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:** 07 February 2022



## Summary

In their works, Wilson & Cowan (1972) presented a model, later developed by Amari (1977), named Neural Field Equations (NFE). Bressloff (2011) and Coombes (2005), summarise the main theoretical results and enumerate a wide variety of neurological phenomena that can be described by these equations. Such as working memory (Laing et al., 2002), travelling waves, etc. One of the improvements to consider in the simpler NFE, is to take into account the time taken by the stimulus to travel from neurons at different points. This leads to the delayed equation,

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

with $\Omega = [-\frac{L}{2}, \frac{L}{2}]^k$ with $k = 1, 2$; $V(\mathbf{x},t)$ 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; $K\left(||\mathbf{x}-\mathbf{y}||_2\right)$ is the average strength of connectivity between neurons located at points $\mathbf{x}$ and $\mathbf{y}$ in an isotropic and homogeneous neural field. When the coupling is positive (resp. negative) the synapses are excitatory (resp. inhibitory); $S(V)$ is the firing rate function, which converts the potential to the respective firing rate result; $\alpha$ is the constant decay rate; $d(\mathbf{x},\mathbf{y}) = \frac{||\mathbf{x}-\mathbf{y}||_2}{v}$ represents the delay, which is assumed to only depend on the distance, between points $\mathbf{x}$ and $\mathbf{y}$, and on the transmission speed $v$. If $v$ is sufficiently high, $d$ can be neglected and Equation 1 is reduced to a non-delayed NFE, $d = 0$. Moreover, $V$ satisfies the initial condition $V(\mathbf{x}, t_0) = V_0(\mathbf{x}, t_0),\ t_0 \in [-\tau_{max},\ 0]$, wherein $\tau_{max}$ is the maximum delay correspondent to $\Omega$.

Kuehn & G Riedler (2014) considered the stochasticity inherent to neuronal interactions, and presented a spectral method to stochastic non-delayed NFE with additive white noise and spatial correlation. So, considering a finite speed in this scenario, the following delayed stochastic NFE can be written,

$$\alpha\, dV(\mathbf{x},t) = \left[I(\mathbf{x},t) - V(\mathbf{x},t) + \int_{\Omega} K\left(||\mathbf{x}-\mathbf{y}||_2\right) S\left[V\left(\mathbf{y}, t - d(\mathbf{x},\mathbf{y})\right)\right] d^2\mathbf{y}\right] dt + \epsilon dW(\mathbf{x},t), \quad (2)$$

whereas $\epsilon$ is the additive noise level and $W$ is a $Q$-Wiener process. With $V(\mathbf{x}, t_0) = V_0(\mathbf{x}, t_0),\ t_0 \in [-\tau_{max},\ 0]$.

## Statement of need

Studies suggest that Neural Field Equations, can be applied to cognitive robotics. The architecture of autonomous robots, able to interact with other agents in solving a mutual task, is

*first author

strongly inspired by the processing principles and the neuronal circuitry in the primate brain (Erlhagen & Bicho, 2006). Furthermore, recent efforts made by Ferreira et al. (2020) show a necessity of efficient numerical methods capable of computing NFE in real time, in order to endow robots with working memory mechanisms.

The common numerical methods for Neural Field Equations use the classical quadrature methods, which have $\mathcal{O}\left(N^{2k}\right)$ complexity, where $k$ is the dimension of the domain, therefore, they do not scale well. In the case of non-delayed equations, one can directly apply Fast Fourier Transforms (FFT), $\mathcal{O}\left(N^k \log^k N\right)$, to compute the numerical solution. However, when $v < \infty$, this is no longer the case. Motivated by this, Hutt & P Rougier (2013) proposed a novel numerical scheme, where the authors rewrote the integral in a suitable way, such that, the delayed NFE could be numerically approximated using FFT techniques.

`NeuralFieldEq.jl` aims to enhance the method derived in (Hutt & P Rougier, 2013) and implemented by Nichols & Hutt (2015), and adapt to the stochastic scenario presented by Kuehn & G Riedler (2014). The solver uses Real Fast Fourier Transforms (RFFT) provided by Frigo & Johnson (2005), reducing the overall computations. And is written in `Julia` (Bezanson et al., 2015), which can be performant like low-level languages without sacrificing the usual features of high-level languages. Furthermore, its built to handle NFE in the scenarios aforementioned and their respective combinations, i.e. non-delayed or delayed equations in deterministic or stochastic 1D or 2D neural fields.

Sequeira & Lima (2021) exploited this versatility and efficiency to study stochastic neural fields where low transmission velocities play an important role, facilitating the exploration of new ideas and experiments.

## Package usage

1. Specify parameters, initial condition and functions by using `Input1D` or `Input2D`:
   - The functions are defined as External input: `I(x,t)` or `I(x,y,t)`; Kernel: `K(x)` or `K(x,y)`; And Firing rate: `S(V)`.
   - The initial condition can be constant or a function. In the latter case, must be defined as `V0(x)` or `V0(x,y)`;
   - The parameters: `a`-$\alpha$, `v`-velocity, `L`-domain length, `N`-number of spatial nodes, `T`-time span and `n`-number of time steps;
   - The inputs must be wrapped in the following order: `nf = Input1D(a,v,V0,L,N,T,n,I,K,S)`;
   - **Remark:** Currently, to obtain the non-delayed problem, the velocity must satisfy: $v > \frac{L}{\sqrt{2}\Delta t}$ in 2D and $v > \frac{L}{2\Delta t}$ in 1D.
2. Prepare the NFE through function `probNFE`. Example: `NFE = probNFE(nf)`;
3. Solve the equation using `solveNFE`:
   - Declare an array, `t`, with the time instants where the solution is saved;
   - Compute deterministic equation: `Vdet = solveNFE(NFE,t)`;
   - Compute stochastic equation with noise magnitude `eps`, spatial correlation `xi`, for `np` paths: `Vsto = solveNFE(NFE,t,eps,xi,np)`;
   - **Remark:** Currently, `xi=0.1` is the default value.
4. Handle the output of `solveNFE` (callable object):
   - Call deterministic solution at `ti` $\in$ `t`: `Vdet(ti)`;
   - Call $p^{th}$ trajectory at `ti`: `Vsto(ti,p)`;
   - Call mean stochastic solution at `ti`. Example: `Vsto(ti)`.

The Journal of Open Source Software

## Example and code performance

The example shown below is adapted from Kulikov et al. (2019)

```
using NeuralFieldEq, Plots
I(x,t) = -3.4 + 8.0*exp(-x^2/(2.0*3^2))
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
nf  = Input1D(1.0,20.0,0.0,100,512,20.0,200,I,K,S);
NFE = probNFE(nf)
t   = [5.0,10.0,20.0] # Choose instants where sol is saved
V   = solveNFE(NFE,t) # Compute deterministic solution
Vsto = solveNFE(NFE,t,0.05,100) # eps = 0.05. xi = 0.1. 100 paths
plot(V.x,[V(1),Vsto(1),Vsto(1,4)],xlabel="x",ylabel="Action Potential",
     label=["Det Solution" "Sto Mean Solution" "4th path"])
```

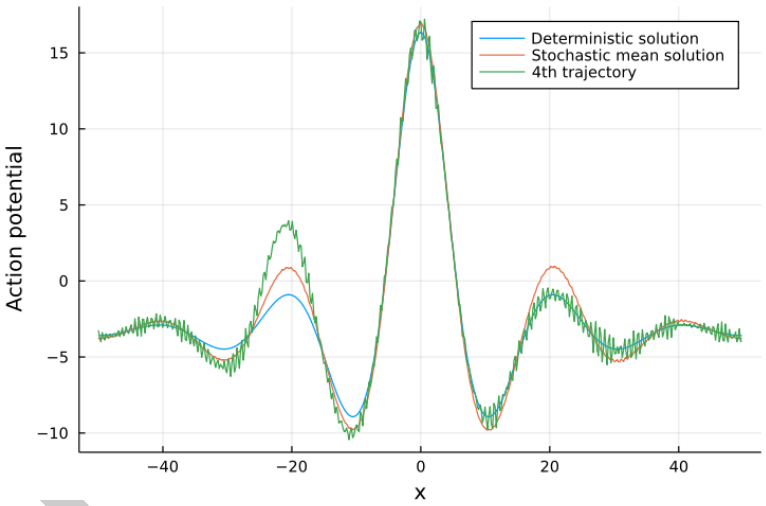


**Figure 1:** Caption for example figure.

Regarding the solver performance, the table below shows the time spent (in seconds) in computing one time step with `N` nodes, for the example listed above and its `2D` version. Both equations were computed with `v=400`.

| $N$ | 1D | 2D |
|---|---|---|
| 128 | 8.6e-6 | 1.4e-3 |
| 256 | 2.1e-5 | 9.2e-3 |
| 512 | 3.1e-5 | 3.8e-2 |
| 1024 | 6.2e-5 | 0.155 |
| 2048 | 1.3e-4 | 0.621 |
| 4096 | 2.6e-4 | 2.72 |

## Acknowledgements

I want to thank my advisor, 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

Bressloff, P. (2011). Spatiotemporal dynamics of continuum neural fields. *Journal of Physics A: Mathematical and Theoretical*, *45*, 033001. https://doi.org/10.1088/1751-8113/45/3/033001

Coombes, S. (2005). Waves, bumps, and patterns in neural field theories. Biol. Cybern. 93(2), 91-108. *Biological Cybernetics*, *93*, 91–108. https://doi.org/10.1007/s00422-005-0574-y

Erlhagen, W., & Bicho, E. (2006). The dynamic neural field approach to cognitive robotics. *Journal of Neural Engineering*, *3*, R36–54. https://doi.org/10.1088/1741-2560/3/3/R02

Ferreira, F., Wojtak, W., Sousa, E., Louro, L., Bicho, E., & Erlhagen, W. (2020). Rapid learning of complex sequences with time constraints: A dynamic neural field model. *IEEE Transactions on Cognitive and Developmental Systems*, *PP*, 1–1. https://doi.org/10.1109/TCDS.2020.2991789

Frigo, M., & Johnson, S. G. (2005). The design and implementation of FFTW3. *Proceedings of the IEEE*, *93*(2), 216–231. https://doi.org/10.1109/JPROC.2004.840301

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

Laing, C., C. Troy, W., Gutkin, B., & Ermentrout, B. (2002). Multiple bumps in a neuronal model of working memory. *SIAM Journal on Applied Mathematics*, *63*. https://doi.org/10.1137/S0036139901389495

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, 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