universidade de Lisboa.

Sequeira, T., & Lima, P. (2021). Numerical simulations of one- and two-dimensional stochastic neural field equations with delay - submitted. *Journal of Computational Neuroscience*.

Wilson, H., & Cowan, J. (1972). Excitatory and inhibitory interactions in localized populations of model neurons. *Biophysical Journal*, *12*, 1–24. https://doi.org/10.1016/S0006-3495(72)86068-5