Mean Field Conditions explained by General Linear System Morphisms: Computational Approximation and Application to Neuronal Pool Connections

Alexandre Muzy, Bernard P Zeigler

Mean Field Conditions explained by General Linear System Morphisms: Computational Approximation and Application to Neuronal Pool Connections

Alexandre Muzy*, Bernard P. Zeigler†

March 27, 2020

1 Introduction

In computational neuroscience, based on methods from statistical mechanics, mean field approach is used widely to abstract analytically a large pool of neurons into a single behaviorally equivalent unit that can be employed as a component in larger scale networks (Faugeras, Touboul, and Cessac 2009; Nykamp et al. 2017; Ostojic 2014; El Boustani and Destexhe 2009). Considering homogeneous behaviors (all neurons have the same state and output behaviors) and all-to-all or randomly uniform couplings between neurons, particular equations of neurons are derived while preserving the average firing rate of the pool. Usually, the connections between the neurons are taken to be inversely proportional to an infinite number of neurons leading to a weak coupling between neurons. The results obtained are specific to the derived neuronal equations derived. Both structure and dynamics of the pool of neurons constitute a rough abstraction of the complexity of an actual neuronal pool but this abstraction is worth to infer more knowledge about the global behavior resulting from the interactions between neurons.

On the other hand, linear systems (Zadeh and Desoer 1963) constitute a general analytical tool. Based on the linear properties of system dynamics, the set of parameters of the corresponding equations is usually studied through phase diagrams and in/stability of the dynamics. Including non/linear systems, general system theory (Arbib 1972; Klir 1985; Mesarovic and Takahara 1989; Mesarovic and Takahara 1975; Wymore 1967; Arnold 1994; Harrison 1969; Ho 1992) has been developed to reason very generally over abstract states and system dynamics. This abstraction level is right for manipulating and reasoning

*Université Côte d'Azur, I3S CNRS, France, Email: alexandre.muzy@cnrs.fr.

†Chief Scientist, RTSync Corp, 530 Bartow Drive Suite A Sierra Vista, AZ 85635, United-States of America.

over system structures and behaviors. The intertwined structures and dynamics can be studied analytically to infer general properties thus providing more knowledge on the systems before simulating them.

This article presents the implementation of an abstraction of linear systems using system morphism representations and mean field conditions. Computational modeling is done using a system specification formalism (Zeigler, Muzy, and Kofman 2018). Base networks of linear systems are abstracted into lumped networks. The lumping is detailed based on the base network. Mean field conditions ensure the preservation of the average activity in the network. Also, the coupling of networks is studied to be able to construct networks of networks while still usefully preserving the dynamics of the whole system. *Finally, the mathematical framework proposed allows connecting computational systems modularly through their input/output interfaces following an engineering approach.* It allows differentiating between the convergence of the dynamics of networks of systems and the computational error introduced. Although usual mean field conditions lead to zero error in the dynamics, these conditions can be relaxed (with a finite number of systems and non-uniform couplings among the systems) identifying the frontier between analytical analysis and the necessity of simulation to better understand the dynamics of the overall network. Finally, all the results obtained are discussed in the context of neuronal network and a linear version of the Wilson–Cowan model (Wilson and Cowan 1973). "Rather than focus on the microscopic properties of neurons, Wilson and Cowan analyzed the collective properties of large numbers of neurons" (Destexhe and Sejnowski 2009).

In Section 2, mathematical system and mean field theories are introduced. In Section 3, the mathematical framework of linear time invariant systems with inputs/outputs is defined. In Section 4, the mean field abstraction is clearly defined for linear systems using computational morphisms. Section 6 presents the abstraction of network dynamics based on the connections in the network. Section 8 discusses the results obtained for linear systems in the context of neuron models. Section 8 discusses the computational error approximation achieved at each network transition in the context of approximate morphisms and brain simulation. Finally, in Section 10, conclusions are provided.

2 Mathematical general system and mean field theories

2.1 Mathematical system theory

Mathematical general systems consist of state-based systems with inputs and outputs. These systems can be linear or non-linear, with few hypotheses about their structure (e.g., as time invariance described in the mathematical framework section), making these structures very abstract. Input/Output (I/O) interactions of systems make them very realistic but require adding particular mathematical properties to derive theorems about the expected behavior of these

publication	non linear state correlation between neurons	all-to-all coupling btwn networks	many-to-many coupling btwn networks	infinite nb of neurons	non infinite nb of neurons	weights inv. prop. to nb of neurons	arbitrary degree distribution in a network
(Faugeras, Touboul, and Cessac 2009)	x	x		x		x	
(Nykamp et al. 2017)	x			x			x
(El Boustani and Destexhe 2009)				x			x
(Ostojic 2014)				x			x
us		x	x	x	x		x

Table 1: Mean field conditions in neural networks.

systems. In the theory of modeling and simulation (Zeigler, Muzy, and Kofman 2018), a computational specification of general systems has been proposed. The computational systems considered here consist of linear systems.

2.2 Mean field theory

Many references using the mean field hypotheses in the context of neural networks could be cited here. In Table 1, we focus on the main usual hypotheses and breakthrough results with respect to neural network structures (Nykamp et al. 2017; El Boustani and Destexhe 2009; Ostojic 2014) (allowing neuronal networks with arbitrary random degree distributions) and behavior (Faugeras, Touboul, and Cessac 2009) (exploring the state correlation between neurons). For mathematical convergence to a fixed point, usual hypotheses consist of all-to-all couplings between networks, an infinite number of neurons, weights inversely proportional to the number of neurons. We will show here that the all-to-all couplings between networks is *sufficient* but not *necessary* and that a special way of generating couplings (based on a permutation between the influencing components sampled by the influenced components) is *sufficient and necessary*. The same way, we will show that an infinite number of neurons in a network is sufficient but not necessary and that a finite number of neurons is possible.

3 Mathematical framework

3.1 I/O general state-based systems

Definition 1. A *deterministic general I/O system is a structure* (cf. behavior introduced in Figure 1)

$$SYS = (\delta, \lambda)$$

Where

$\delta : Q \times \Omega \to Q$ is the *transition function*, with Q the *set of states*, Ω the *set of (piecewise continuous) input segments* $\omega :< t, t_2 >\to X$[1], with $< t_1, t_2 >$ the *interval of the segment*[2], and X the *set of input values*. The sets of input values X and states Q are arbitrary.

$\lambda : Q \to Y$ is the *output function*, which can be considered as a (partial) observation of the state of the system.

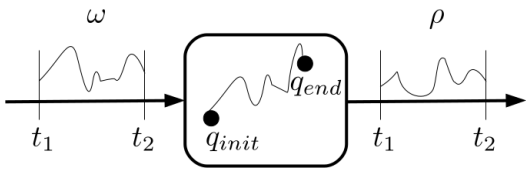

Figure 1: General I/O system dynamics: When receiving an input segment $\omega \in \Omega$, the system achieves a transition from initial state $q_{init} \in Q$ to final state $q_{end} \in Q$ and generates an output segment $\rho \in$ P.

For one input segment $\omega \in \Omega$ defined over an interval $< t_1, t_2 >$, with t_1 and t_2 not fixed[3], the system goes continuously from one initial state $q_{init} \in Q$ to one final state $q_{end} \in Q$ by its transition function: $q_{end} = \delta(q_{init}, \omega)$. To do so, intermediate states are computed for particular (allowed) time breakpoints $t \in< t_1, t_2 >$ based on the *composition property of the transition function* (cf. Figure 2): $\delta(q, \omega) = \delta(\delta(q, \omega_{t>}), \omega_{<t})$, with $\omega_{t>} = \omega|_{<t_1,t>}$ and $\omega_{t>} = \omega|_{<t,t_2>}$ being respectively the *left sub-segment* and the *right sub-segment of* ω. Finally, the system generates an *output segment* $\rho \in$ P such that $\rho :< t_1, t_2 >\to Y$ and $\rho_{t>} = \lambda(\delta(q, \omega_{t>}))$. The *set of input segments*, Ω, is the union of all input segments $\omega_{t>}$ and $\omega_{<t}$ and the *set of output segments*, P, is the union of all output segments $\rho_{t>}$ and $\rho_{<t}$.

[1] A piecewise continuous input segment is a map from each time point $t \in< t_1, t_2 >$ (with t_1 and t_2 not fixed) to a corresponding input value $x \in X$.

[2] Signs '<' and '>' correspond either to a square brackets '[' or a square bracket ']'.

[3] Segments can be also defined as starting from time 0 showing then that they can be translated, this is the *time invariance property of systems* (Zeigler, Muzy, and Kofman 2018).

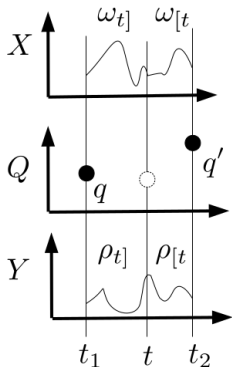

Figure 2: Composition of segments.

The *current state* is the minimal information to deterministically compute the *next state* in a very large state space. The system is markovian. However, notice that a current state can be seen as the result of previous input-state transitions (Zadeh and Desoer 1963). Then, the state of the system can be considered at a higher dependence order, a state being the result of several previous state transitions. Notice also that the system holds inputs and outputs, which is a more general and convenient principle for modeling complex systems, although it makes these systems more unpredictable. (Ivanov 2013) proves also that previous inputs can be stored in states showing the equivalence of both closed and open system structures.

Systems are very abstract and general structures that proved to map all usual modeling formalisms (Zeigler, Muzy, and Kofman 2018). They allow integrating and comparing these formalisms. However, abstract does not mean trivial in the sense that the properties shown for arbitrary inputs, states and outputs can be shown to hold at a lower specification level, i.e., for specific inputs, states and outputs.

Systems can be *time invariant*, i.e., any input segment $\omega :< t_1, t_2 >\rightarrow X$, applied at time t_1 can be applied at a time t_3, leading to the same state and output transitions. Defining a translation operator for each time $t \in T$, as $TRANS_\tau : \Omega \rightarrow \Omega$, for an input segment ω, $\omega' = TRANS(\omega)$, with $\omega'(t+\tau) = \omega(t)$ for all $t \in< t_1, t_2 >$. Then, a system $SYS = (\delta, \lambda)$ is time invariant for all input segments $\omega \in \Omega$ and all times $\tau \in T$, if:

1. Ω is *closed under translation*: for $\omega \in \Omega \Rightarrow TRANS_\tau(\omega) \in \Omega$.
2. δ is time invariant: for all states $q \in Q$, $\delta(q, \omega) = \delta(q, TRANS_\tau(\omega))$.

3.2 Linear time invariant systems

Definition 2. A Linear time invariant System (LSYS) is a structure

$$LSYS = (\delta, \lambda)$$

Where

X, Q, Y are dimensional vector spaces over $\mathbb{R}$,
$\delta(q,\omega) = qe^{Al(\omega)} + \int_0^{l(\omega)} e^{A(l(\omega)-\tau)}B\omega(\tau)d\tau$ is the transition function[4],
$\lambda(q) = CQ$ is the output function,
$A : Q \to Q$, $B : X \to Q$ and $C : Q \to Y$ are linear operators.

Definition 3. A matrix representation $\begin{cases} \frac{dq(t)}{dt} = Aq(t) + Bx(t) \\ y(t) = Cq(t) \end{cases}$ is represented by a Linear Time Invariant System as $\begin{cases} \delta(q,\omega) = qe^{At} + \int_0^t e^{A(t-\tau)}B\omega_{\tau>}d\tau \\ \lambda(q) = CQ \end{cases}$.

Definition 4. A linear time invariant system consists of linear transition and output functions with additive and distributive properties:

1. $\delta(q_1 + q_2, \omega_{1,t>} + \omega_{2,t>}) = \delta(q_1, \omega_{1,t>}) + \delta(q_2, \omega_{2,t>})$
2. $\delta(aq, a\omega_{t>}) = a\delta(q, \omega_{t>})$
3. $\lambda(\delta(q_1 + q_2, \omega_{1,t>} + \omega_{2,t>})) = \lambda(\delta(q_1, \omega_{1,t>})) + \lambda(\delta(q_2, \omega_{2,t>}))$

Let us prove these properties.

Proposition 1. $\delta(q_1 + q_2, \omega_{1,t>} + \omega_{2,t>}) = \delta(q_1, \omega_{1,t>}) + \delta(q_2, \omega_{2,t>})$

Proof. Based on matrix representation,

$$\begin{aligned} \delta(q_1 + q_2, \omega_{1,t>} + \omega_{2,t>}) &= e^{At}q_1 + \int e^{A(t-\tau)}B\omega_{1,\tau>}d\tau + e^{At}q_2 + \int e^{A(t-\tau)}B\omega_{2,\tau>}d\tau \\ &= e^{At}q_1 + e^{At}q_2 + \int e^{A(t-\tau)}B\omega_{1,\tau>}d\tau + \int e^{A(t-\tau)}B\omega_{2,\tau>}d\tau \\ &= e^{At}(q_1 + q_2) + \int e^{A(t-\tau)}B(\omega_{1,\tau>} + \omega_{2,\tau>})d\tau \\ &= \delta(q_1, \omega_{1,t>}) + \delta(q_2, \omega_{2,t>}) \end{aligned}$$

□

Proposition 2. $\delta(aq, a\omega_{t>}) = a\delta(q, \omega_{t>})$

Proof. Similar to Proposition 1. □

Proposition 3. $\lambda(\delta(q_1 + q_2, \omega_{1,t>} + \omega_{2,t>})) = \lambda(\delta(q_1, \omega_{1,t>})) + \lambda(\delta(q_2, \omega_{2,t>}))$

Proof. Starting from additive properties,

$$\begin{aligned} \lambda(\delta(q_1 + q_2, \omega_{1,t>} + \omega_{2,t>})) &= \lambda(\delta(q_1, \omega_{1,t>}) + \delta(q_2, \omega_{2,t>})) && \text{by } Proposition\,1 \\ &= C(\delta(q_1, \omega_{1,t>}) + \delta(q_2, \omega_{2,t>})) && \text{by } Definition\,2 \\ &= C\delta(q_1, \omega_{1,t>}) + C\delta(q_2, \omega_{2,t>}) && \text{by linearity of } C \\ &= \lambda(\delta(q_1, \omega_{1,t>})) + \lambda(\delta(q_2, \omega_{2,t>})) \end{aligned}$$

□

In the sequel we will use linear time invariant systems called linear systems for short.

[4] Notice that all segments are translated to 0, for simplicity.

3.3 General system morphisms

I/O general systems can be considered as abstract machines achieving *temporal computations (or executions of system (output) transitions functions*). A temporal computation relies on a delay (possibly zero) between inputs and outputs. Computations take time. The number of the computations should be finite to guarantee that the simulation ends.

Simulation and computers consist more and more of a huge number of components interacting together. A fundamental modeling challenge remains the development of a guiding mathematical framework to constructively set and analyze the behavior of networks of components at both local and global levels. The difficulty of developing such modeling structures is due to the number of local state computations and to the interactions between the components (the temporal coordination of the distributed computations). To abstract local system behaviors into network ones, relying on local (temporal) state computations, *system morphisms* can be used.

Definition 5. A *system morphism* or generalized homomorphism[5], between a detailed system SYS (or base system) and another abstract system SYS' (or lumped system), is a pair (g, h) such that (cf. Figure 3):

1. $g : \Omega \to \Omega'$, is the input mapping,
2. $h : \overline{Q} \to^{onto} Q'$, where $\overline{Q} \subseteq Q'$, is the state mapping,
3. for all $q \in \overline{Q}$, $\omega' \in \Omega'$,$h(\delta(q,\omega)) = \delta'(h(q), g(\omega))$ (transition function preservation)
4. $k : P \to P'$, is the output mapping.

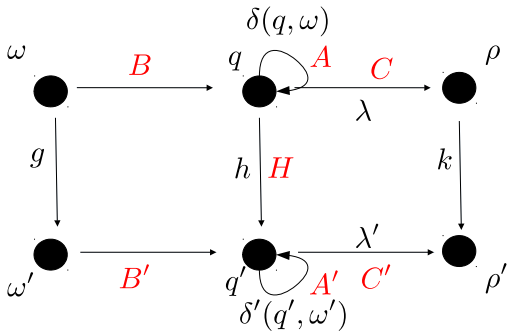

Figure 3: Commutative diagram of morphism mappings from a small system SYS to a big system SYS'. In red are indicated the matrix representations.

[5]We will use the term "homomorphism" in the sequel for state-to-state mapping in a system is considered and "morphism" when input/output systems are considered.

Theorem 1. *Fixed point preservation under system morphisms: If there exists a fixed point in the base system SYS, for a particular base input segment $\omega \in \Omega$, there exists a fixed point in the lumped system SYS', for the lumped input segment $\omega' \in \Omega'$ corresponding to $\omega \in \Omega$.*

Proof. Based on transition function preservation of Definition 5:

1. For all $q \in \overline{Q}$, $\omega' \in \Omega'$,$h(\delta(q, g(\omega))) = \delta'(h(q), \omega')$,
2. For input segment ω, let q^* be a fixed point of transition function δ: $\delta(q^*, \omega) = q^*$

Set $q = q^*$ in 1., $h(\delta(q^*, \omega)) = \delta'(h(q^*), g(\omega))$, then by 2., $h(q^*) = \delta'(h(q^*), \omega')$, so $h(q^*)$ is a fixed point of δ' for lumped input segment $\omega' = g(\omega)$. □

4 Network lumping based on mean field conditions

4.1 Mean field network morphism

After having informally used networks of linear systems let's define more formally such networks. Figure 4 shows the morphism between a base network and a lumped network.

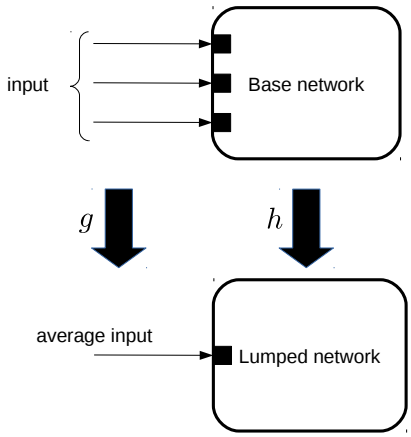

Figure 4: Morphism between base and lumped networks. An example of input average is provided here.

The morphism of networks can be achieved based on mean field conditions.

Definition 6. Usual mean field conditions can be summarized into only *two sufficient and necessary conditions* for abstracting linear networks:

1. Homogeneity: All the components of the network have the same dynamics (or state transition/output function) and the network connectivity is permutation based (to guarantee that the sum of all component inputs is a multiple of the lumped model state - no input is missing or appears many times, we will see how to define it in a moment),

2. Stability: The transition function of the network resultant system admits a fixed point, the system having output-to-input feedback.

Mean field morphism between a base network model and a lumped network model is presented in Figure 5.

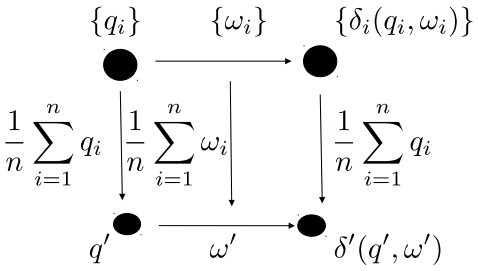

Figure 5: Commutative diagram of mean field network morphism.

4.2 Dynamic matching and network connectivity

Matching the local state transitions in the detailed network to the global state transitions in the lumped network, requires a correct sampling of the local states. As we will see, this sampling is achieved at each transition by the transitions between influencing and influenced components in the network. This is why the sampling depends on network connectivity. We will see how to ensure a full deterministic sampling by generating in a constructive way the couplings between linear systems. This sampling is based on a permutation of the global states of the network.

Example 1. Let's detail the structure lumping achieved at coupling level between the components of a network and a simple *homomorphism* between the states of base and lumped networks. Based on homogeneity condition, the transitions of a base network embedding two pools of interacting components and a lumped network with two corresponding single states is shown in Figure 6. Following this commutative diagrams, the homomorphism requirement, for all states (q_A, q_B) in $Q_A \times Q_B$ and all states $(q_{A'}, q_{B'})$ in $Q_{A'} \times Q_{B'}$, is $\delta'(h(q_A, q_B)) = h(\delta(q_A, q_B))$, for $h(q_A, q_B) = (\sum_{i=1}^{n_A} q_{A_i}, \sum_{i=1}^{n_B} q_{B_i})$, i.e., taking the sum of the states of the n_A (resp. n_B) components in pool A (resp. B). At each time, the states of A' and B', i.e., (q_A, q_B) in $Q_{A'} \times Q_{B'}$ must turn out to be the same whether you compute them by:

1. First computing the base model transition and the projecting down using h, or

2. First projecting down using h and then computing the lumped model transition.

Besides, structural homogeneity requires the number senders in pool A being the same for each receiver in pool B.

Remark 1. Usual all-to-all coupling of components in the base network taken when mean field theory is applied to neural networks (Cessac 2019; Faugeras, Touboul, and Cessac 2009) is sufficient but not necessary. It can be seen that homomorphism requirement holds for either all-to-all, one-to-one, many-to-one couplings. Also, in the computational context, the number n of components needs not to be infinite.

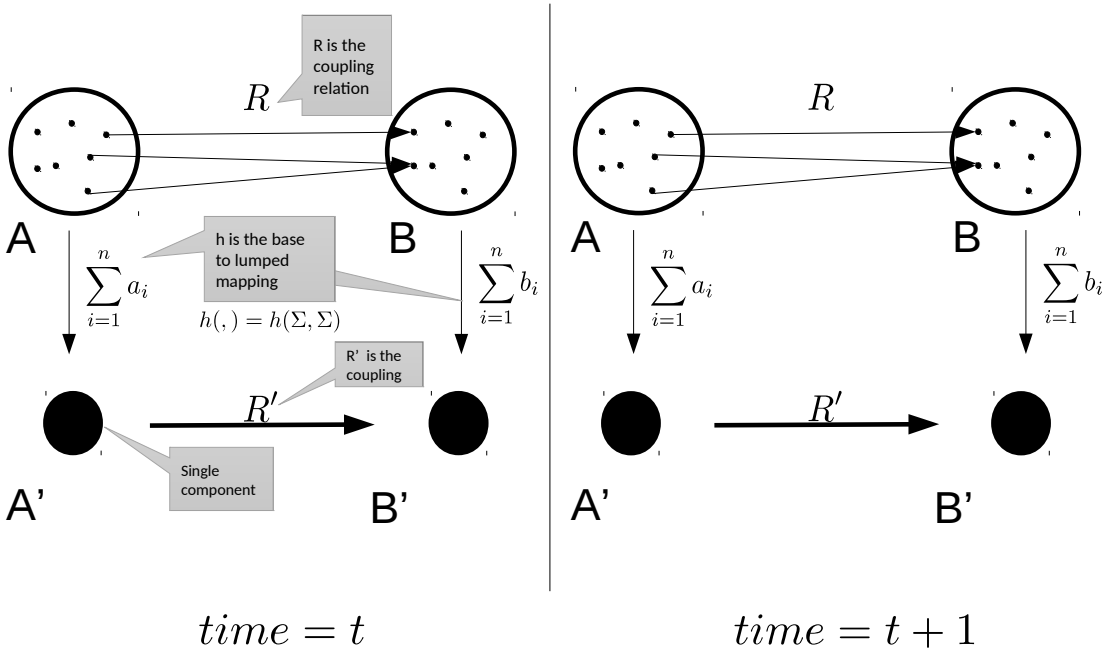

Figure 6: Base and lumped model transitions at times t and $t + 1$ based on a simple homomorphism h between a base network and a lumped network. In the base network (resp. lumped network), components in pool A (resp. A') influence components in pool B (resp. B').

Both homogeneity and stability assumptions lead, for example, to the network connectivity of Figure 7. At connectivity level, each receiver gets the same values and has the same number of senders.

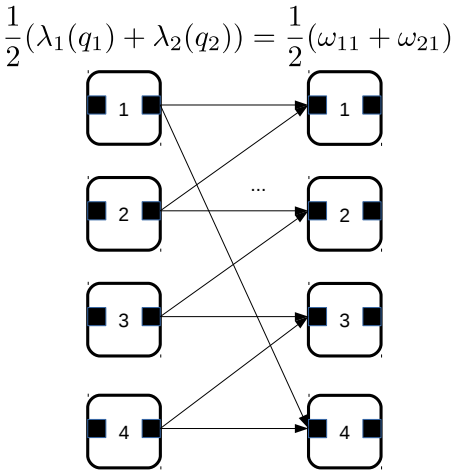

Figure 7: Example of cyclic *permutation-based* two-to-one and self internal coupling in a mean field (lumped) network. The senders are represented on the first column and the receivers on the second. For the sake of simplicity, the same component is represented twice. The homomorphism here is the mapping of component states.

Connectivity homogeneity of mean field is achieved using a permutation-based connectivity. Think of permutations as global constraints on the connectivity of the network. Both examples in Figures 8 and 9 are permutation related. In the first example, the connectivity consists of an identity mapping and a (non-cyclic) permutation mapping. In the second example, connectivity consists of an identity mapping and a cyclic permutation mapping. Permutations are one-to-one and onto maps. Think the connectivity between sender and receiver indexes like: $(0, x)(1, y)(2, z)$, where $0, 1, 2$ are the *indexes of senders*, and x, y, z are the *possible indexes of receivers*. Permutations are generated by choosing in a sequence from a set without replacing the choices - that's how the number of permutations is obtained: $n! = n*(n-1)*(n-2)*...2*1$. Here we have x that can be $1, 2, 3$; y can be $2, 3$ if we chose 1, and z can only be 2 if we chose 3. So there are $3! = 3*2*1 = 6$ possible mappings. Combinations (vs. permutations) are made if replacement is made each time we make a choice. So then we get $n^n = n*n*n....*n$ or $3*3*3 = 27$ here (all possible mappings). Not allowing replacement is equivalent to not allowing a receiver to get more than one "hit" in each map. This is saying that the resources being hit - i.e., being connected - are limited and only a limited amount is available to any node to be connected. In the finite case, this also implies that the hits must be balanced - components cannot receive more than one hit for each map- so each one receives exactly m hits for m maps. In the first example, the component 8 is hit twice - once for each map.

In a 2-D cellular space, the identity and each direction can be a map - so there are 5 maps: identity, North, South, East, West (2 directions in each

dimension). Every cell gets one input from its northern neighbor in a finite space (torroidal wrapping makes cyclic permutation); similarly, for south, east, west. Note in a N by M space, there are N cycles of size M for each direction - so cycles do not include the whole space of $N*M$ cells. A physical process could sweep from east to west and connect each cell to its nearest neighbor; similarly it could go west to east and north-south, south-north to make all connections. In the random net case, instead of locally selecting influencees at random as in a random graph, we break the process into globally selecting m maps (for m coupling directions) where each map is forced to be a permutation by resource constraints.

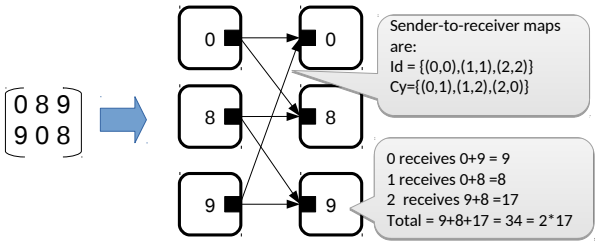

Figure 8: Connectivity example of components in a network with identity mapping Id and cyclic permutation mapping $Perm$.

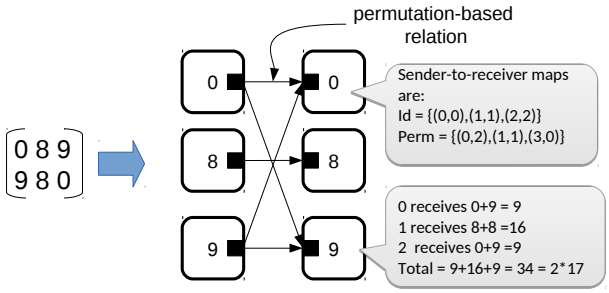

Figure 9: Connectivity example of components in a network with identity mapping Id and non-cyclic permutation mapping $Perm$.

By examples in Figures 8 and 9, it can be seen graphically that permutation (and identity) maps lead to many-to-one, one-to-one, or all-to-all couplings. The multiset of senders received by any receiver is represented by the disjoint union of maps, which leads to an onto mapping. For example, let's have two permutation maps and two components with indexes $0, 1$: $Perm = \{(0,1),(1,0)\}$ and $Id = \{(0,0),(1,1)\}$, then the disjoint union consists of $Perm \amalg Id = \{(0,1),(1,0),(0,0),(1,1)\}$, which is all-to-all coupling.

Figure 10 shows a matrix representation of the permutation-based connectivity. It can be seen that summing the rows (global state permutations) is equal

to summing the column (local state computations). This shows that whatever the permutation mappings chosen, all the states at network level are fully sampled locally by components. Finally, it can be seen that the number of inputs times the sum of component states is also equal to the sum of rows (and also of columns). Thus a morphism from base to lumped model can be based on the sum (or average) of component states .

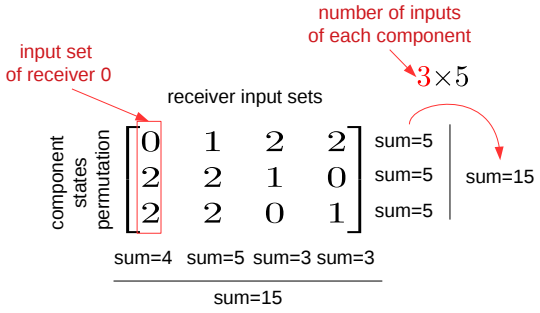

Figure 10: Matrix representation of a connectivity permutation: Each cell consists of a component state, each column of the inputs received from a sender and each row consists of the global state of the network composed of four components (s_1, s_2, s_3, s_4). The first row represents the identity mapping.

In conclusion, for deterministic couplings:

- Full sampling (permutation-based) coupling leads to no error,
- Partial sampling (or permutation deficient) coupling leads to error,
- Random couplings (e.g., Erdős-Renyi, which is uniform so good to average error): A full sampling is 100% and a deficient sampling is less %.

In deterministic couplings, we are interested in maps. To get no error all-to-all is sufficient but nut necessary. *Identity or permutation coupling maps are sufficient and necessary* (because they lead to a full sampling).

4.3 Base and lumped networks of systems

Let's define now more precisely the base and lumped model structures.

Definition 7. A *base network of systems* consists of

$$\eta = (\delta, \lambda, \{S_i\})$$

Where

$\delta(q, \omega) = \times_i \delta_i(q_i, \omega_i)$, for $i = 1, ..., n$, is the *base network transition function* based on the *component transition functions* δ_i,which depend on: (i) the *base network state*, $q = (..., q_i, ...)$, with q_i the *component states* and on

the *base network external input* $\omega = (..., \omega_i, ...)$, with ω_i the *component inputs*.

$\lambda(q) = \times_i \lambda_i(\delta_i(q_i))$ is the *base network output function* based on the *component output functions* λ_i,

$\{S_i\} = \{S_i^{ext} \cup S_i^{int}\}$ is the *set of senders to the network* (cf. Example 1) such that $S_i = \{j \mid \omega_i = \lambda_j(\delta_j(q_j, \omega_j)\}$ is the *set of sending components* j *external or internal* to the network (as pool A in Figure 6) connected to the input, $\omega_i \in \Omega_i$, of *a receiving* component i.

Definition 8. A *lumped network of systems* consist of

$$\eta' = (\delta', \lambda', \{S_i'\})$$

Where

$\delta'(q', \omega')$ is the *transition function of the network*, for all *lumped states* $q' \in \overline{Q'}$, and *lumped input* $\omega' \in \Omega'$,

$\lambda'(\delta'(q'))$ is the *output transition function of the network*.

$\{S'\}$ is the *set of senders to the network* such that $S' = \{j \mid \omega' = \lambda_j'(\delta_j'(q_j', \omega_j')\}$ is the *set of sending components* j to the network, connected to the input $\omega' \in \Omega'$.

Remark 2. In a *base network,* a *pool of states* of the components consists of *equivalent states* mapped into a single state of the lumped network. At dynamics level, for each transition, each pool of states in the base network matches a single state in corresponding lumped network. The mapping (or homomorphism) between state pools to single states can be generalized (instead of a simple average or summation) as a *census* of states in a pool, i.e., as counting the number of components in a particular state in the pool. For mean field conditions, the summation-based aggregation of the states in a pool requires permutation-based couplings between the components.

The method for modeling and lumping networks consists of the following independent steps: (1) Specify the base network (using Definition 7), (2) specify the lumped network (using Definition 8), and (3) Specify the morphism triple (g, h, k) (using Definition 5).

Notice that this method works for any I/O systems. Let's now make it specific to networks of linear systems and mean field morphisms.

4.4 Lumping a network of linear systems based on mean field morphisms

Let's consider now a base network of linear components (i.e., having the linear properties defined in Definition 4) and a lumped network *as a single linear system.*

Based on mean field assumptions (cf. Definition 6), applying the lumping method steps consists of:

1. Specify both transition and output functions of the base network of linear systems (using Definition 7) $\eta_{linear} = (\delta_{linear}, \lambda_{linear}, \{S_{linear_i}\})$: Based on dynamics homogeneity mean field conditions of Definition 6, all transition functions have the same behavior, i.e., $\delta_i \stackrel{def}{=} \delta_j$, for all components i, j of the network and all output functions have the same behavior, i.e., $\lambda_i \stackrel{def}{=} \lambda_j$, for all components i, j of the network..

2. Specify both transition and output functions of the lumped network (using Definition 8) $\eta'_{linear} = (\delta'_{linear}, \lambda'_{linear}, \{S'_{linear}\})$: Based on mean field conditions on dynamics homogeneity (cf. Definition 6), $\delta' \stackrel{def}{=} \delta_i$, where δ_i are the transition functions in the base network and $\lambda' \stackrel{def}{=} \lambda_i$, where λ_i are the output functions in the base network and considering internal coupling relation from the component outputs of the network to the output of the network (as described in Example 1)..

3. Specify the morphism (g, h) (using Definition 5 and with no output mapping for simplification): $g(..., \omega_i, ...)$ is the input mapping (a sum or average of base component states as in Example 1) with $(..., \omega_i, ...)$ the inputs of the components in the base network (external or internal to the base network), $h(..., q_i, ...)$ is the homomorphism (a sum or average of base component states as in Example 1) with q_i the *component states* in the *base network.* Finally, the lumped transition function thus consisting of $\delta'(q', \omega') = \delta'(h(\times_i q), g(\times_i \omega_i))$.

We will discuss in the next sections the properties and examples of such linear networks.

5 Mean field conditions explained by structure morphisms

5.1 Homogeneity and stability condition

When applied to a network, a system morphism is called a *structure morphism.* Let's take a simple example of a network of linear systems to expose the notion of structure morphism with respect to mean field conditions.

Example 2. A network of two 1-D linear components with identical structure (following homogeneity condition of mean field theory (cf. Definition 6)) consists of:

$$\begin{cases} q_1\text{'} = aq_1 \ + bx_1 \\ y_1 = cq_1 \end{cases} \quad \text{and} \quad \begin{cases} q_2\text{'} = aq_2 \ + bx_2 \\ y_2 = cq_2 \end{cases}$$

The stability condition of mean field theory (cf. Definition 6), based on feedback, can be represented by different homomorphisms and structures of the network as long as each component receives the same values, i.e., they have the same number of influencers of the same kind (or dynamic transition) through different feedback loops:

1. A network with feedback being the (same) average output to each component: $x_1 = x_2 = \frac{y_1 + y_2}{2} = \frac{cq_1 + cq_2}{2} = \frac{c(q_1 + q_2)}{2}$.
2. A network where components have self and independent feedback loops: $x_1 = \frac{y_1}{2}$ and $x_2 = \frac{y_2}{2}$, also leads to $x_1 = x_2 = \frac{c(q_1 + q_2)}{2}$.
3. A network where components have cross feedback loops: $x_1 = y_2$ and $x_2 = y_1$, also leads to $x_1 = x_2 = \frac{c(q_1 + q_2)}{2}$.

In all feedback loop cases, taking the homomorphism $h(q_1, q_2) = q_1 + q_2$, the state of the lumped network is given by: $q_1' + q_2' = a(q_1 + q_2) + b(x_1 + x_2) = a(q_1 + q_2) + 2bx_1 = a(q_1 + q_2) + bc(q_1 + q_2)$, thus leading to equation:

$$Q' = (a + bc)Q \tag{1}$$

With $Q = q_1 + q_2$.

In conclusion, based on different homogeneity and stability conditions, different structures lead to the same lumped network dynamics.

Remark 3. This simple example shows that the stability of the lumped network dynamics depends on the sign of parameters $a + bc$: if $a + bc > 0$, the system dynamics is exponentially growing, the system thus being unstable while for $a + bc < 0$, the system dynamics is stable.

Following previous assumptions, the relationship between the base and lumped model networks consists of:

Theorem 2. *If there exists a fixed point in the base network of linear systems, $\eta_{base} = (\delta, \lambda, \{S_i\})$, for a particular input segment $\omega \in \Omega$, there exists a fixed point in the lumped network of linear systems, $\eta_{lumped} = (\delta', \lambda', \{S_i'\})$, for the lumped input segment $\omega' \in \Omega'$ corresponding to $\omega \in \Omega$.*

Proof. In Theorem 1, we proved that fixed points are preserved by homorphisms. Here we extend this result at network level taking, for the sake of simplicity, an all-to-all coupling and a homomorphism based on the average of component states. The proof can be easily extended to all other cases of Example 1, by considering the set of senders. The fixed point in the base network consists of $\omega = \lambda(\delta(q, \omega))$. In a lumped network averaging both states and inputs, the latter equation can be developed as follows:

$$\begin{array}{rcll}
\omega & = & \lambda(\delta(q, \omega)) & \\
& = & \lambda(..., \delta_i(q_i, \omega_i), ...) & \text{based on base network structure (cf. Definition 7)} \\
& = & \lambda'(\frac{1}{n}\sum_{i=1}^{n} \delta_i(q_i, \omega_i)) & \text{based on lumped network structure (cf. Definition 8)} \\
& = & \lambda'(\delta_i(\frac{1}{n}\sum_{i=1}^{n} q_i, \frac{1}{n}\sum_{i=1}^{n} \omega_i)) & \text{based on linear system properties (cf. Definition 4)} \\
& = & \lambda'(\delta'(q', \omega')) &
\end{array}$$

□

Remark 4. As shown by this theorem, the usual mean field assumption requiring the number of components n to be infinite is sufficient but not necessary.

5.2 Matrix expression of homomorphisms

For linear systems, the matrix expression of homomorphisms is a simple and constructive way to implement general systems morphisms. We provide here the conditions to fulfill for such homomorphism to hold.

Theorem 3. *From a matrix representation point of view, let $H : Q \rightarrow^{onto} Q$ be a linear mapping, then H is a homomorphism from a big system SYS to a small system SYS', if the following conditions hold (cf. Figure 3):*

1. *$H'A = AH$, being the state-to-state linear mapping (or state mapping for short) with A "large" compared to corresponding parameter A' of the small system being "small",*
2. *$HB = B'$, being the input-to-state linear mapping (or input mapping for short) with B "large" compared to corresponding parameter B' of the small system being "small",*
3. *$C'H = C$, being the state-to-output linear mapping (or output mapping for short) with C "large" compared to corresponding parameter C' of the small system being "small".*

Proof. Let us take the simple example of the network of two 1-D linear components (cf. Example 2). The matrix form of the linear operators consists of $A = \begin{bmatrix} a & 0 \\ 0 & a \end{bmatrix}$, $Q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$, $B = \begin{bmatrix} b & b \end{bmatrix}$, $X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, $C = \begin{bmatrix} c \\ c \end{bmatrix}$ and $Y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$. Taking homomorphism $H \begin{bmatrix} 1 & 1 \end{bmatrix}$ and applying state linear mapping condition, we obtain $A'H = a \begin{bmatrix} 1 & 1 \end{bmatrix} = \begin{bmatrix} a & a \end{bmatrix}$ and $HA = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} a & 0 \\ 0 & a \end{bmatrix} = \begin{bmatrix} a & a \end{bmatrix}$. So $A'H = HA$. Applying input linear mapping, $HB = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} b & b \end{bmatrix} = [2b] = B'$. Applying output linear mapping, $C'H = [c] \begin{bmatrix} 1 & 1 \end{bmatrix} = \begin{bmatrix} c & c \end{bmatrix} = C$. Hence, the parameters of the lumped system consist of: $A' = [a]$, $B' = [2b]$, and $C' = [c]$, with $Q' = (a + bc)Q$, $Y' = Y = [y_1]$, and $X' = X = [x_1]$ through identity mapping $Id = [1]$. □

6 Lumping connected networks

6.1 Lumping large coupled networks into small coupled networks

Figure 11 shows the morphism between two base networks and two lumped networks.

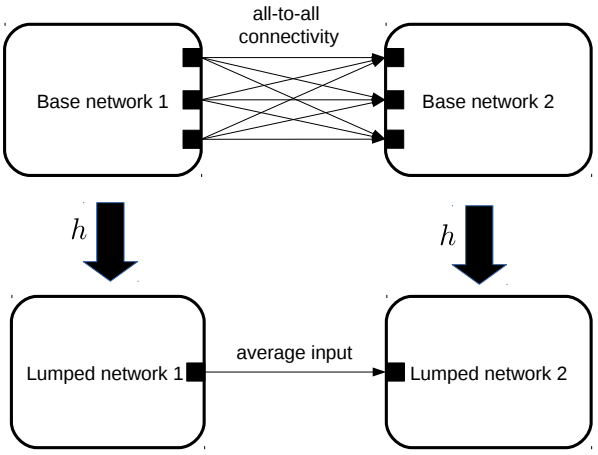

Figure 11: Morphism between two base networks and two lumped networks, an example of all-to-all coupling between the base networks.

The set of coupled base networks is closed under morphism, meaning that coupling base networks, a morphism holds when coupling corresponding lumped networks.

Theorem 4. *The dynamics of base linear networks is closed under morphism.*

Proof. Figure 12 shows the commutative diagram of the morphism between two base networks and two lumped networks, with the same number of components n. We prove here that the state transition of the target lumped network 2 depends on the output transition of the source network 1:

$$\begin{aligned}
\delta'(q'_1), \delta'(q'_2, \omega'_2) &= \delta'(\tfrac{1}{n}\textstyle\sum_{k=1}^{n} q_k^1), \delta'(\tfrac{1}{n}\sum_{k=1}^{n} q_k^2, \tfrac{1}{n}\sum_{k=1}^{n}\omega_k) \\
&= \delta'(\tfrac{1}{n}\textstyle\sum_{k=1}^{n} q_k^1), \delta'(\tfrac{1}{n}\sum_{k=1}^{n} q_k^2, \tfrac{1}{n}\sum_{k=1}^{n}\sum_{k\in S_i}\lambda_k(q_k)) \\
&= \delta'(\tfrac{1}{n}\textstyle\sum_{k=1}^{n} q_k^1), \delta'(\tfrac{1}{n}\sum_{k=1}^{n} q_k^2, \lambda'(\tfrac{2}{n}\sum_{k=1}^{n} q_k)) \qquad if\ all\ |S_i| = 2n \\
&= \delta'(q'_1), \delta'(q'_2, 2\lambda'(q')) \qquad \forall q' \in Q_1 \cup Q_2
\end{aligned}$$

□

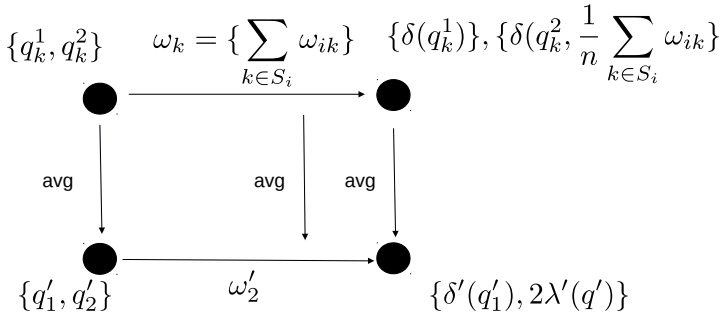

Figure 12: Commutative diagram of the morphism between two base networks and two lumped networks.

Figure 13 summarizes the large to small correspondence between the structures of base and lumped networks of networks.

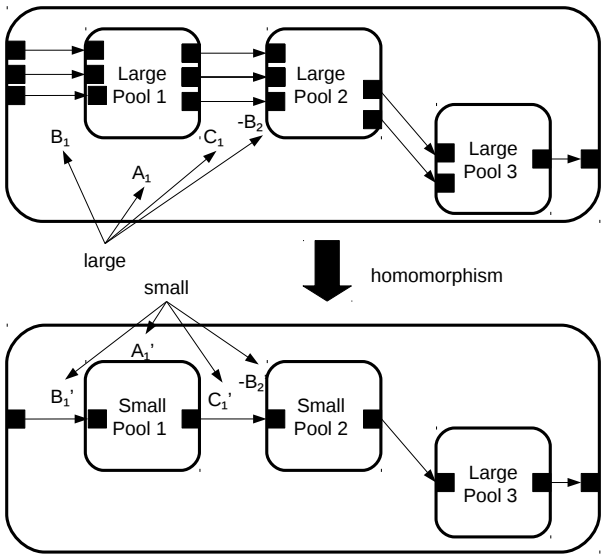

Figure 13: Couplings and lumping of a network of 3 networks. In the base network, the number of inputs (B), components and states in the state space (A), and outputs (C) is *large*. In the lumped network, corresponding numbers of lumped parameters $\{A', B', C'\}$ are *small*.

We will see hereafter that these conditions can be used to model network model connections.

6.2 Simple discrete formulation

Figure 14 describes the lumping of two connected pools of components with no internal couplings. The discrete time state dynamics of a neuron $i = 1, ..., n$ consists of

$$q_i(t+1) = q_i(t) + I_i(t)$$

Where I_i is the *input of component* i.

For a lumped network, the *lumped state* at time t, $Q(t)$, consists of the sum of the component states in corresponding base network, $Q(t) = \sum_{i=1}^{n} q_i(t)$, and the *lumped input* at time t, $I(t)$, is the average of external inputs, $I(t) = \sum_{i=1}^{n} \frac{I_i(t)}{n}$. So the state of a lumped of:

$$Q(t+1) = Q(t) + I(t)$$

With $I^1(t)$ as output of the first pool and input to the second pool. Then,

$$Q^2(t+1) = Q^2(t) + Q^1(t)$$

Feeding back the output of a pool to itself, it is obtained for each component, $I_i(t) = \sum_{i=1}^{n} \frac{Q_i(t)}{n}$, so $I(t) = \sum_{i=1}^{n} \sum_{i=1}^{n} \frac{q_i(t)}{n} = Q(t)$. So, $Q(t+1) = Q(t)+Q(t)$ and

$$Q(t+1) = 2Q(t)$$

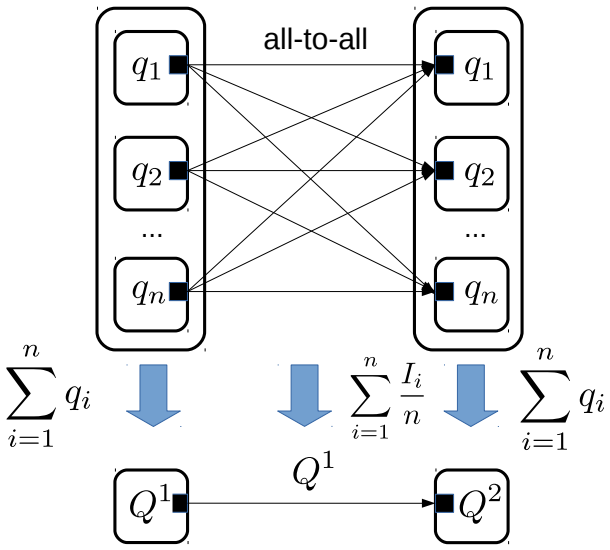

Figure 14: Simple discrete formulation of the lumping of two fully coupled networks.

This illustrates a *homomorphism* $h(\{q_i\}) = \sum_{i=1}^{n} q_i$. Starting with $q_i(t)$ the transition to the next state is $q_i(t+1) = q_i(t) + I_i(t) = q_i(t) + \sum_{i=1}^{n} \frac{Q_i(t)}{n}$. Applying h to both $\{q_i(t)\}$ and $\{q_i(t+1)\}$, we have $Q(t+1)$ and $Q(t) + \sum_{i=1}^{n} \sum_{i=1}^{n} \frac{Q_i(t)}{n} = 2Q(t)$ which match the lumped model transition $Q(t+1) = 2Q(t)$.

The solution of the lumped model is

$$Q(t) = Q(0)2^t$$

Remark 5. In neuronal pools, averaging the inputs can be conceived as averaging the firing rates of neurons when taking their inputs as firing rates.

6.3 Example of Cascade networks

Figure 15 shows an example of linear components connected via permutation coupling. All components are linear with the same coefficients multiplying the

input ports. But the inputs to the components do not have to be the same as required by all-to-all coupling.

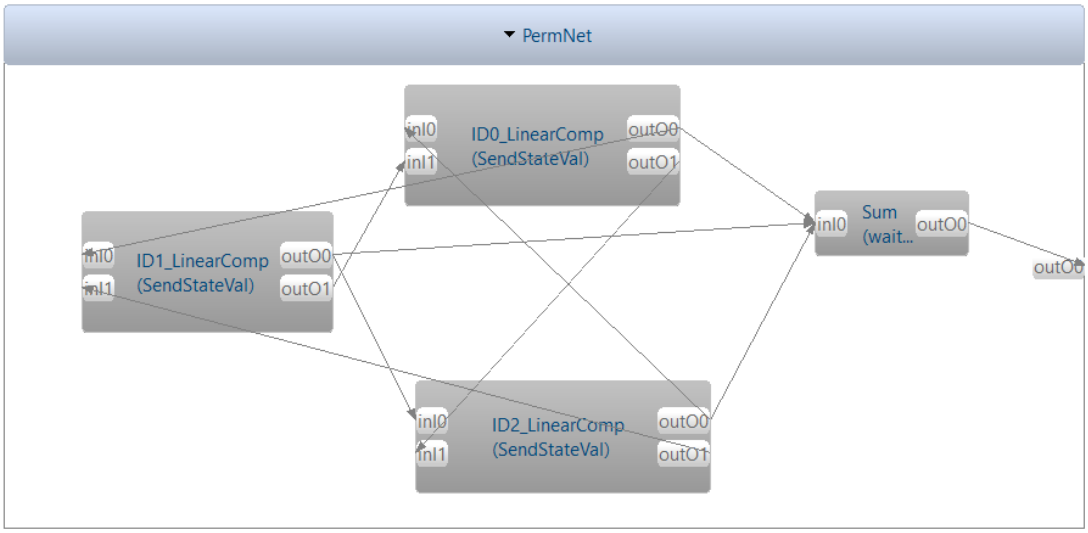

Figure 15: Example of a network called "PermNet" of linear components connected via permutation coupling on input/output ports, each port type connects all components through a permutation, i.e "$out0$" $\rightarrow$ "$in0$", "$out1$" $\rightarrow$ "$in1$", etc. (here only two). The output sum is what the lumped model predicts.

Figure 16 shows the connection of two networks of linear components connected via permutation coupling and how to check that the lumped models predict the activity in the base models. No self input is needed for fixed point. Convergence happens for the sum of coefficients adding to strictly less than 1. More precisely, equations and parameters consist of:

- The first base network: The input port "In0" of components "ID0", "ID1","ID2" consists of parameter a_0 while input port "In0" consists of a_1. The transition function components consists after using coupling of:

$$\begin{cases} q_{first}(ID0)(t+1) = a_0 q_{first}(ID2)(t) + a_1 q_{first}(ID1)(t) \\ q_{first}(ID1)(t+1) = a_0 q_{first}(ID0)(t) + a_1 q_{first}(ID2)(t) \\ q_{first}(ID2)(t+1) = a_0 q_{first}(ID1)(t) + a_1 q_{first}(ID0)(t) \end{cases}$$

- Corresponding first lumped network:

$$Q_{First} = q_{first}(ID0) + q_{first}(ID1) + q_{first}(ID2)$$

With:

$$\begin{aligned} Q_{First}(t+1) &= q_{first}(ID0)(t+1) + q_{first}(ID1)(t+1) + q_{first}(ID2)(t+1) \\ &= a_0 Q_{First}(t+1) + a_1 Q_{First}(t+1) \\ &= (a_0 + a_1) Q_{First}(t+1) \end{aligned} ,$$

with h parameter mapping: $a = (a_0 \ + a_1)$.

- First Base Component to Second Base Component Coupling is all-to-all for both couplings:

$$\begin{cases} q_{second}(ID0)(t+1) = a_0 q_{second}(ID2)(t) + a_1 q_{second}(ID1)(t) + a_0 Q_{First} + a_1 Q_{First} \\ q_{second}(ID1)(t+1) = a_0 q_{second}(ID0)(t) + a1 q_{second}(ID2)(t) + a_0 Q_{First} + a_1 Q_{First} \\ q_{second}(ID2)(t+1) = a_0 q_{second}(ID1)(t) + a_1 q_{second}(ID0)(t) + a_0 Q_{First} + a1 Q_{First} \end{cases}$$

- Corresponding second lumped network: $Q_{Second}(t+1) = a(Q_{Second}(t) + bI(t))$, with $I(t) = bQ_{First}$ and h parameter mapping: $a = (a_0 \ + a_1)$, $b = 3$. Then,$Q_{Second}(t+1) = (a_0 + a_1)(Q_{Second}(t) + 3 * Q_{First}(t))$ and $Q_{Second}(t+1) = (a_0 + a_1)Q_{Second}(t) + 3 * (a_0 + a_1)Q_{First}(t)$.

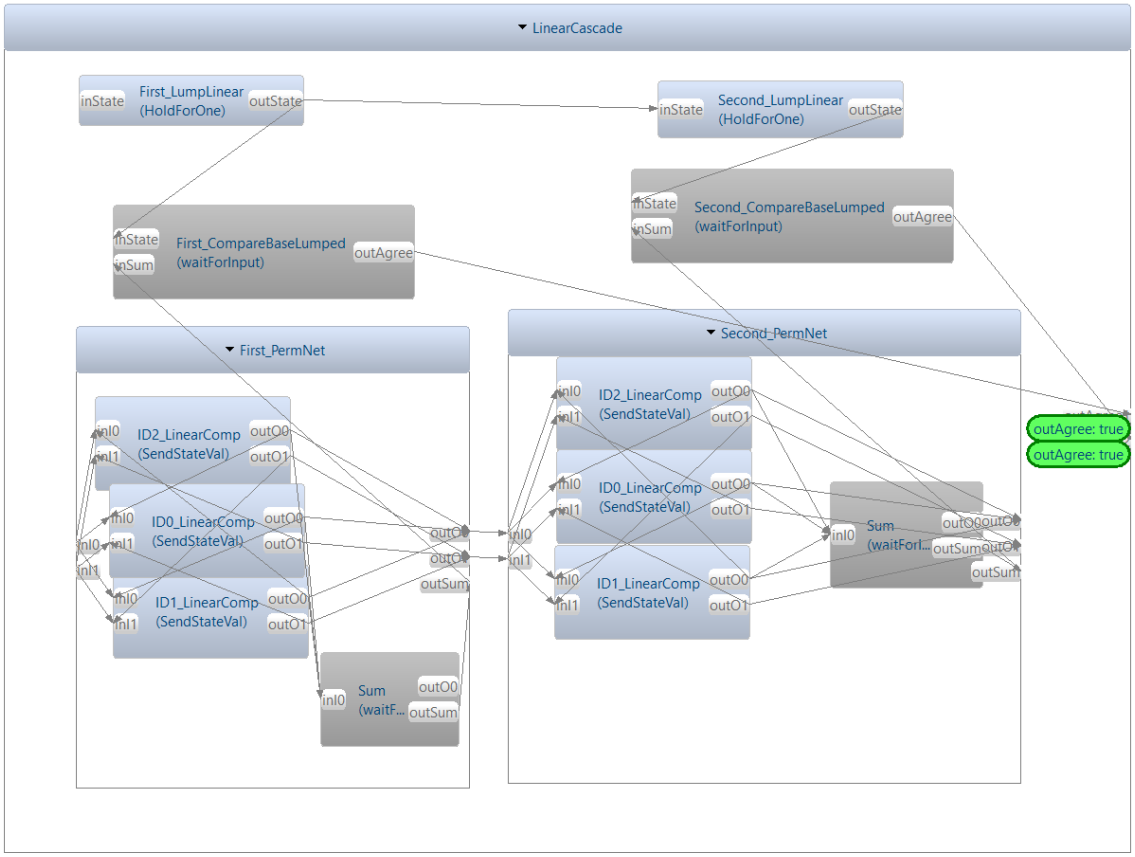

Figure 16: Two base networks "First_PermNet" and "Second_PermNet" are connected. Each one is lumped into networks "First_LumpLinear" and "Second_LumpLinear" are connected. "First_CompareBaseLumped" and "Second_CompareBaseLumped" compare if the activity dynamics of each base network matches corresponding lumped network.

7 Computational approximation

7.1 Uniformly to non-uniformly connected network in a discrete case

Based on the simple discrete example in Sub-section 6.2, the *exact lumped state* can be exactly computed as $Q(t) = Q(0)2^t$ with uniformly connected pools and an large number of components. However, real network simulation frequently consists of non-uniformly coupled networks, of finite size, leading to an approximation of the dynamics of the lumped network. Figure 17 shows the simulation lumping approximation, $\frac{Q_{sim}(t)}{Q(t)}$, where $Q_{sim}(t)$ is the *simulated lumped state*. It

can be seen that for a finite number of components, $n = 100$, the approximation of the lumped state is acceptable. The error increases after for smaller probabilities of connection. In a fully connected network a component samples all the states of other components. With, e.g., a probability $p = 30\%$, a component samples randomly 30% of all the components, at each step.

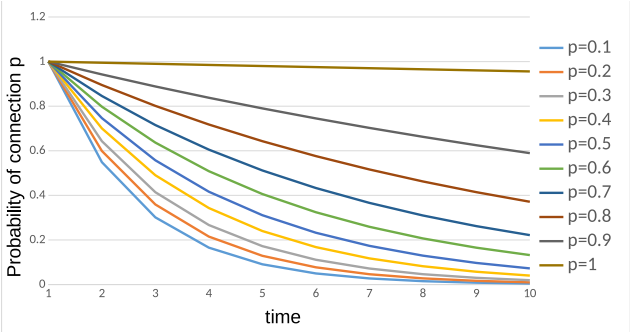

Figure 17: Lumping approximation as the ratio $\frac{Q_{sim}(t)}{Q(t)}$. Simulation are obtained according to the probability of connection p between the component pools, and with a number of components $n = 100$. Parameters of Equation 1 are $a = 1$ and $bc = -0.1$, so the exact state consists of $Q(t) = Q(0)0.9^t$, thus exhibiting a decreasing stable behavior.

We will see in the neural network application that stability and dynamics error accumulated at each state transition should not be confused.

7.2 Network activity vs deficiency in permutation connectivity

Figure 18 shows how a permutation-based connectivity can be degraded. The *domain of map D* consists of $\{0, 1, 2, 3, 4, 5\}$ and the *range of map D* consists of $\{1, 2, 3, 4, 5\}$. Map D is not a permutation since 0 is not hit. The departure from permutation is measured by the number of misses (=|domain Map|-|range Map|). But note that every miss also results in a multiple hit.

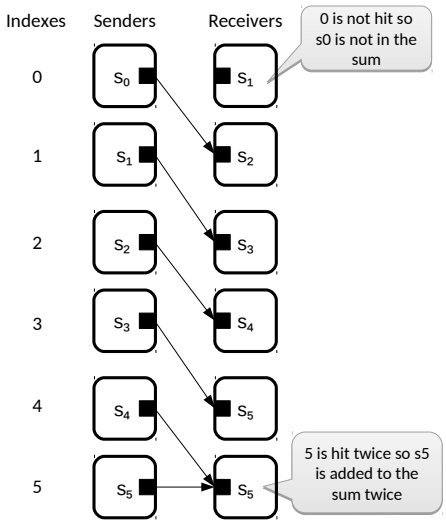

Figure 18: Error in sum due to permutation deficiency. The map D is not a permutation and consists of $D = \{(0,1),(1,2),(2,3),(3,4),(4,5),(5,5)\}$.

Let the component states, s_i, be either 0 or 1. Then, there is zero error when a missed component has a 0 value. Also a component that receives more than one input may contribute several ones to the sum. So the error in the sum may be zero even when the map is not a permutation. The error due to a permutation deficiency in a map D is then computed based on $TrueSum = \sum_{i \in Dom(D)} s_i$ and $ReceivedSum = \sum_{i \in Dom(D)} s_{D(i)}$, through the $Error = |TrueSum - ReceivedSum|$ and the $RelativeError = \frac{Error}{TrueSum}$.

Let's define the probability p of a 1 at a component where there are n ($= 10\,000$, eg.) components. So with $p = 0$, there are no 1's only 0's and $p = 1$ generates all 1's. So p represents how active the network is. Given a map, the number of 1's it sees versus the true number is the *Error*, where $p = 0$ and $p = 1$ there are obviously no error.

The computational experiment generates maps at random and takes the average of their errors as the values plotted in Figure 19. The error turns out to be max at $p = 0.5$. The average number of 1's (the true number) increases directly with p so the relative error decreases with p. This result can be interpreted as getting at worst a random activity error prediction based on a permutation connectivity deficiency.

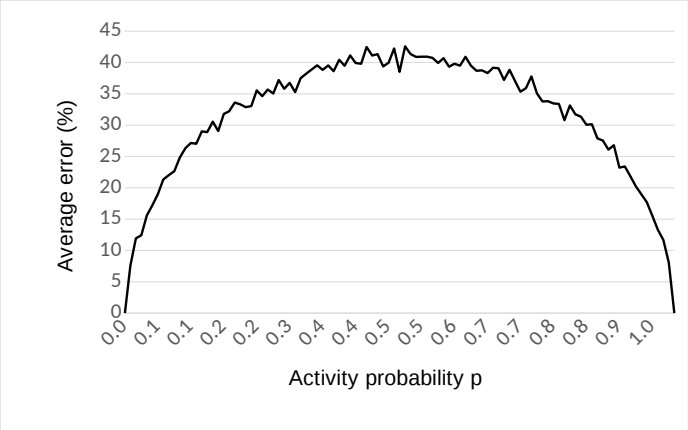

Figure 19: Average error with respect to the activity probability p, for a number of components $n = 10\,000$. For each activity probability p, 1 000 maps were generated.

Figure 20 shows that increasing the network activity probability, the average relative error goes to zero.

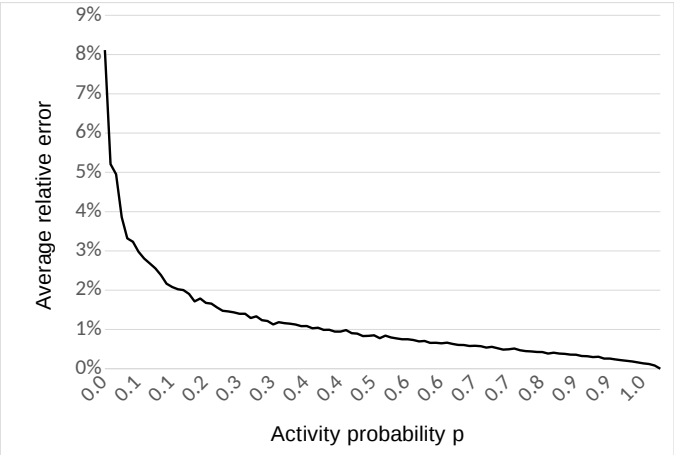

Figure 20: Average relative error with respect to the activity probability p, for a number of components $n = 10\,000$. For each activity probability p, 1 000 maps were generated.

Figure 21 shows the average relative error with respect to the number of components. It can be seen that increasing the number of neurons (and notice that it is possible to simulate easily 4 000 000 components), the error goes to zero. The connectivity is fixed at $p = 0.5$ for this experiment since it shows

the maximum error. Note that the plot uses the negative logarithm to show decrease in error over several decades.

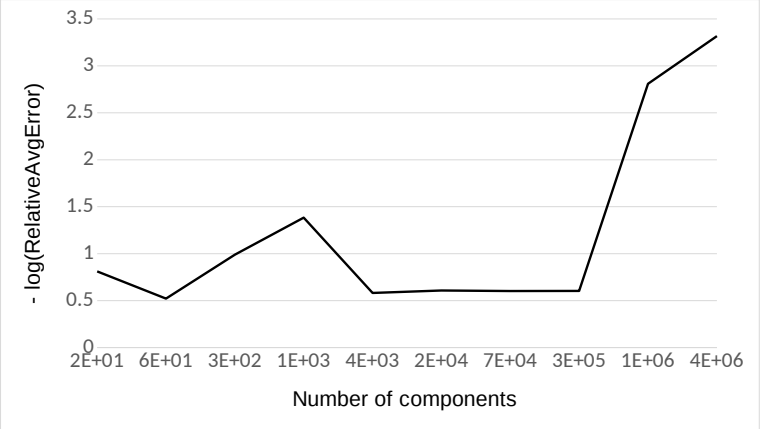

Figure 21: Negative log plot of average relative error with respect to the number of components up to 4 000 000 of components.

8 Application to neuronal networks

8.1 Motivation

Figure 22 describes the lumping application onto models of brain regions. The base model consists of many details preventing simulation under reasonable execution times, while the lumped model abstracting the base model details aims to be simulatable.

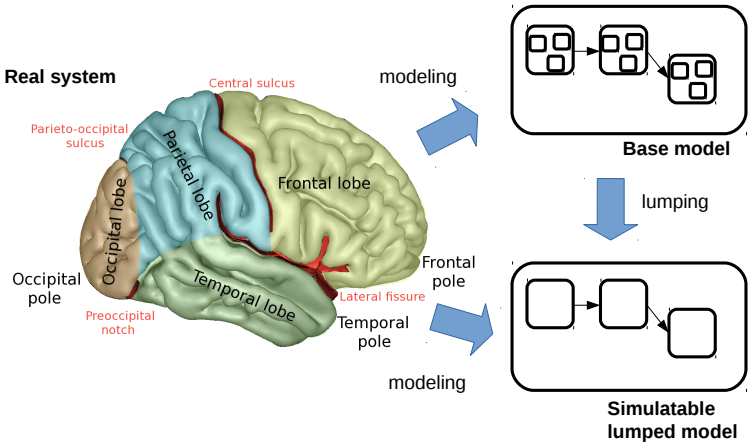

Figure 22: Modeling of the brain regions and model lumping for simulatability. Free picture of the brain lobes from Wikipedia.

Figure 23 presents a coupling between two brain regions that can be considered as coupling two neuronal networks with inhibitory external synaptic connections coming from another network (or brain region). Inside the networks, synaptic connections are considered to be excitatory. This assumption follows the one of balanced networks (Brunel 2000).

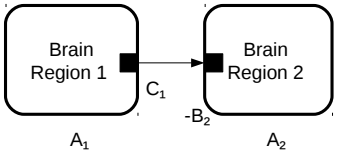

Figure 23: Couplings between two brain regions. The synaptic inputs received by Brain region 2 from Brain region 1 are inhibitory thus leading to matrix $C_1 - B_2$ for output-to-input mapping.

The goal is to get simulatable models of brain regions. This can be understood easily by a simple analytical formula. Imagine a detailed network with 10^6 discrete neuron models and 100 possible states each. The total number of possible network states is then $(10^6)^{100} = 10^{600}$, which is not simulatable. Being able to lump this detailed network into 1000 lumped components having 10 possible states each, the possible number of states of the lumped network is then 1000^{10}, which makes the lumped model simulatable.

8.2 Neuron dynamics model

A well known model of neuron dynamics is the the Amari-Wilson-Cowan model (Amari 1972; Wilson and Cowan 1972; Wilson and Cowan 1973), slightly modified by taking external input current to the network as external inputs from another network:

$$\frac{dq_i}{dt} = aq_i + \sum_{j \in S_i} b_{ij} f(x_j) \tag{2}$$

Where q_i represents the *voltage of the neuron* i, $a \leq 0$ is the *leak parameter*, b_{ij} is a *synaptic weight* (with $b_{ij} < 0$ an *inhibitory synapse* and $b_{ij} > 0$ an *excitatory synapse*), x_j is the input voltage received from an influencing (external or internal to the network) neuron j in S_i, and f is a typical non-linear sigmoid function $f(x) = \frac{1}{2}(1 + tanh(gx))$, g a gain parameter (cf. Figure 24). "In region I (low voltage), the neuron does not emit spikes. In region II, $f(x_j)$ is roughly linear. In region III (high voltage), the firing rate reaches a plateau, fixed by the refractory period"(Cessac 2019).

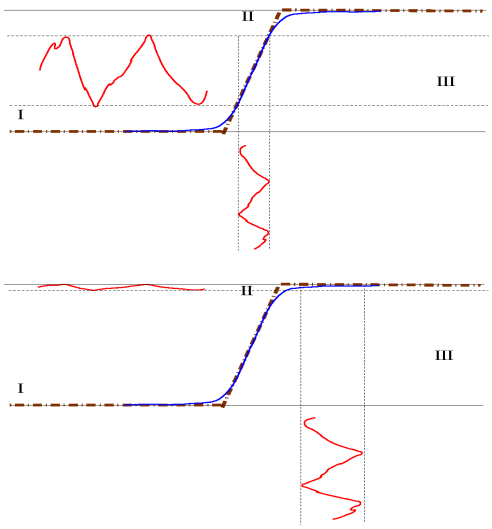

Figure 24: The sigmoidal shape of the function f and its effects on voltage fluctuations. Top: When neuron's voltage fluctuates around the inflection point of the sigmoid, and if the gain g is large enough fluctuations are amplified. Bottom: When the neuron's voltage fluctuates in the flat parts of the sigmoid (here, the saturated region) fluctuations are damped. Dashed red lines correspond to a piecewise linear approximation of the sigmoid, allowing to delimit regions I, II, III (legend and figure from (Cessac 2019)).

Lumping neurons consists of using the linear part of the sigmoid, with parameters $bcg = b$ for simplification, where c is the *signal attenuation on the*

axon, b is the *pulse attenuation on synapses and dendrites.*

9 Computational approximation, dynamics and stability

Figure 25 describes the error dynamics in the lumping model. In stable linear systems, any two states converge to one state unless there is a break of the convergence because of :

1. The non linearities of the sigmoid,
2. Positive synaptic weights, b_{ji},
3. Random sampling couplings instead of a uniform ones in brain regions. Then, the dynamics could differ just a little bit from the average.

In all cases, lumping analytic results are not possible anymore and simulation is required to study the dynamics between brain regions.

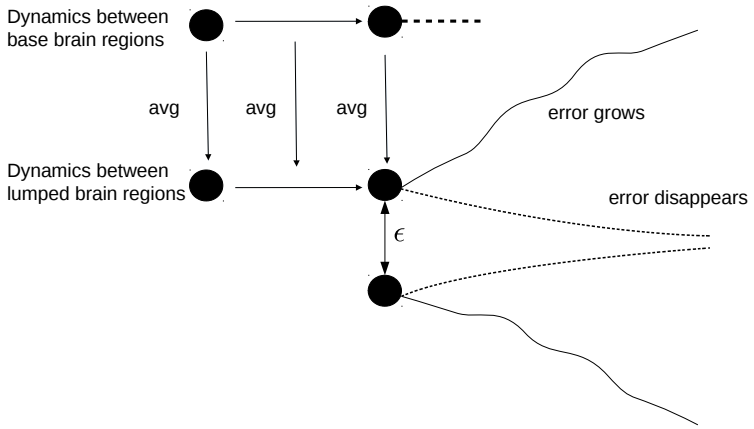

Figure 25: The error depends on the dynamics between base and lumped brain regions: Usually, the brain is a stable system where the error of the lumped model used for study should disappear. Sometimes, the brain turns unstable (as during epileptic crises).

10 Conclusion

Sufficient and necessary mean field conditions have been proposed in Definition 6 for abstracting I/O general linear networks. These conditions proved to be implemented by system morphisms leading to fixed points in networks (cf. Theorems 1 and 2) and allowing lumping coupled networks (cf. Theorem 6.1) with no error at each system transition. The error at each transition has been

explored based on deficient permutation-based couplings in the network as well as a non infinite number of components. We believe that this whole approach provides a computational method for both abstracting general linear systems and investigating the computational error induced at each transition, based on a more realistic network connectivity. Our next work will consist of investigating probabilistic mean field conditions (as e.g. shown in (El Boustani and Destexhe 2009; Brunel 2000)) and their error approximation based on system theory morphisms.

References

Amari, Shun-Ichi (1972). "Characteristics of random nets of analog neuron-like elements". In: *IEEE Transactions on Systems, Man, and Cybernetics* 5, pp. 643–657.

Arbib, Michael A (1972). *Theories of abstract automata*.

Arnold, André (1994). *Finite Transition Systems. International Series in Computer Science*.

Brunel, Nicolas (2000). "Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons". In: *Journal of computational neuroscience* 8.3, pp. 183–208.

Cessac, Bruno (Oct. 2019). "Linear response in neuronal networks: from neurons dynamics to collective response". In: *Chaos* 29.103105. DOI: 10.1063/1.5111803. URL: https://hal.inria.fr/hal-02280089.

Destexhe, Alain and Terrence J Sejnowski (2009). "The Wilson–Cowan model, 36 years later". In: *Biological cybernetics* 101.1, pp. 1–2.

El Boustani, Sami and Alain Destexhe (2009). "A master equation formalism for macroscopic modeling of asynchronous irregular activity states". In: *Neural computation* 21.1, pp. 46–100.

Faugeras, Olivier D, Jonathan D Touboul, and Bruno Cessac (2009). "A constructive mean-field analysis of multi population neural networks with random synaptic weights and stochastic inputs". In: *Frontiers in Computational Neuroscience* 3, pp. 1–28. ISSN: 1662-5188. DOI: 10.3389/neuro.10.001.2009. URL: http://dx.doi.org/10.3389/neuro.10.001.2009.

Harrison, Michael A (1969). *Lectures on linear sequential machines*. Tech. rep.

Ho, Yu-Chi (1992). *Discrete event dynamic systems: analyzing complexity and performance in the modern world*. IEEE.

Ivanov, E. (2013). *Investigation of abstract systems with inputs and outputs as partial functions of time*. Phd dissertation. Taras Shevchenko National University of Kyiv, Ukrain.

Klir, George J (1985). *Architecture of systems complexity*. Saunders, New York.

Mesarovic, M.D. and Y. Takahara (1975). *General Systems Theory: Mathematical Foundations*. LNCIS 116. Academic Press.

— (1989). *Abstract Systems Theory*. LNCIS 116. Springer.

Nykamp, Duane Q et al. (2017). "Mean-field equations for neuronal networks with arbitrary degree distributions". In: *Physical Review E* 95.4, p. 042323.

Ostojic, Srdjan (2014). "Two types of asynchronous activity in networks of excitatory and inhibitory spiking neurons". In: *Nature neuroscience* 17.4, p. 594.

Wilson, Hugh R and Jack D Cowan (1972). "Excitatory and inhibitory interactions in localized populations of model neurons". In: *Biophysical journal* 12.1, pp. 1–24.

— (1973). "A mathematical theory of the functional dynamics of cortical and thalamic nervous tissue". In: *Kybernetik* 13.2, pp. 55–80.

Wymore, Wayne (1967). *A mathematical theory of systems engineering*. Wiley.

Zadeh, L.A. and C.A. Desoer (1963). *Linear system theory: the state space approach*. McGraw-Hill series in system science. McGraw-Hill.

Zeigler, Bernard P, Alexandre Muzy, and Ernesto Kofman (2018). *Theory of Modeling and Simulation: Discrete Event & Iterative System Computational Foundations*. Academic press.