NPDS Toolbox: Neural Population (De)Synchronization toolbox for Matlab

Mohammad Mahdi Moayeri^{a,1}, Mohammad Hemami^{a,1}, Jamal Amani
Rad^{b,*}, Kourosh Parand^{a,b,c}

^a*Department of Computer and Data Sciences, Faculty of Mathematical Sciences, Shahid
Beheshti University, Tehran, Iran*

^b*Department of Cognitive Modeling, Institute for Cognitive and Brain Sciences, Shahid
Beheshti University, Tehran, Iran*

^c*Department of Statistics and Actuarial Science, University of Waterloo, Waterloo,
Canada*

Abstract

The study of synchronous or asynchronous in (stochastic) neuronal populations is an important concept both in theory and in practice in neuroscience. The NPDS toolbox provides an interactive simulation platform for exploring such processes in Matlab looking through the lens of nonlinear dynamical systems. NPDS includes two main components: neural population (de)synchronization, and neural dynamics. One can investigate distribution controls on various neural models such as HH, FHN, RH, and Thalamic. Also, it supports many numerical approaches for simulation: finite-difference, pseudo-spectral, radial basis function, and Fourier methods. In addition, this toolbox can be used for population phase shifting and clustering.

Keywords: Synchronous, Neuronal population, Phase distribution control, Brain dynamics, Nonlinear dynamical systems, Numerical simulation.

*Corresponding author

Email addresses: mahdi.myr@gmail.com;m_moayeri@sbu.ac.ir (Mohammad Mahdi Moayeri), gaslakh@gmail.com;m_hemami@sbu.ac.ir (Mohammad Hemami), j.amanirad@gmail.com;j_amanirad@sbu.ac.ir (Jamal Amani Rad), k_parand@sbu.ac.ir (Kourosh Parand)

¹These authors contributed equally to this work.

1. Introduction

Synchronization is a vital process in most complex dynamic systems, especially in neural networks within the brain. In fact, various cognitive functions such as decision making, learning, perception, memory, etc. are the result of the synchronization of neuronal populations [1, 2, 3, 4]. However, abnormal and excessive neuronal synchrony in different parts can be one of the reasons for some neurological disorders such as Parkinson’s disease or epilepsy [5]; thus, a fine balance between synchronization and desynchronization is functionally and behaviorally important [6]. Using the control mechanisms in the neural population is a practical approach to modulate seizure activity which is considered by the researchers. Among various types of control strategies [7, 8, 9], neurons phase distribution control [4, 10] is one of the suitable options because it has remarkable features; for instance, in this method, this control takes the analysis of high-dimensional systems more tractable, which reduces the time of solving the problems and improves the performance of the control. It also optimizes the energy consumed by defining a proportional control. Another feature of this method is its applicability to any experimental and neuronal model with respect to its phase response curve (PRC), which makes this technique applicable to any system having a (de)synchronization challenge [10, 4]. However, according to Moehlis et al. [4, 10] and our opinion, the performance of control strategies, especially this phase distribution control, relies heavily on numerical methods to simulate these dynamic nonlinear models, so that using an advanced and more accurate numerical simulation approach to implement the control on the population of synchronized neurons makes the control performance more accurate, minimizes the control energy consumption while achieving the desired control objective. One of the most important ways to achieve these efforts as well as the desired goals is to design a software toolbox. Various powerful software toolboxes have been designed to simulate the dynamic behavior of a neuron and networks, such as Neuron [11] and Brian [12], XPPAUT [13], and bdttoolbox [14], but to the best of our knowledge, no toolbox has ever been developed to simulate the dynamic behavior of synchronous or asynchronous neural networks, as well as to examine professional controllers to change the synchronization behavior of these networks of neurons looking through the lens of their phase distribution.

This toolbox is designed in order to investigate the theories of neuron dynamics and synchronization of (stochastic) neural populations without any

special knowledge of programming and scientific simulations. The main contributions of the work are: (1) controlling the neural oscillators synchronization by phase distribution controls can be simulated without any programming efforts. (2) The proportional controllers, such as bang-bang or user-defined control inputs can be implemented. (3) There are various phase response curves (PRC) related to different neural models such as Hodgkin-Huxley (HH), Fitzhugh-Nagumo (FHN), Rose-Hindmarsh (RH) and Thalamic. (4) The dynamics of the aforementioned models can be investigated in this toolbox. (5) User-defined distributions or well-known distributions such as Von-Mises or uniform ones can be used for initial and final neurons phase distributions. (6) Different numerical approaches are developed for simulations. (7) The dynamics of neuronal populations can be deterministic or stochastic with a Gaussian white noise.

2. Problems and Background

The main purpose of this toolbox is to control the synchronization in populations of identical and uncoupled neural oscillators with/without noise by the phase-based control system which is introduced in [4, 15, 10]. As we mentioned earlier, this model has some remarkable advantages. Not only does it make the analysis of high-dimensional neural dynamical systems more convenient, but it also makes the designing of the control systems experimentally more applicable.

Using phase reduction and expressing the population dynamics with the probability of their distribution, the problem converts to a partial differential equation (PDE). Actually, this PDE depends on the presence or absence of noise. If there is no noise in the system, for each oscillator of the system we have:

$$\dot{\theta}_j = \omega + \mathcal{Z}(\theta)U(t), \quad (1)$$

where $\mathcal{Z}(\theta)$ is the phase response curve (PRC) depending on the neural dynamic model [16] and $U(t)$ is the control input. In addition, $j = 1, \dots, M$ is the number of oscillators of the system. Moreover, the dynamics of the probability distribution $\rho(\theta, t)$ implied in the advection equation is as follows [4, 10]:

$$\frac{\partial \rho(\theta, t)}{\partial t} = -\frac{\partial}{\partial \theta} \left((\omega + U(t)^T \mathcal{Z}(\theta)) \rho(\theta, t) \right). \quad (2)$$

On the other hand, these equations for a noisy system are defined as:

$$\dot{\theta}_j = \delta + \mathcal{Z}(\theta) [u(t) + \sqrt{2D} \eta_j(t)], \quad (3)$$

70 where $\sqrt{2D}\eta(t)$ is a Gaussian white noise with zero mean and variance $2D$
71 affecting the control input. In addition, we consider the following equation
72 for representing the dynamics of the probability distribution of oscillators
73 [17, 18].

$$\frac{\partial \rho(\theta, t)}{\partial t} = -\frac{\partial}{\partial \theta} \left((\omega + U(t)^T \mathcal{Z}(\theta)) \rho(\theta, t) \right) + \mathcal{B} \frac{\partial^2 \rho(\theta, t)}{\partial \theta^2}, \quad (4)$$

74 where

$$\mathcal{B} = \frac{2D}{\pi} \int_0^{2\pi} \mathcal{Z}(\theta) d\theta. \quad (5)$$

75 3. Software Framework and implementation details

76 The toolbox has a GUI which is designed by GUIDE in MATLAB and
77 through which users can evaluate the synchronization or asynchrony of dif-
78 ferent neuronal populations without any programming knowledge and ex-
79 amine them using changing their phase distribution. This toolbox contains
80 eighty MATLAB files and functions to provide numerous options for users in
81 simulations including different options in numerical methods, default control
82 algorithms, population distributions, and neural models. Moreover, there are
83 some options that allow users to explore their models with simple user-defined
84 MATLAB codes. NPDS toolbox includes two main parts described below:
85 The most important and main part is neural population (De)synchronization
86 where one can investigate distribution control models on various neural mod-
87 els such as HH, FHN, RH, and Thalamic neuron models. In order to evaluate
88 the obtained results, there exist different kinds of numerical approaches for
89 simulation. The current version (1.0) supports the 5-point stencil finite dif-
90 ference (FD) method, generalized Lagrange Jacobi pseudo-spectral method,
91 radial basis function (RBF) method, radial basis function generated finite
92 difference method (RBF-FD), and Fourier decomposition method. More-
93 over, we develop fourth-order Runge-Kutta algorithms to solve ODEs (1)
94 and stochastic ODEs (3). In addition to synchronization and desynchroniza-
95 tion, this toolbox can be used for population phase shifting and clustering
96 which has many uses. For instance, clustering has application in rewiring
97 neural plasticity synaptic connections between neurons. For this purpose,
98 one can define arbitrary initial and final distributions and choose a numeri-
99 cal algorithm to investigate the control algorithm response. This toolbox is

100 developed in such a way that users can follow the control strategy perfor-
101 mance in simulation and evaluate their control algorithm in terms of accuracy,
102 convergence, and efficiency according to the selected numerical method.

103 Another part of NPDS toolbox is neural dynamics which interested users
104 in dynamical systems might be attracted to. There are some powerful compu-
105 tational toolkits for simulation of neural dynamical models such as Xppaut
106 [13], Virtual Brain [19] and Brain Dynamics Toolbox [14]. NPDS toolbox
107 users can examine the dynamics of the models introduced in the Neural pop-
108 ulation (De)synchronization part as neuronal models. They can change the
109 parameters or initial condition of each model and see their impact on the
110 dynamics of the problem. Moreover, there are some useful options such as
111 showing phase portraits, adding the vector field and stream to the figures or
112 reporting the CPU time and dynamical system state. In fact, this part com-
113 plements the previous one and provides users with more complete information
114 about the neural toolbox model. This toolbox uses standard ODE solvers
115 (`ode23tb`, `ode45` and `ode15s`) to solve the neurons dynamical systems.

116 We provide a diagram (please see the figure here: <https://github.com/cmplab/npds-toolbox/blob/main/docs/Pictures/Arch.png>) to dis-
117 play the architecture of the toolbox functions and the relations between
118 them. In this figure, the sources of graphical user interface files are rep-
119 resented by rectangles. We have two main part i.e `NPDSToolbox.m` and
120 `NeuronDynamic.m`. These files are the main parts of the toolbox, which
121 are represented by two diamonds and can be run from the command line
122 directly. Moreover, `About.m` can be run from the command line, but it is
123 not one of the main files of the toolbox and just gives a brief overview of the
124 toolbox. This file is shown by a diamond in the figure. There are some main
125 functions. These functions call the other functions to do their task correctly.
126 On the other hand, regular functions are called by the main ones and these
127 functions do not need to call other functions. These two types of functions
128 are displayed by two ellipses and one ellipse, respectively. Some functions
129 invoke simple functions defined inside the same file. Simple functions are
130 shown by the ellipse dotted line. A diamond inside a rectangle expresses a
131 static file that creates a user-defined function file when the user intends to
132 define a new function. Finally, the cloud-like shape is `PARAMETER_GUIDE.md`
133 file which is a guide for model parameters.
134

135 4. Illustrative Examples

136 In this section, examples of the two main parts of the toolbox are pro-
 137 vided. These examples are based on one of the neuronal models named
 138 the Rose-Hindmarsh (RH) model. First, we use the NeuronDynamic GUI
 139 to investigate RH neuron dynamic and show the effects of values of pa-
 140 rameters on the state (resting, bursting, damping, and transition states) of
 141 the dynamical system. Then, in the next section, a control strategy is de-
 142 signed to desynchronize an RH neural population using the neural population
 143 (de)synchronization part, and we describe how to define user-defined initial
 144 distribution and control strategy.

145 4.1. Neurondynamic

146 The RH dynamical system is defined as follows [17, 18]

$$\begin{aligned}\dot{x} &= y - ax^3 + bx^2 - z + I, \\ \dot{y} &= c - dx^2 - y, \\ \dot{z} &= r(s(x - X_r) - z),\end{aligned}$$

147 where x is a dimensionless variable related to the membrane potential. Also,
 148 y is called the spiking variable and measures the rate of transportation
 149 of sodium and potassium ions, and z corresponds to adaptation current.
 150 a, b, c, d, r, s, X_r are model parameters. Moreover, I is the applied cur-
 151 rent to the neuron. The model parameters in the toolbox are explicitly easily
 152 modifiable. Additionally, the initial values and the applied current can be
 153 selected from specified intervals whose bounds can be changed by the users.
 154 Note that in the toolbox toolbar, the button makes to change the inter-
 155 vals of the initial values and the applied current to the desired ones. The left
 156 and right plots in the display panel show the dynamic and phase portraits
 157 of the model, respectively. Figure 1 shows four different states including
 158 *resting*, *burst*, *damping* and *transition* by changing the model parameters
 159 and intervals. The buttons and add vector field and stream to the
 160 phase portrait, used in figures 1a, 1b and 1c. Also, in Figures 1b and 1c, both
 161 portraits have grids in different sizes using buttons and . By selecting
 162 the checkboxes below the dynamic portrait, one can select the variables to
 163 show in the figure. In order to display the results more appropriately, the
 164 checkbox *scale mode* scales the behavior of the variables in the interval $[0, 1]$
 165 (See Figure 1c). In addition, the checkboxes below the phase portrait specify

Figure 1: Different dynamic states of the Rose-Hindmarsh model.

the coordinate axes of the phase portrait. At the same time, by selecting the 3D checkbox, the three-dimensional phase portrait of models with more than two variables can be demonstrated (see figure 1d). Finally, *neuron type*, *dynamic state*, and CPU time are reported in the below text box.

4.2. Neural population (De)synchronization

As the next test case, we tend to desynchronize a semi-synchronous population of RH neural oscillators by a noisy bang-bang control. A semi-synchronous population of neural oscillators can be interpreted as a partial synchronization of neural oscillators in two distinct zones that are poorly connected. To apply the desynchronization process, the initial semi-synchronous population should be presented as a distribution. How to define a new distribution in the toolbox is described in the following section.

4.3. Defining initial and final distribution

In the toolbox, Von-Mises (single-peak) and uniform distributions are provided as defaults. However, the initial distribution in this example is

equivalent to a two-peaks distribution. According to the Von-Mises distribution, the multi-peaks distributions can be like a moving wave with a constant velocity which is derived from Von-Mises distribution modification as follows [17, 18]:

$$\sum_{i=1}^p \frac{\alpha_i \exp(c_i \cos(\Omega - l_i - \omega \Delta \tau))}{2\pi \beta_0(c_i)}, \quad (6)$$

where α_i are average coefficients, p is the number of peaks, c_i and l_i are the concentration and location of each peak, β_0 is the first kind of Bessel function. Note that the following condition should always behold in the distribution design

$$\sum_{i=1}^p \alpha_i = 1. \quad (7)$$

The parameters Ω , ω , $\Delta \tau$, $\beta_0(\cdot)$ are equivalent to the keywords `doamin`, `omega`, `i*dt`, `besseli(0,.)`, respectively. In addition, in order to perform the desynchronization process, apart from the initial distribution, its derivative should also be implemented. So, the desired initial distribution and its derivative can defined as variables `dist` and `dif_dist`. Consider the following example.

```
function [dist,dif_dist]=user_defined_initial_dist(
    domain,location,concentration,omega,i,dt)
conc=4;
alpha=0.5;
loc=pi/10;
%Defining initial distribution
dist=alpha*exp(conc*cos(domain-loc-omega*i*dt))/(2*pi*
    besseli(0,conc))...
    +alpha*exp(conc*cos(domain-6*loc-omega*i*dt))/(2*pi
    *besseli(0,conc));
%Defining derivative of initial distribution
dif_dist=alpha*(-conc*sin(domain-loc-omega*i*dt)).*exp
    (conc*cos(domain-loc-omega*i*dt))...
    /(2*pi*besseli(0,conc))+alpha*(-conc*sin(domain-6*
    loc-omega*i*dt)).*...
    exp(conc*cos(domain-6*loc-omega*i*dt))/(2*pi*
    besseli(0,conc));
end
```


213 4.4. A user-defined control strategy

214 In the current version of the toolbox, the appropriate and simple control
215 and explosion inputs can be used in a predefined way. However, in this sec-
216 tion, we want to show how the user defines a custom controller, for example,
217 a noisy bang-bang control input as follow

$$\begin{cases} \eta + u_{\max}, & I \geq 0, \\ \eta - u_{\min}, & I < 0, \end{cases}$$

$$I = \int_0^{2\pi} (\rho(\theta, \Delta\tau) - \rho_f(\theta, \Delta\tau)) \mathcal{Z}(\theta) \rho(\theta, \Delta\tau) d\theta,$$

$$\eta = cf * error(i) \mathcal{N}(0, 1),$$

218 where $u_{\max}$ and $u_{\min}$ are upper and lower limits of control input, $error(i)$ is
219 error of result in step i and cf is a constant. To design this control we need a
220 domain of the problem, current distribution, final distribution, PRC, current
221 error matrices, and current step which are determined with keywords **domain**,
222 **phi**, **phif**, **prc**, **error(.)**, and **iteration_number**, respectively. Also, the
223 desired control can define as a variable **u**.

224 Consider the following example:

```
225 function u=user_defined_control(varargin)
226     .
227     .
228     .
229
230     cf=50;
231     %Defining the noisy bang-bang control
232     I=(trapz(domain, (phi-phi_f).*prc'.*phi));
233     u_max=12; % Upper limit
234     u_min=-12; % Lower limit
235     r=cf*error(iteration_number)*rand(1,1);
236     if I>0
237         u=u_max+r;
238     else
239         u=u_min-r;
240     end
```

241 Figure 2 shows the initial and final results of performing the introduced
242 desynchronization process. According to this figure, we can evaluate the

Figure 2: The result of applying noisy bang-bang control to desynchronize a population of neural oscillators.

243 proposed control strategy that can desynchronize the neural oscillators by
 244 radial basis function generated finite difference method [18] with consuming
 245 about 1803 units of energy.

246 5. Conclusions

247 The NPDS toolbox is an interactive graphical tool in which students/
 248 engineers/researchers in computational neuroscience can evaluate the syn-
 249 chronization or asynchrony of user-defined (stochastic) neuronal populations
 250 without any programming knowledge and examine professional controllers
 251 to change the synchronization behavior of these networks of neurons looking
 252 through the lens of their phase distribution. This graphical interface imposes
 253 no limit on the size of the neural population, the type of dynamics of the
 254 neurons involved, the design of the phase change controller, nor the numeri-
 255 cal simulation approach. The NPDS will continuously be extended with new
 256 features such as adding coupled neuronal populations as neural networks,
 257 improving numerical simulation approaches, and adding more PRCs as well
 258 as user-defined PRCs. Once any new feature is implemented, it can be easily
 259 shared with other toolbox users.

260 Acknowledgements

261 MATLAB[®] is a registered trademark of The Mathworks, Inc., 3 Apple
262 Hill Drive, Natick, MA 01760–2098 USA, 508-647-7000, Fax 508-647-7001,
263 info@mathworks.com, www.mathworks.com.

264 References

- 265 [1] L. Glass, Synchronization and rhythmic processes in physiology, *Nature*
266 410 (2001) 277–284.
- 267 [2] T. Womelsdorf, P. Fries, The role of neuronal synchronization in selective
268 attention, *Curr. Opin. Neurobiol.* 17 (2007) 154–160.
- 269 [3] P. J. Uhlhaas, W. Singer, Neural synchrony in brain disorders: relevance
270 for cognitive dysfunctions and pathophysiology, *Neuron* 52 (2006) 155–
271 168.
- 272 [4] B. Monga, G. Froyland, J. Moehlis, Synchronizing and desynchroniz-
273 ing neural populations through phase distribution control, 2018 Annual
274 American Control Conference (ACC) (2018) 2808–2813.
- 275 [5] C. P. Warren, S. Hu, M. Stead, B. H. Brinkmann, M. R. Bower, G. A.
276 Worrell, Synchrony in normal and focal epileptic brain: The seizure on-
277 set zone is functionally disconnected, *J Neurophysiol.* 104 (2010) 3530–
278 3539.
- 279 [6] A. Schnitzler, J. Gross, Normal and pathological oscillatory communi-
280 cation in the brain, *Nat. Rev. Neurosci.* 6 (2005) 285–296.
- 281 [7] W. Yu, J. Cao, W. Lu, Synchronization control of switched linearly
282 coupled neural networks with delay, *Neurocomputing* 73 (2010) 858–
283 866.
- 284 [8] M. A. A. Ahmed, Y. Liu, W. Zhang, F. E. Alsaadi, Exponential synchro-
285 nization via pinning adaptive control for complex networks of networks
286 with time delays, *Neurocomputing* 225 (2017) 198–204.
- 287 [9] Y. Ma, Y. Zheng, Projective lag synchronization of markovian jumping
288 neural networks with mode-dependent mixed time-delays based on an
289 integral sliding mode controller, *Neurocomputing* 168 (2015) 626–636.

- 290 [10] B. Monga, J. Moehlis, Phase distribution control of a population of
291 oscillators, *Physica D.* 398 (2019) 115–129.
- 292 [11] N. T. Carnevale, M. L. Hines, *The NEURON Book*, Cambridge Univer-
293 sity Press, 2006.
- 294 [12] D. F. M. Goodman, R. Brette, *BRIAN simulator*, *Scholarpedia* 8 (2013)
295 10883.
- 296 [13] B. Ermentrout, *Simulating, analyzing, and animating dynamical sys-*
297 *tems: a guide to XPPAUT for researchers and students*, SIAM, 2002.
- 298 [14] S. Heitmann, M. J. Aburn, M. Breakspear, *The brain dynamics toolbox*
299 *for Matlab*, *Neurocomputing* 35 (2018) 82–88.
- 300 [15] B. Monga, D. Wilson, T. Matchen, J. Moehlis, Phase reduction and
301 phase-based optimal control for biological systems: a tutorial, *Biol Cy-*
302 *bern.* 113 (2018) 11–46.
- 303 [16] E. Brown, J. Moehlis, P. Holmes, On the phase reduction and response
304 dynamics of neural oscillator populations, *Neural Comput.* 16 (4) (2004)
305 673–715.
- 306 [17] M. M. Moayeri, J. A. Rad, K. Parand, Desynchronization of stochasti-
307 cally synchronized neural populations through phase distribution con-
308 trol: a numerical simulation approach, *Nonlinear Dyn.* 104 (2021) 2363–
309 2388.
- 310 [18] M. Hemami, J. A. Rad, K. Parand, Phase distribution control of neural
311 oscillator populations using local radial basis function meshfree tech-
312 nique with application in epileptic seizures: A numerical simulation
313 approach, *Commun. Nonlinear. Sci. Numer. Simul.* (2021) 105961doi:
314 <https://doi.org/10.1016/j.cnsns.2021.105961>.
- 315 [19] V. K. Jirsa, O. Sporns, M. Breakspear, G. Deco, A. R. McIntosh, To-
316 wards the virtual brain: network modeling of the intact and the damaged
317 brain, *Arch Ital Biol.* 148 (3) (2010) 189–20.

318 Required Metadata

319 Current executable software version

Nr.	(executable) Software metadata description	
S1	Current software version	V 1.0
S2	Permanent link to executables of this version	https://github.com/cmplab/npds-toolbox
S3	Legal Software License	BSD-3-Clause License
S4	Computing platform/Operating System	Matlab 2016a or newer
S5	Installation requirements & dependencies	None
S6	If available, link to user manual - if formally published include a reference to the publication in the reference list	https://npds.readthedocs.io/en/latest/
S7	Support email for questions	cmplab@sbu.ac.ir

Table 1: Software metadata (optional)

320 Current code version

Nr.	Code metadata description	
C1	Current code version	V 1.0
C2	Permanent link to code/repository used of this code version	https://github.com/cmplab/npds-toolbox
C3	Legal Code License	BSD-3-Clause License
C4	Code versioning system used	Github
C5	Software code languages, tools, and services used	Matlab 2016a or newer
C6	Compilation requirements, operating environments & dependencies	None
C7	If available Link to developer documentation/manual	https://npds.readthedocs.io/en/latest/
C8	Support email for questions	cmplab@sbu.ac.ir

Table 2: Code metadata (mandatory)