NMMGenerator: An automatic neural mass model generator from population graphs

Maxime Yochum, Julien Modolo

NMMGenerator: An automatic neural mass model generator from population graphs

Maxime Yochum

Univ Rennes, INSERM, LTSI - U1099, F-35000 Rennes, France

E-mail: maxime.yochum@inserm.fr

Julien Modolo

Univ Rennes, INSERM, LTSI - U1099, F-35000 Rennes, France

E-mail: julien.modolo@inserm.fr

December 2019

Abstract. Neural mass models are among the most popular mathematical models of brain activity, since they enable the rapid simulation of large-scale networks involving different neural types at a spatial scale compatible with electrophysiological experiments (e.g., local field potentials). However, establishing neural mass model (NMM) equations associated with specific neuronal network architectures can be tedious and is an error-prone process, restricting their use to scientists who are familiar with mathematics. In order to overcome this challenge, we have developed a user-friendly software that enables a user to construct rapidly, under the form of a graph, a neuronal network with its populations and connectivity patterns. The resulting graph is then automatically translated into the corresponding set of differential equations, which can be solved and displayed within the same software environment. The software is proposed as open access, and should assist in offering the possibility for a wider audience of scientists to develop NMM corresponding to their specific neuroscience research questions.

Keywords: Neural mass models, Connectivity graphs, Local field potentials.

1. Introduction

Neural mass models (NMMs) are a popular class of models that simulate the electrical activity of neuronal assemblies. NMMs are extensively used in neuroscience, since those models reproduce electrophysiological signals recorded from healthy [1, 2, 3] or impaired brains as in Alzheimer's disease [4] or epilepsy [5, 6]. The main principle of a NMM is to consider populations of neurons instead of individual neurons, connected by excitatory and inhibitory projections between those populations [7, 8, 9]. Interestingly, NMMs are able to reproduce realistic local field potential (LFP) signals recorded experimentally,

since their spatial scale of description is compatible with the sources originating LFPs. Successful examples of NMMs application are evoked potentials [10, 11], epileptiform activities [12] or cortical connectivity [13]. More recently, NMMs have been used to understand the effect of deep brain stimulation on the activity of targeted regions [14] or the effect of sleep regulation [15]. A significant advantage of NMMs is that they include key physiological information: the connection between modeled populations corresponds to actual neuronal synaptic connections, the firing rate (FR) to post-synaptic potential (PSP) conversion uses a sigmoid function based on experimental data [16], and the PSP to FR conversion uses a 2nd order low-pass filter transcribing the synaptic conversion process and includes physiological time constants.

A state-space representation of a NMM can be derived from a graph representing populations and connections among them, as shown in figure 1. This state-space representation results in a set of ordinary differential equations (ODE). The main issue with this representation is that the ODE set must be rewritten for each graph. More precisely, if a population, or a link between sup-populations, or an external input is added or removed, the ODE equation set of the neural mass model needs to be rewritten. Therefore, it can be tedious to compute by hand the set of ODE equations associated with NMM graphs. In addition, this step can be difficult for researchers not familiar with mathematical or computer coding operations.

In this paper, we propose an innovative method that automatically generates the NMM state-space ODE directly from a user-generated graph. The graph features populations and associated projections (links) them, where populations can represent one class or type of neuron (pyramidal / basket cell, or glutamatergic / GABAergic for instance). The automatic conversion method was implemented in an open-source software, enabling the streamlined design of NMM graphs. Our software then creates a python class model from a NMM graph that can be directly used to run simulations. The python class model can be used directly through the python interpreter or can be used in a basic computation GUI (provided within the software), where the parameters of the model (e.g., synaptic gains, time constants) can be modified and the model output (LFPs, pulse densities and PSPs) can be displayed.

A software proposed by [17] was proposed to study coupling between two NMMs. Contrary to our software, the complexity to incorporate physiological knowledge into the model equations is not addressed. Indeed, the user needs to create directly block diagrams, which is a step made automatically in our software using physiology-based graphs. Furthermore, the software proposed by [17] requires Matlab and Simulink knowledge, and are not free, which can refrain from the use of these tools by users not familiar with mathematical or computer coding operations.

The software described here enables the generation of source-level (LFP) signals, that therefore cannot be interpreted directly as EEG signals. Indeed, source-level signals are summed (mixing problem) at the level of EEG electrodes, therefore obtained EEG signals from the LFP signals generated with our software requires having a head model at disposal (head and tissue geometries, conductivities, EEG sensor positions). Once

the head model is known, a lead-field matrix describing the contribution of each source (neuronal population) to the signal recorded by each EEG sensor can be computed, and used to project LFP signals onto EEG sensors, i.e. obtaining EEG signals.

2. Material and Methods

2.1. Generalities about NMMs

A NMM is a model accounting for a large number of neurons in a local brain network. The NMM is then an average of individual neuronal states, corresponding to post-synaptic potentials and firing rates of several populations (principal cells, excitatory or inhibitory interneurons). NMMs feature two kind of conversions: the first one is a FR to PSP conversion, called "pulse-to-wave". It is a linear transformation which is modeled by a 2nd order low-pass filter, where the impulse response (Green's function) is given by:

$$h_k = \frac{H_k}{\tau_k} e^{-\frac{t}{\tau_k}}, \tag{1}$$

where H is the synaptic gain, τ is the time constant (its reciprocal is denoted λ) and k is the considered population with its neurotransmitter type (e.g., glutamate, GABA). Usually, this filter is represented as a set of two 1st-order ordinary differential equations given by:

$$\begin{aligned} \dot{y}_{k_a}(t) &= y_{k_b}(t) \\ \dot{y}_{k_b}(t) &= H_k \lambda_k x(t) - 2\lambda_k y_{k_b}(t) - \lambda_k^2 y_{k_a}(t) \end{aligned} \tag{2}$$

where $x(t)$ is the input signal (a firing rate of another population) and $y_{k_a}(t)$ is the output signal (the post-synaptic potentials).

The second one is a PSP to FR conversion which is called "wave-to-pulse", converting the average membrane potential of a population into the average firing rate of the same population which is the output of the population. This conversion is non-linear and typically takes the form of a sigmoid function.

$$sigm_k(v) = \frac{2e_{0_k}}{1 + e^{r_k\left(v_{0_k} - v\right)}}, \tag{3}$$

where e_{0_k} is half of the maximal population firing rate, v_{0_k} is the average membrane potential for which e_{0_k} is reached, and r_k is the slope. In the model, the interactions between populations are modeled using connectivity constants named C_k, accounting for the average number of synaptic contacts. In this paper, since an interconnection is considered as unidirectional, the general format of connectivity constants is $C_{k_e - k_r}$, where k_e is the population emitting the FR, and k_r is the population receiving the FR. Noise can be added to the model to account for the average density of afferent action potentials, and is considered as the external input of the neural mass model. This noise

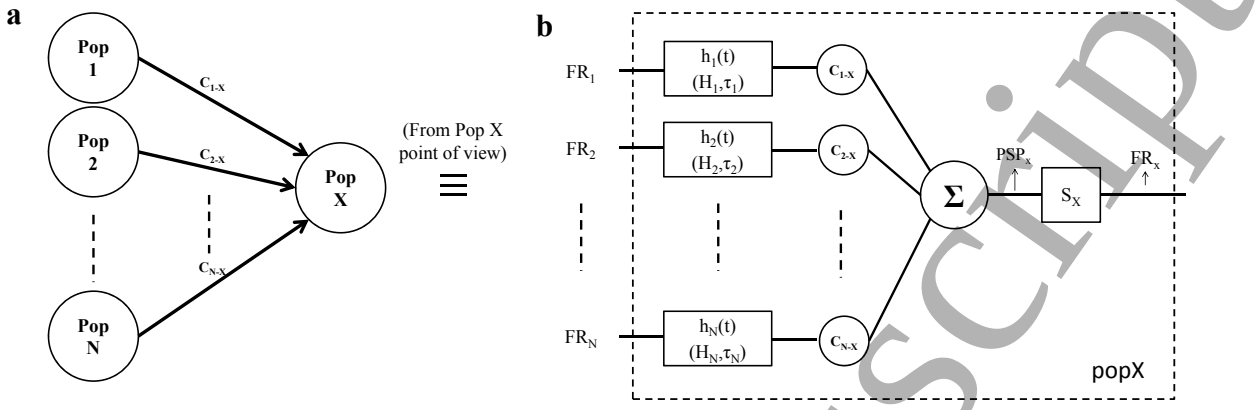

Figure 1. (a). Connectivity graph where only our population of interest (Pop_X) is shown, with all population projecting towards it. Every population from Pop_1 to Pop_N is emitting towards Pop_X through a connectivity parameter C_i. **(b)**. Corresponding diagram that allows to compute FR_X and PSP_X of Pop_X.

is usually a Gaussian white noise (denoted $p_k(t)$) which is excitatory. However, in our implementation, let us mention that it could also be inhibitory.

2.2. Graph and block diagram representation of a population

Each population of a NMM can be seen as an independent graph or a block diagram. A general example of such representations is shown in figure 1 where the population of interest is the Pop_X. In Panel a, several populations $Pop_{1\text{-}N}$ are connected toward the Pop_X through connectivity parameters $C_{(1-N)-X}$ which are considered as the weights of connections. This graph is equivalent to the block diagram represented in panels b to d. The populations $Pop_{1\text{-}N}$ send their FRs to the population Pop_X which are then converted to PSPs with the "pulse-to-wave" filter. Those PSPs are then multiplied by connectivity constants before being summed up to give the total PSP_X. Note that if a synaptic contact is inhibitory, the corresponding PSP must be inverted (multiply by -1) to account for the hyper-polarization of the membrane instead of a depolarization. To be able to compute the PSP_X, we need to convert the firing rate that are projected to the Pop_X. Therefore, a set of ODEs (see eq. 2) must be created for each link that exists between populations projecting toward the population Pop_X. So, for the Pop_X, we obtain the following system of ODEs:

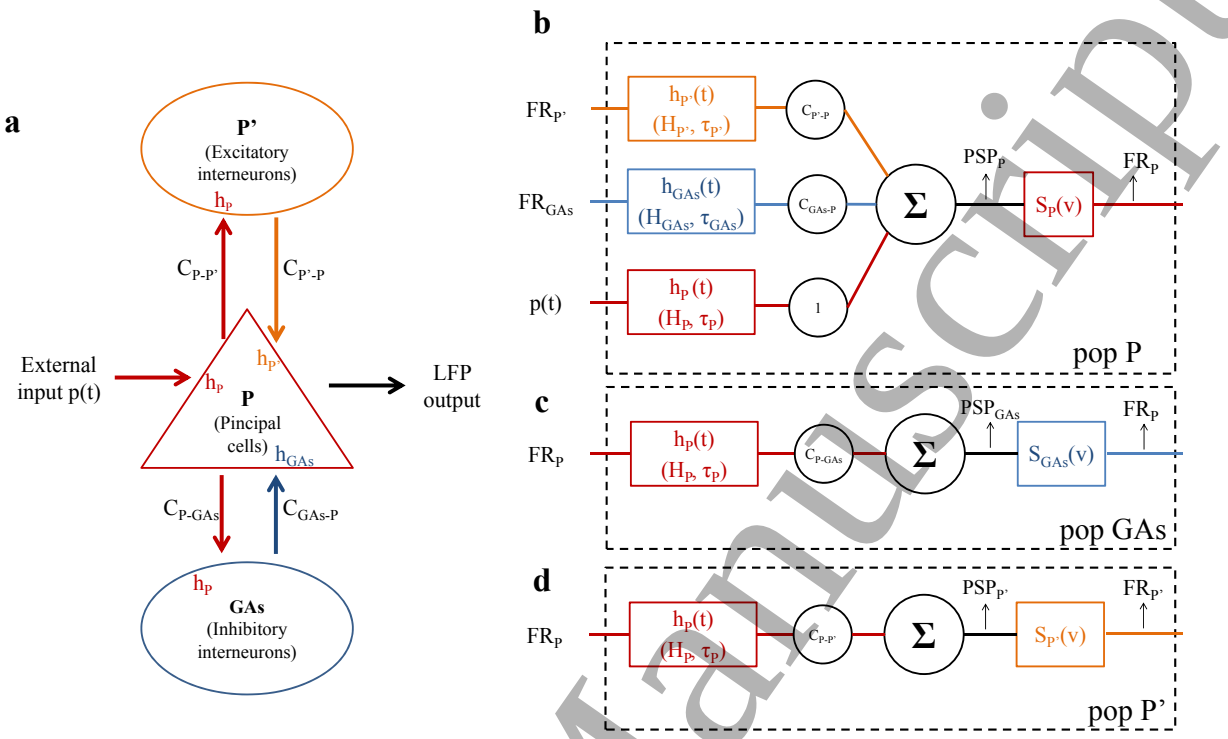

Figure 2. (a). Jansen-Rit NMM with three populations (Principal cells, Excitatory interneurons and inhibitory interneurnons). **(b)**. Block diagram for the principal cells population. **(c)**. Block diagram for the excitatory interneurons population. **(d)**. Block diagram for the inhibitory interneurons population.

$$
\begin{aligned}
\dot{y}_{1_a}(t) &= y_{1_b}(t) \\
\dot{y}_{1_b}(t) &= C_{1-X} H_1 \lambda_1 F R_1 - 2\lambda_1 y_{1_b}(t) - \lambda_1^2 y_{1_a}(t) \\
\dot{y}_{2_a}(t) &= y_{2_b}(t) \\
\dot{y}_{2_b}(t) &= C_{2-X} H_2 \lambda_2 F R_2 - 2\lambda_2 y_{2_b}(t) - \lambda_2^2 y_{2_a}(t) \\
&\dots \\
\dot{y}_{N_a}(t) &= y_{N_b}(t) \\
\dot{y}_{N_b}(t) &= C_{N-X} H_N \lambda_N F R_N - 2\lambda_N y_{N_b}(t) - \lambda_N^2 y_{N_a}(t)
\end{aligned}
\tag{4}
$$

2.3. Graph and block diagrams representation of multi-population network

In the following, we use as an example the Jansen and Rit NMM [11], which is one of the simplest NMM that we found in the literature. Figure 2 presents the graph (panel (a)) associated with the Jansen-Rit NMM. Three populations are present in this model: principal / pyramidal cells (P: Glutamatergic), excitatory interneurons (P': Glutamatergic) and inhibitory interneurons (GAs: GABA_A,slow). The principal cells population is connected to both excitatory and inhibitory interneurons (through $C_{P-P'}$ and C_{P-GAs} respectively), which in return are connected to Principal cells (through $C_{P'-P}$ and C_{GAs-P}). An external input is added to the principal cells (p(t)).

The complete graph can be decomposed into three separate block diagrams, one for each modeled population. Then, panel b stands for principal cells (P), panel c stands for excitatory interneurons (P'), and panel d stands for inhibitory interneurons (GAs). Note that there exists one "wave-to-pulse" (sigmoid function) per population, and one "pulse-to-wave" (2nd order filter) per link toward the populations (external input included). The number of ODE in the ODE set is defined the number of links between populations (including the noise populations). However, some "pulse-to-wave" functions can be redundant, which is the case for $h_P(FR_P)$ of the block diagram of Figure 2 (c) and 2 (d). Indeed, those two pulse-to-wave function are identical (H_P and τ_P along with their inputs FR_P) from a mathematical point of view, since those two functions are coming from the same population (pyramidal cells P). Then, their outputs are identical and those two functions can be combined into a single pulse-to-wave function. It is then possible to convert the entire graph with two methods: one 2nd order ODE per link (without pulse-to-wave function combination), or one 2nd ODE per population (with pulse-to-wave function combination), the external input counting as a population.

2.3.1. One 2nd order ODE per link. To derive the ODE set, each link must give a 2nd order ODE according to eq. 2. Indeed, as apparent from the panels b-d in figure 2, each link corresponds to a "pulse-to-wave" function. We can then write:

$$\begin{aligned}
\dot{y}_{P_a}(t) &= y_{P_b}(t) \\
\dot{y}_{P_b}(t) &= C_{P'-P} H_P \lambda_P FR_{P'} - 2\lambda_P y_{P_b}(t) - \lambda_P^2 y_{P_a}(t) \\
\dot{y}_{P_c}(t) &= y_{P_d}(t) \\
\dot{y}_{P_d}(t) &= C_{GAs-P} H_P \lambda_P FR_{GAs} - 2\lambda_P y_{P_d}(t) - \lambda_P^2 y_{P_c}(t) \\
\dot{y}_{P'_a}(t) &= y_{P'_b}(t) \\
\dot{y}_{P'_b}(t) &= C_{P-P'} H_{P'} \lambda_{P'} FR_P - 2\lambda_{P'} y_{P'_b}(t) - \lambda_{P'}^2 y_{P'_a}(t) \\
\dot{y}_{GAs_a}(t) &= y_{GAs_b}(t) \\
\dot{y}_{GAs_b}(t) &= C_{P-GAs} H_{GAs} \lambda_{GAs} FR_P - 2\lambda_{GAs} y_{GAs_b}(t) - \lambda_{GAs}^2 y_{GAs_a}(t) \\
\dot{y}_{N_a}(t) &= y_{N_b}(t) \\
\dot{y}_{N_b}(t) &= H_P \lambda_P p(t) - 2\lambda_P y_{N_b}(t) - \lambda_N^2 y_{N_a}(t)
\end{aligned} \tag{5}$$

The PSPs equations are:

$$\begin{aligned}
PSP_P(t) &= y_{N_a}(t) + y_{P_a}(t) - y_{P_c}(t) = LFP(t) \\
PSP_{P'}(t) &= y_{P'_a}(t) \\
PSP_{GAs}(t) &= y_{GAs_a}(t)
\end{aligned} \tag{6}$$

and the FRs equations are:

$$\begin{aligned}
FR_P(t) &= sigm_P(PSP_P(t)) \\
FR_{P'}(t) &= sigm_{P'}(PSP_{P'}(t)) \\
FR_{GAs}(t) &= sigm_{GAs}(PSP_{GAs}(t))
\end{aligned} \tag{7}$$

2.3.2. One 2nd order ODE per population. To derive the set of ODEs, each "pulse-to-wave" results in a 2nd order ODE according to eq. 2. However, some "pulse-to-wave" functions are redundant. This is the case for the $h_P(FR_P)$ in panels c-d in figure 2. More generally, the "pulse-to-wave" function can be placed right after the sigmoid function ("wave-to-pulse"). This results in only one 2nd order ODE per population, plus another one by external input (noise) added to the model. Therefore, the resulting set of ODEs is:

$$\begin{aligned}
\dot{y}_{P_a}(t) &= y_{P_b}(t) \\
\dot{y}_{P_b}(t) &= H_P \lambda_P sigm\left(PSP_P(t)\right) - 2\lambda_P y_{P_b}(t) - \lambda_P^2 y_{P_a}(t) \\
\dot{y}_{P'_a}(t) &= y_{P'_b}(t) \\
\dot{y}_{P'_b}(t) &= H_{P'} \lambda_{P'} sigm\left(PSP_{P'}(t)\right) - 2\lambda_{P'} y_{P'_b}(t) - \lambda_{P'}^2 y_{P'_a}(t) \\
\dot{y}_{GAs_a}(t) &= y_{GAs_b}(t) \\
\dot{y}_{GAs_b}(t) &= H_{GAs} \lambda_{GAs} sigm\left(PSP_{GAs}(t)\right) - 2\lambda_{GAs} y_{GAs_b}(t) - \lambda_{GAs}^2 y_{GAs_a}(t) \\
\dot{y}_{N_a}(t) &= y_{N_b}(t) \\
\dot{y}_{N_b}(t) &= H_P \lambda_P p(t) - 2\lambda_P y_{N_b}(t) - \lambda_N^2 y_{N_a}(t)
\end{aligned} \tag{8}$$

The PSPs equations are:

$$\begin{aligned}
PSP_P(t) &= y_{N_b}(t) + C_{P'-P} y_{P'_a}(t) - C_{GAs-P} y_{GAs_a}(t) \\
PSP_{P'}(t) &= C_{P-P'} y_{P_a}(t) \\
PSP_{GAs}(t) &= C_{P-Gas} y_{P_a}(t)
\end{aligned} \tag{9}$$

and the FRs equations are:

$$\begin{aligned}
FR_P(t) &= sigm_P(PSP_P(t)) \\
FR_{P'}(t) &= sigm_{P'}(PSP_{P'}(t)) \\
FR_{GAs}(t) &= sigm_{GAs}(PSP_{GAs}(t))
\end{aligned} \tag{10}$$

2.4. The Neural Mass Model Generator software

The source code of the neural mass model generator (NMMGenerator) software is available at `https://gitlab.com/yocmax/nmm_generator`. A Manual of the software along with a demo video (available at `https://gitlab.com/yocmax/nmm_generator/blob/master/Demo_NMM_Generator.mp4`) is provided with the software. A screenshot of the GUI main window is displayed in figure 3 where the Wendling NMM has been generated.

In the NMMGenerator software, the user builds in a user-friendly manner a graph representing the neural mass model for which equations should be computed. To do so, the user places on the graph three different items: populations, noises and links. In order to account for the difference of parameter values for the three items, the user needs to fill in particular values for each item, such as:

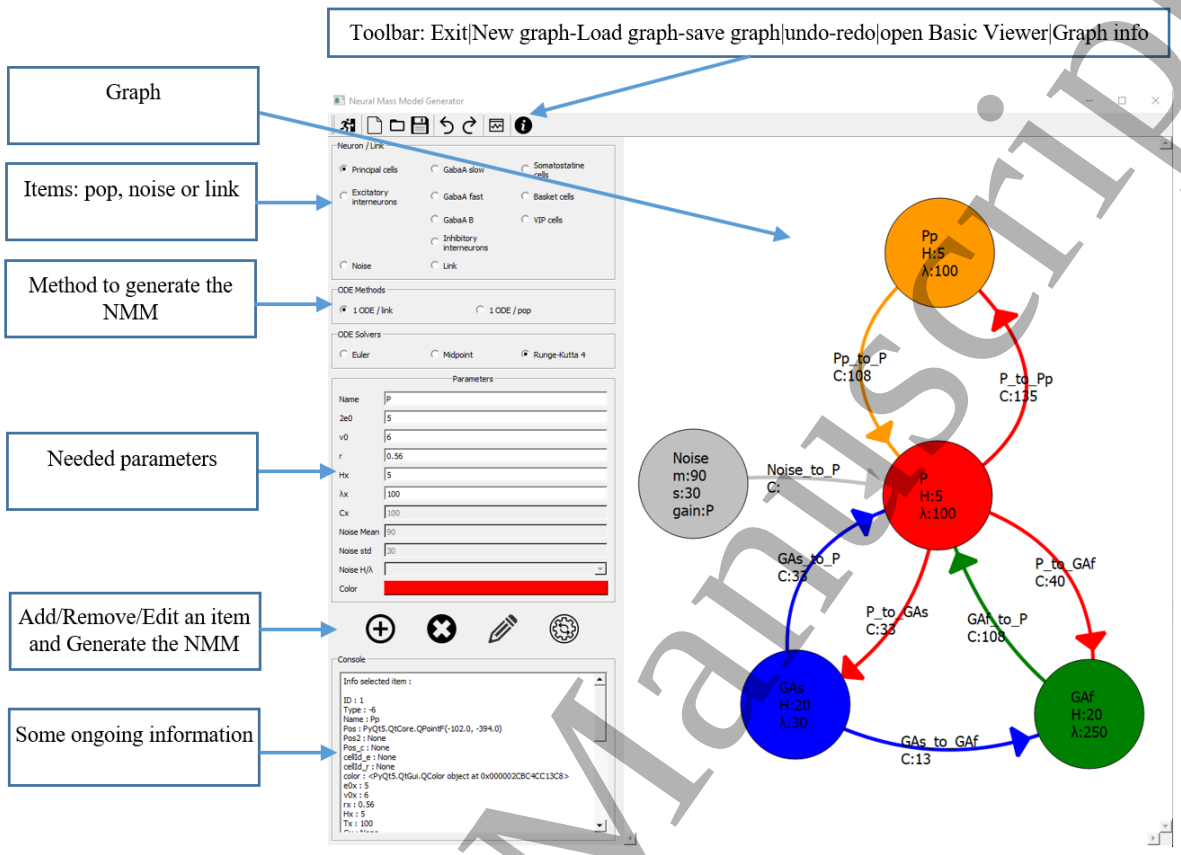

Figure 3. Screenshot of the GUI main window. The Wendling NMM is being displayed.

- a population: a name, $2e_{0n}$, v_{0n}, r_n, H_n and λ_n
- a link: $C_{na\_to\_nb}$
- a noise: a name, $mean_n$, std_n

Where n being are the names of the concerned population or noise, except for links which have na for the emitter population or noise and nb for the receiver population.

The core of the software uses a list of items that have been added to the graph. Each item used for the NMM generation is saved in that list. A class item has been created to store every variables necessary for the NMM generation. Table 1 displays variables stocked in the item class. This class has some particular variables to identify the type of item (population, noise or link). The variables cellId_e and cellId_r enable identifying the link between two populations, and also the directionality of that link ('e' stands for emitter and 'r' stands for receiver).

The items presented in the graph (populations, connections, noise) are sufficient to generate the corresponding NMM set of equations. The list of items enables knowing the number of different populations, the number of external noises as well as the number of connections linking populations and noises together. The graph created by the user can be saved under two file formats: a text file (.txt) that can be modified by hand,

Table 1. Item class, common to population, noise or link

name	info
ID	ID of the item
Type	population, noise or link
Name	Name of the item (automatic for link)
cellId_e	ID of source population or noise for a link
cellId_r	ID of target population for a link
color	color of population or noise
e0x	value for e_0 of the sigmoid function
v0x	value for v_0 of the sigmoid function
rx	value for r of the sigmoid function
Hx	value for H of the transfer function
Tx	value for T of the transfer function
Cx	value for C of the connectivity
mean	mean for the noise
std	standard deviation for the noise
Noise pop	related population for gain and time constant

and a binary file (.nmm). Those files can be loaded in the software later for further modifications.

The NMMGenerator software is provided with a NMM Basic Simulator (NMMBS) viewer GUI that provides the possibility to simulate in a user-friendly manner the generated NMM.

The NMMBS GUI allows to load a NMM class model generated with the NMMGenerator interface and to tune the various parameters corresponding with the NMM (parameters for "pulse-to-wave" transfer functions, "wave-to-pulse" sigmoid functions, connectivities and noises). The user can select the simulation duration along with the sampling frequency, and may use a Butterworth band pass filter on the result signals. In order to keep a logical link with the created graph, the color of signal plots for each population is maintained.

3. Results

In this section, two popular NMMs have been built using our software. The first one is the Jansen and Rit model [11], and the second one is the Wendling model [18]. The corresponding graphs and their NMM python classes are provided in the gitlab folder.

Let us note that the source codes provided in this paper have been slightly modified to increase readability. Since the provided source code is part of a python class, every attribute or method belonging to that class should be referred to with a `'self'` keyword, those have been removed from the paper, but do appear in the original source code.

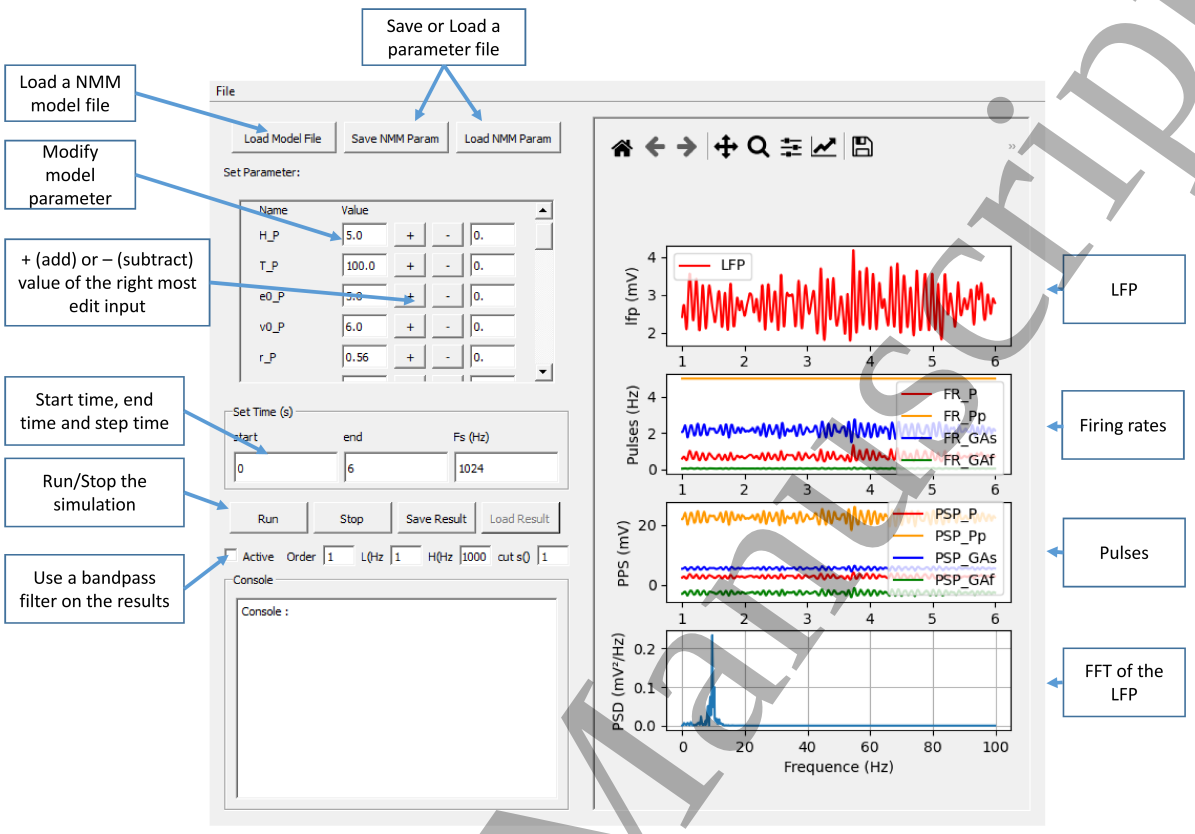

Figure 4. Screenshot of the NMM Basic Stimulator viewer GUI. Currently, the Wendling NMM corresponds to the one in figure 3.

3.1. Jansen Rit model

The graph included in the software for the Jansen and Rit NMM is presented in figure 5 (a), and corresponds to the graph from [11]. ODE equations for this graph have been given as an example in the Methods section 2.3.1 when one ODE per link is generated and in section 2.3.2 when one ODE per population is generated. The corresponding Python ODE derivative function generated by the software for the two methods are provided below.

- One ODE per link (corresponding with the ODE equation set from eq. 5):

```
dydt[0] = y[1]
dydt[1] = PTW(+C_P_to_Pp*sigm_P(+y[8]+y[2]-y[4]), y[0], y[1], H_P , T_P)
dydt[2] = y[3]
dydt[3] = PTW(+C_Pp_to_P*sigm_Pp(+y[0]), y[2], y[3], H_Pp , T_Pp)
dydt[4] = y[5]
dydt[5] = PTW(+C_GAs_to_P*sigm_GAs(+y[6]), y[4], y[5], H_GAs , T_GAs)
dydt[6] = y[7]
dydt[7] = PTW(+C_P_to_GAs*sigm_P(+y[8]+y[2]-y[4]), y[6], y[7], H_P , T_P)
dydt[8] = y[9]
dydt[9] = PTW(noise_var_N, y[8], y[9], H_P , T_P)
```

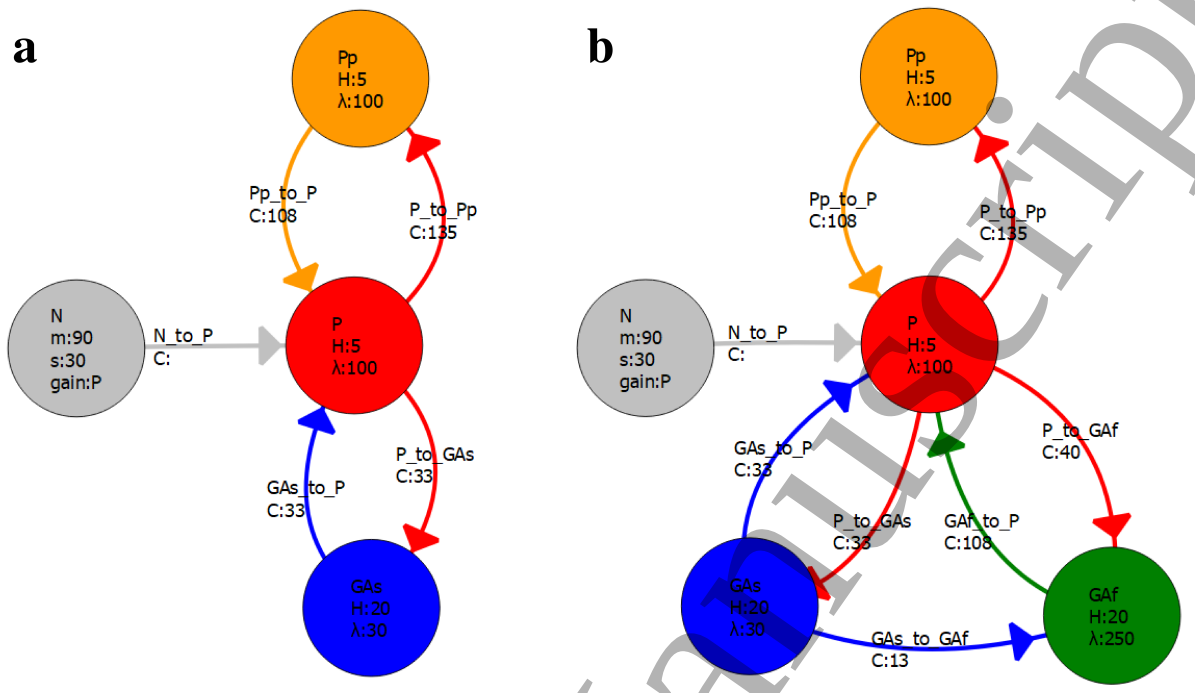

Figure 5. Two graphs made in the software: **(a)** the Jensen and Rit NMM and **(b)** the Wendling NMM.

ODE 0 and 1 correspond to the link from P to Pp, ODE 2 and 3 correspond to the link from Pp to P, ODE 4 and 5 correspond to the link from GAs to P, ODE 6 and 7 correspond to the link from P to GAs, and ODE 8 and 9 correspond to the link from N to P in Figure 5(a).

- One ODE per population (corresponding with the ODE equation set from eq. 8):

```
dydt[0] = y[1]
dydt[1] = PTW(+sigm_P(+y[6]+C_Pp_to_P*y[2]-C_GAs_to_P*y[4]), y[0],
              y[1], H_P, T_P)
dydt[2] = y[3]
dydt[3] = PTW(+sigm_Pp(+C_P_to_Pp*y[0]), y[2], y[3], H_Pp, T_Pp)
dydt[4] = y[5]
dydt[5] = PTW(+sigm_GAs(+C_P_to_GAs*y[0]), y[4], y[5], H_GAs, T_GAs)
dydt[6] = y[7]
```

ODE 0 and 1 correspond to the PSP generated by population P, ODE 2 and 3 correspond to the PSP generated by population Pp, ODE 4 and 5 correspond to the PSP generated by population GAs, and ODE 6 and 7 correspond to the PSP generated by the noise N in Figure 5(a).

Where the `PTW` function computes the second 1st-order ODE of the "pulse to wave" function as

```
def PTW(y0,y1,y2,G,T):
    return (G*T*y0 - 2*T*y2 - T*T*y1)
```

with `y0` is the input signal of the transfer function, `y1` and `y2` are the 1st order ODE states, `G` the transfer function gain, and `T` the transfer function time constant λ. For example, for the second 1st order ODE of eq. 5, the `PTW` function automatically computes $\dot{y}_{P_b}(t)$ from $C_{P'-P}$ (`C_P_to_Pp`), H_P (`H_P`) , λ_P (`T_P`) , $FR_{P'}$ (`sigm_P(y[8]+y[2]-y[4])+`) , $y_{P_b}(t)$ (`y[0]`) , $y_{P_a}(t)$ (`y[1]`).

The `Sigm_x` function computes the "wave-to-pulse" sigmoid function according to the corresponding population parameters. For example, the sigmoid function for the P population is :

```
def sigm_P(v):
    return  e0_P/(1+np.exp(r_P*(v0_P-v)))
```

3.2. Wendling model

The graph included in the software for the Wendling NMM is presented in figure 5 (b), and corresponds to the graph from [18]. The corresponding Python ODE derivative function generated by the software for the two methods are given below.

- One ODE per link: ODE equations for the Wendling graph with One ODE per link are:

$$
\begin{aligned}
\dot{y}_{P_a}(t) &= y_{P_b}(t)\\
\dot{y}_{P_b}(t) &= C_{P'-P}H_P\lambda_P FR_{P'} - 2\lambda_P y_{P_b}(t) - \lambda_P^2 y_{P_a}(t)\\
\dot{y}_{P_c}(t) &= y_{P_d}(t)\\
\dot{y}_{P_d}(t) &= C_{GAs-P}H_P\lambda_P FR_{GAs} - 2\lambda_P y_{P_d}(t) - \lambda_P^2 y_{P_c}(t)\\
\dot{y}_{P_e}(t) &= y_{P_f}(t)\\
\dot{y}_{P_f}(t) &= C_{GAf-P}H_P\lambda_P FR_{GAf} - 2\lambda_P y_{P_f}(t) - \lambda_P^2 y_{P_e}(t)\\
\dot{y}_{P'_a}(t) &= y_{P'_b}(t)\\
\dot{y}_{P'_b}(t) &= C_{P-P'}H_{P'}\lambda_{P'} FR_P - 2\lambda_{P'} y_{P'_b}(t) - \lambda_{P'}^2 y_{P'_a}(t)\\
\dot{y}_{GAs_a}(t) &= y_{GAs_b}(t)\\
\dot{y}_{GAs_b}(t) &= C_{P-GAs}H_{GAs}\lambda_{GAs} FR_P - 2\lambda_{GAs} y_{GAs_b}(t) - \lambda_{GAs}^2 y_{GAs_a}(t)\\
\dot{y}_{GAf_a}(t) &= y_{GAf_b}(t)\\
\dot{y}_{GAf_b}(t) &= C_{P-GAf}H_{GAf}\lambda_{GAf} FR_P - 2\lambda_{GAf} y_{GAf_b}(t) - \lambda_{GAf}^2 y_{GAf_a}(t)\\
\dot{y}_{GAf_c}(t) &= y_{GAf_d}(t)\\
\dot{y}_{GAf_d}(t) &= C_{Gas-GAf}H_{GAf}\lambda_{GAf} FR_{Gas} - 2\lambda_{GAf} y_{GAf_d}(t) - \lambda_{GAf}^2 y_{GAf_c}(t)\\
\dot{y}_{N_a}(t) &= y_{N_b}(t)\\
\dot{y}_{N_b}(t) &= H_P\lambda_P p(t) - 2\lambda_P y_{N_b}(t) - \lambda_N^2 y_{N_a}(t)
\end{aligned}
\tag{11}
$$

The PSPs equations are:

$$\begin{aligned} PSP_P(t) &= y_{N_a}(t) + y_{P_a}(t) - y_{P_c}(t) - y_{P_e}(t) = LFP(t) \\ PSP_{P'}(t) &= y_{P'_a}(t) \\ PSP_{GAs}(t) &= y_{GAs_a}(t) \\ PSP_{GAf}(t) &= y_{GAf_a}(t) - y_{GAf_c}(t) \end{aligned} \tag{12}$$

and the FRs equations are:

$$\begin{aligned} FR_P(t) &= sigm_P(PSP_P(t)) \\ FR_{P'}(t) &= sigm_{P'}(PSP_{P'}(t)) \\ FR_{GAs}(t) &= sigm_{GAs}(PSP_{GAs}(t)) \\ FR_{GAf}(t) &= sigm_{GAf}(PSP_{GAf}(t)) \end{aligned} \tag{13}$$

```
dydt[0] = y[1]
dydt[1] = PTW(+C_P_to_Pp*sigm_P(+y[14]+y[2]-y[4]-y[10]), y[0], y[1], H_P, T_P)
dydt[2] = y[3]
dydt[3] = PTW(+C_Pp_to_P*sigm_Pp(+y[0]), y[2], y[3], H_Pp, T_Pp)
dydt[4] = y[5]
dydt[5] = PTW(+C_GAs_to_P*sigm_GAs(+y[6]), y[4], y[5], H_GAs, T_GAs)
dydt[6] = y[7]
dydt[7] = PTW(+C_P_to_GAs*sigm_P(+y[14]+y[2]-y[4]-y[10]), y[6], y[7], H_P, T_P)
dydt[8] = y[9]
dydt[9] = PTW(+C_GAs_to_GAf*sigm_GAs(+y[6]), y[8], y[9], H_GAs, T_GAs)
dydt[10] = y[11]
dydt[11] = PTW(+C_GAf_to_P*sigm_GAf(-y[8]+y[12]), y[10], y[11], H_GAf, T_GAf)
dydt[12] = y[13]
dydt[13] = PTW(+ C_P_to_GAf* sigm_P(+y[14]+y[2]-y[4]-y[10]), y[12],
               y[13], H_P, T_P)
dydt[14] = y[15]
dydt[15] = PTW(noise_var_N, y[14], y[15], H_P, T_P)
```

ODE 0 and 1 correspond to the link from P to Pp, ODE 2 and 3 correspond to the link from Pp to P, ODE 4 and 5 correspond to the link from GAs to P, ODE 6 and 7 correspond to the link from P to GAs, ODE 8 and 9 correspond to the link from GAs to GAf, ODE 10 and 11 correspond to the link from GAf to P, ODE 12 and 13 correspond to the link from P to GAf, and ODE 14 and 15 correspond to the link from N to P in Figure 5(b).

- One ODE per population: ODE equations for the Wendling graph with One ODE

per population are:

$$\begin{aligned}
\dot{y}_{P_a}(t) &= y_{P_b}(t) \\
\dot{y}_{P_b}(t) &= H_P \lambda_P sigm\left(PSP_P(t)\right) - 2\lambda_P y_{P_b}(t) - \lambda_P^2 y_{P_a}(t) \\
\dot{y}_{P'_a}(t) &= y_{P'_b}(t) \\
\dot{y}_{P'_b}(t) &= H_{P'} \lambda_{P'} sigm\left(PSP_{P'}(t)\right) - 2\lambda_{P'} y_{P'_b}(t) - \lambda_{P'}^2 y_{P'_a}(t) \\
\dot{y}_{GAs_a}(t) &= y_{GAs_b}(t) \\
\dot{y}_{GAs_b}(t) &= H_{GAs} \lambda_{GAs} sigm\left(PSP_{GAs}(t)\right) - 2\lambda_{GAs} y_{GAs_b}(t) - \lambda_{GAs}^2 y_{GAs_a}(t) \\
\dot{y}_{GAf_a}(t) &= y_{GAf_b}(t) \\
\dot{y}_{GAf_b}(t) &= H_{GAf} \lambda_{GAf} sigm\left(PSP_{GAf}(t)\right) - 2\lambda_{GAf} y_{GAf_b}(t) - \lambda_{GAf}^2 y_{GAf_a}(t) \\
\dot{y}_{N_a}(t) &= y_{N_b}(t) \\
\dot{y}_{N_b}(t) &= H_P \lambda_P p(t) - 2\lambda_P y_{N_b}(t) - \lambda_N^2 y_{N_a}(t)
\end{aligned} \tag{14}$$

The PSPs equations are:

$$\begin{aligned}
PSP_P(t) &= y_{N_b}(t) + C_{P'-P} y_{P'_a}(t) - C_{GAs-P} y_{GAs_a}(t) - C_{GAf-P} y_{GAf_a}(t) \\
PSP_{P'}(t) &= C_{P-P'} y_{Pa}(t) \\
PSP_{GAs}(t) &= C_{P-Gas} y_{P_a}(t) \\
PSP_{GAf}(t) &= C_{P-Gaf} y_{P_a}(t) - C_{Gas-Gaf} y_{GAs_a}(t)
\end{aligned} \tag{15}$$

and the FRs equations are:

$$\begin{aligned}
FR_P(t) &= sigm_P(PSP_P(t)) \\
FR_{P'}(t) &= sigm_{P'}(PSP_{P'}(t)) \\
FR_{GAs}(t) &= sigm_{GAs}(PSP_{GAs}(t)) \\
FR_{GAf}(t) &= sigm_{GAf}(PSP_{GAf}(t))
\end{aligned} \tag{16}$$

```
dydt[0] = y[1]
dydt[1] = PTW(+sigm_P(+y[8]+C_Pp_to_P*y[2]-C_GAs_to_P*y[4]-
          C_GAf_to_P*y[6]), y[0], y[1], H_P, T_P)
dydt[2] = y[3]
dydt[3] = PTW(+sigm_Pp(+C_P_to_Pp*y[0]), y[2], y[3], H_Pp, T_Pp)
dydt[4] = y[5]
dydt[5] = PTW(+sigm_GAs(+C_P_to_GAs*y[0]), y[4], y[5], H_GAs, T_GAs)
dydt[6] = y[7]
dydt[7] = PTW(+sigm_GAf(-C_GAs_to_GAf*y[4]+C_P_to_GAf*y[0]), y[6],
            y[7], H_GAf, T_GAf)
dydt[8] = y[9]
dydt[9] = PTW(noise_var_N, y[8], y[9], H_P, T_P)
```

ODE 0 and 1 correspond to the PSP generated by population P, ODE 2 and 3 correspond to the PSP generated by population Pp, ODE 4 and 5 correspond to the PSP generated by population GAs, ODE 6 and 7 correspond to the PSP generated by population GAf, and ODE 8 and 9 correspond to the PSP generated by the noise N in Figure 5(b).

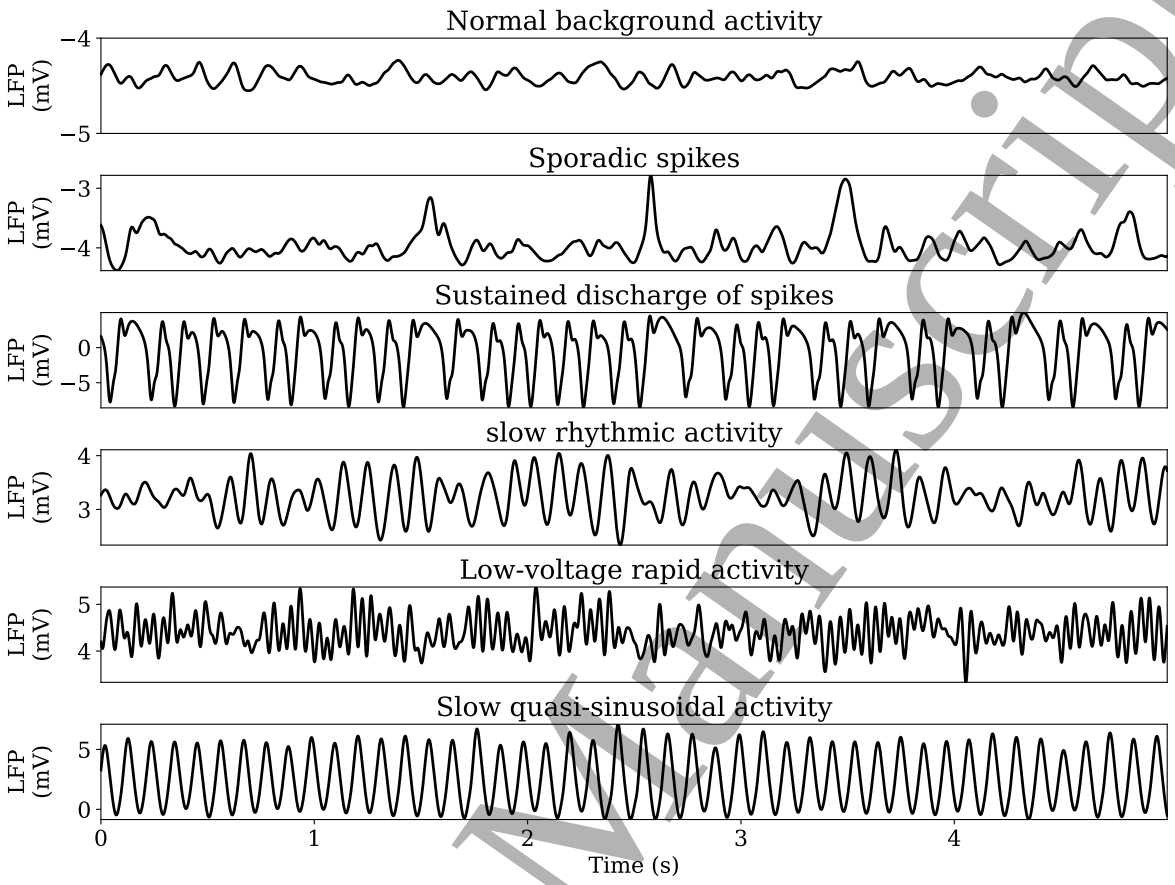

Figure 6. Results obtained with the Wendling NMM generated using our software. The 6 typical activity patterns are represented, and reproduce faithfully the patterns of the original Wendling NMM that can be found in Figure 3 in [18].

As an example, Figure 6 displays results of the Wendling NMM generated from the graph in Figure 5(b). Six qualitatively different activity patterns are presented from top to bottom: normal background activity, sporadic spikes, sustained discharge of spikes, slow rhythmic activity, low-voltage rapid activity and slow quasi-sinusoidal activity. Those can be compared with the use of the original model as in Fig 3 of the original paper [18], confirming that the model automatically generated from the graph created using our software accurately matches the output of the Wendling NMM.

4. Discussion and concluding remarks

Here, we have described a software that enables a user to automatically build a python class model of a NMM, based on a graph generated using a user-friendly interface. The graph is built out from items that can either be neural populations or links between neural populations. A neural population can represent either a type of neuron (e.g., pyramidal or basket cells), or the type of neurotransmitter used by the neuronal population (e.g., glutamatergic or GABAergic). To our knowledge, this software is the first of its kind to translate automatically a neural network model, constructed under the

form of a graph, directly into a set of differential equations that can be solved directly within the same software environment. The software is proposed for free and open-source, through a GitLab link where the source code can be found along with a software manual and a demonstration video where the Wendling NMM is built. In addition, save files of the Jansen and Rit NMM and the Wendling NMM are also provided along with the software. A dedicated part of the software enables running simulations of the generated NMM, where NMM parameters can be modified and result signals (LFPs, post-synaptic potentials and firing rates) can be displayed.

Among the limitations of our approach, let us cite that this software is only adapted for running simulations at the mesoscopic level, since the basic level of description is the neural population level. Therefore, running cellular-scale simulations cannot be performed using our framework, but could be developed using a similar methodology. Another possible issue is the tuning of the connectivity parameters within a model: while the post-synaptic response and firing rate functions are well defined and constrained based on neurophysiology, exploring the parameter space of connectivity parameters can be challenging. One possible strategy consists in initiating the network with connectivity values on the order of magnitude of those in the Jansen and Rit or Wendling model, before varying specific connectivity parameters.

Since the software is provided on GitLab, issues can be reported to improve the software with new features or bug corrections. This software can be an appropriate tool for scientists who are not necessarily familiar with mathematical or computer coding manipulations, or as a tutorial tool to present the neural mass approach to neuroscientists. Furthermore, it enables testing rapidly and in a simple manner various graphs with different populations and types of connectivity architectures, and then testing hypotheses on possible pathophysiological mechanisms that would impact neural dynamics at the population level.

Acknowledgments

This work is supported by NIH. Application Number: R01 NS092760-01A1.

References

[1] Melissa Zavaglia, Laura Astolfi, Fabio Babiloni, and Mauro Ursino. A neural mass model for the simulation of cortical activity estimated from high resolution eeg during cognitive or motor tasks. *Journal of neuroscience methods*, 157(2):317–329, 2006.

[2] Mauro Ursino, Filippo Cona, and Melissa Zavaglia. The generation of rhythms within a cortical region: analysis of a neural mass model. *NeuroImage*, 52(3):1080–1094, 2010.

[3] Peng Wang and Thomas R Knösche. A realistic neural mass model of the cortex with laminar-specific connections and synaptic plasticity–evaluation with auditory habituation. *PloS one*, 8(10):e77876, 2013.

[4] Basabdatta Sen Bhattacharya, Damien Coyle, and Liam P Maguire. A thalamo–cortico–thalamic neural mass model to study alpha rhythms in alzheimer's disease. *Neural networks*, 24(6):631–645, 2011.

[5] Marc Goodfellow, Kaspar Schindler, and Gerold Baier. Self-organised transients in a neural mass model of epileptogenic tissue dynamics. *NeuroImage*, 59(3):2644–2660, 2012.

[6] Alejo J Nevado-Holgado, Frank Marten, Mark P Richardson, and John R Terry. Characterising the dynamics of eeg waveforms as the path through parameter space of a neural mass model: application to epilepsy seizure evolution. *Neuroimage*, 59(3):2374–2392, 2012.

[7] FH Lopes da Silva, A Hoeks, H Smits, and LH Zetterberg. Model of brain rhythmic activity. *Biological Cybernetics*, 15(1):27–37, 1974.

[8] FH Lopes Da Silva, A Van Rotterdam, P Barts, E Van Heusden, and W Burr. Models of neuronal populations: the basic mechanisms of rhythmicity. In *Progress in brain research*, volume 45, pages 281–308. Elsevier, 1976.

[9] 3A van Rotterdam, FH Lopes Da Silva, J Van den Ende, MA Viergever, and AJ Hermans. A model of the spatial-temporal characteristics of the alpha rhythm. *Bulletin of mathematical biology*, 44(2):283–305, 1982.

[10] Ben H Jansen, George Zouridakis, and Michael E Brandt. A neurophysiologically-based mathematical model of flash visual evoked potentials. *Biological cybernetics*, 68(3):275–283, 1993.

[11] Ben H Jansen and Vincent G Rit. Electroencephalogram and visual evoked potential generation in a mathematical model of coupled cortical columns. *Biological cybernetics*, 73(4):357–366, 1995.

[12] Fabrice Wendling, Jean-Jacques Bellanger, Fabrice Bartolomei, and Patrick Chauvel. Relevance of nonlinear lumped-parameter models in the analysis of depth-eeg epileptic signals. *Biological cybernetics*, 83(4):367–378, 2000.

[13] Olivier David and Karl J Friston. A neural mass model for meg/eeg: coupling and neuronal dynamics. *NeuroImage*, 20(3):1743–1755, 2003.

[14] Faten Mina, Pascal Benquet, Anca Pasnicu, Arnaud Biraben, and Fabrice Wendling. Modulation of epileptic activity by deep brain stimulation: a model-based study of frequency-dependent effects. *Frontiers in computational neuroscience*, 7:94, 2013.

[15] Michael Schellenberger Costa, Jan Born, Jens Christian Claussen, and Thomas Martinetz. Modeling the effect of sleep regulation on a neural mass model. *Journal of computational neuroscience*, 41(1):15–28, 2016.

[16] Frank H Eeckman and Walter J Freeman. Asymmetric sigmoid non-linearity in the rat olfactory system. *Brain Research*, 557(1-2):13–21, 1991.

[17] Mojtaba Chehelcheraghi, Cees van Leeuwen, Erik Steur, and Chie Nakatani. A neural mass model of cross frequency coupling. *PloS one*, 12(4), 2017.

[18] F Wendling, F Bartolomei, JJ Bellanger, and P Chauvel. Epileptic fast activity can be explained by a model of impaired gabaergic dendritic inhibition. *European Journal of Neuroscience*, 15(9):1499–1508, 2002.