Coupling and Lumping Finite-Size Linear System Realizations of Componentized Neural Networks

Alexandre Muzy, Bernard P Zeigler

Coupling and Lumping Finite-Size Linear System Realizations of Componentized Neural Networks

Alexandre Muzy*, Bernard P. Zeigler†

Abstract

We present here a system morphism methodology to give insight into the lumping process of networks of linear systems. Lumping networks allows reducing the number of components and states to obtain simulatable models. Such lumped networks can be connected together through their input/output interfaces, using an engineering approach componentizing and lumping the network. This opens interesting perspectives for analyzing real networks at computational level (computing units of computers, simulations running on these computing units, models of neural networks based on a finite number of recording electrodes, etc.). In particular, the transposition of our results to brain modeling and simulation is discussed.

1 Introduction

In computational neuroscience, based on methods from statistical mechanics, the mean field approach is used widely to lump a large pool of neurons[1] into a single behaviorally equivalent unit (i.e., both unit and pool having the same input/output trajectories). Further, such a unit can be employed as a unit in larger scale neuronal network models (Faugeras, Touboul, and Cessac 2009; Nykamp et al. 2017; Ostojic 2014; El Boustani and Destexhe 2009). The results obtained are specific to the derived neuronal equations, they are not generic for any (non)linear systems. Both structure and dynamics of the pool of neurons constitute a rough abstraction of the complexity of an actual neuronal pool. However, this abstraction allows inferring 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 allows manipulating and reasoning 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 a morphism methodology for lumping constructively any linear systems. Basic mean field assumptions are summarized as an example of two particular/significant assumptions for lumping. Lumping systems based on morphisms is more general than averaging a single state variable of a system using mean field method. *Lumping* systems consists generally in respecting the input/output behavior of a system *grouping equivalent states* of a base system *into blocks*, and mapping each block to each state of a lumped system (*cf.* Appendix A). The behavior is respected by a transition mapping if states in the same block go to the

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

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

[1]In a network model, *a pool consists of grouping equivalent components together.*

same (possibly other block) under the morphism mapping*s* (between the states, inputs and outputs of the base and lumped models) at each transition of the base model and corresponding lumped *model.* Computational modeling is done using a system specification formalism (cf. for more details on levels of system specifications and morphisms: (Zeigler, Muzy, and Kofman 2018)). The coupling of systems (components) constitutes a network. Base network models of linear systems are then abstracted into lumped network models. The lumping is detailed based on the base network model. Particular mean field conditions ensure the lumping preservation of the mean of the firing rate at neural network level. 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 system models modularly through their input/output interfaces following an engineering approach.* It allows also the preservation of fixed points from components to resulting network. The results are conceived in the context of neuronal network and a linear version of the Amari-Wilson–Cowan model (Wilson and Cowan 1973). "The derivation of the Wilson–Cowan model depended on taking the "thermodynamic limit" (in which the number of neurons approaches infinity), but finite-size effects in systems with a finite number of neurons can be important" (Destexhe and Sejnowski 2009). Although usual mean field methods (except e.g. (El Boustani and Destexhe 2009)) consider an infinite number of neurons, networks with a finite number of neurons are essential for real world applications: on neuromorphic computers with a finite number of processors and cores (Mayr, Hoeppner, and Furber 2019) computing neuron dynamics, for any simulation running on computers with a finite number of processors/cores, or considerning the finite number of electrodes able to record the dynamics of neurons (Hong and Lieber 2019). *Beyond neuronal systems, the goal of this study is to show how exact morphisms*[2] can be used for lumping finite-size networks of any linear systems.

Figure 1 describes the lumping application onto plausible 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 (see (Zeigler, Muzy, and Kofman 2018) for a detailed discussion of simulation computational complexity). Figure 13a 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). **The first goal of lumping** is to get simulatable models of brain regions. This can be understood easily by calculation of state space size. Imagine a detailed network with $1\,000\,000$ discrete neuron models and 100 possible states each. The total number of possible network states is then $100^{1\,000\,000} = 10^{2\,000\,000}$, which is not simulatable (assuming each state needs to be accessed in a simulation, and can be computed in 10^{-10} seconds, the total time required is $10^{2\,000\,000-10}$ seconds, more than the age of the universe ($< 10^{18}$ seconds)). Being able to lump this detailed network into 10 lumped components having 10 possible states each, the possible number of states of the lumped network is then 10^{10}. This takes 1 second, which makes the lumped model quite simulatable (cf. (Mayr, Hoeppner, and Furber 2019)). In conclusion, the number of states being the component state size to the power of the number of components requires really a small number of components! **The second goal of input/output system lumping** is to understand how mean field networks interact through their inputs/outputs. These interactions are made possible by a "flexible" definition of input/output segments of trajectories allowing discontinuities, continuous and/or discrete time bases as well as corresponding state based computations of the components of the network.

[2]Exact morphisms do not induce any error at each state transition between the detailed and the lumped model.

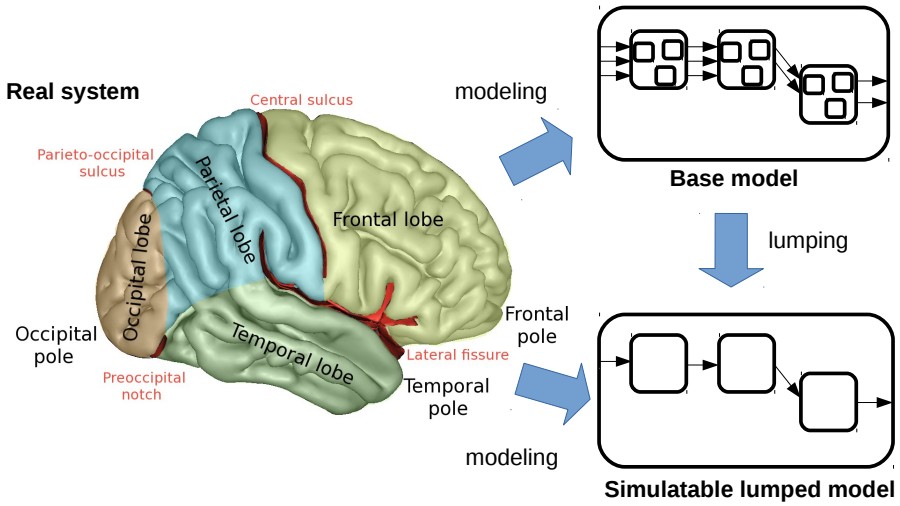


Figure 1: Modeling of the brain regions and model lumping for simulatability. Free picture of the brain lobes from Wikipedia.

In Section 2, mathematical system and mean field theories are introduced. In Section 3, the mathematical framework is defined for general (networks of) input/output systems. Linear input/output systems are introduced. In Section 4, the lumping of linear (networks of) systems is achieved setting morphisms for mean field conditions. Section 5 presents the lumping of linear connected networks based on mean field conditions. Section 6 discusses the results obtained for linear systems in the context of (non)linear neuron models. Finally, in Section 7, conclusion and perspectives are provided.

2 Mathematical general system and mean field theories

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 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.

Originally from statistical mechanics, mean field theory provides techniques for approximating the interactions of many components (particles, atoms, and further individuals, neurons, etc.) into a simpler model considering the average of the interactions of some specific, physically relevant, quantities over a large population of particles. A typical example is the average of local magnetic properties of atoms to produce a mesoscopic quantity, the magnetization, characterizing the magnetic property of a sample containing many atoms. In Neurosciences, the following *modeling assumptions* lead to *particular statistical properties* at population level:

1. Considering fixed random weights of synaptic interactions: The normalization of the weights in $\frac{1}{n}$, with n the *number of neurons* (Amari 1971) allows approximating, by the law of large numbers, the interactions firing activity as the average of neurons firing activity. The normalization of the weights in $\frac{1}{\sqrt{n}}$ (Faugeras, Touboul, and Cessac 2009) allows approximating the interaction firing activity as the average of neurons firing activity but with a non zero variance thus requiring the characterization of the co-variance between neurons.

2. Considering that the membrane potential computation of neurons is subject to noise (Touboul, Hermann, and Faugeras 2012), its fluctuations, periodicity,... should be studied in details.

This study concerns networks with a finite number of components. The connectivity between the components is seen as a map between sending components (senders) and receiving components (receivers). This map can be seen as one-to-one, all-to-all, or generated randomly (using an Erdős-Rényi graph) leading to a uniform distribution of the connections between the components. In the previous studies from neuroscience, uniform connectivity (or all-to-all) condition is usually combined with an infinite number of components to satisfy either the law of large numbers or the central limit theorem, possibly leading to *no error* (based on the models and weight distribution) between the activity in the detailed model and the activity of the mean field model. Conversely, having a finite number of neurons induces a statistical error between detailed and mean field models (El Boustani and Destexhe 2009).

Table 1 compares for representative neural network mean field models: the size finitness, the type of connectivity and the lumping error existence. For all-to-all connectivity of neurons and between networks, proving chaos propagation (Ben Arous and Zeitouni 1999), is *sufficient* but *not necessary* for satisfying mean field conditions. **We will show here that *permutation-based connectivity*[3] inside and between networks of a *finite number* of any linear systems is *sufficient*[4] for exact lumping morphisms to construct mean field network models.**

publication	size	connectivity	lumping error
(Sompolinsky, Crisanti, and Sommers 1988)	infinite	all-to-all	no
(Nykamp et al. 2017)	infinite	arbitrary random degrees	no
(El Boustani and Destexhe 2009)	finite	uniformly random	yes
(Ostojic 2014)	infinite	uniformly random	no
in this article	finite (or infinite)	permutation-based	no

Table 1: Mean field conditions in neural networks.

3 Mathematical framework

3.1 Input/output systems

3.1.1 I/O general state-based systems

Definition 1. A *deterministic general I/O SYStem (SYS) is a structure* (cf. behavior introduced in Figure 2a)

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

Where

$\delta : Q \times \Omega \to Q$ is the *transition function*, with Q the *set of states*, $\Omega \subseteq X^T$ the *set of (piecewise bounded) input segments* $\omega :< t, t_2 >\to X$[5] (a subset of all possible input segments with values in X over the time base T), with $< t_1, t_2 >$ the *interval of the segment*[6] . The sets of input values X and states[7] Q are arbitrary.

[3]Notice that permutation-based connectivity is used also in (Ben Arous and Zeitouni 1999).

[4]Remark that many connectivity structures can generate the same behavior and thus *necessity* is harder to prove in generality and depends on a particular type of model.

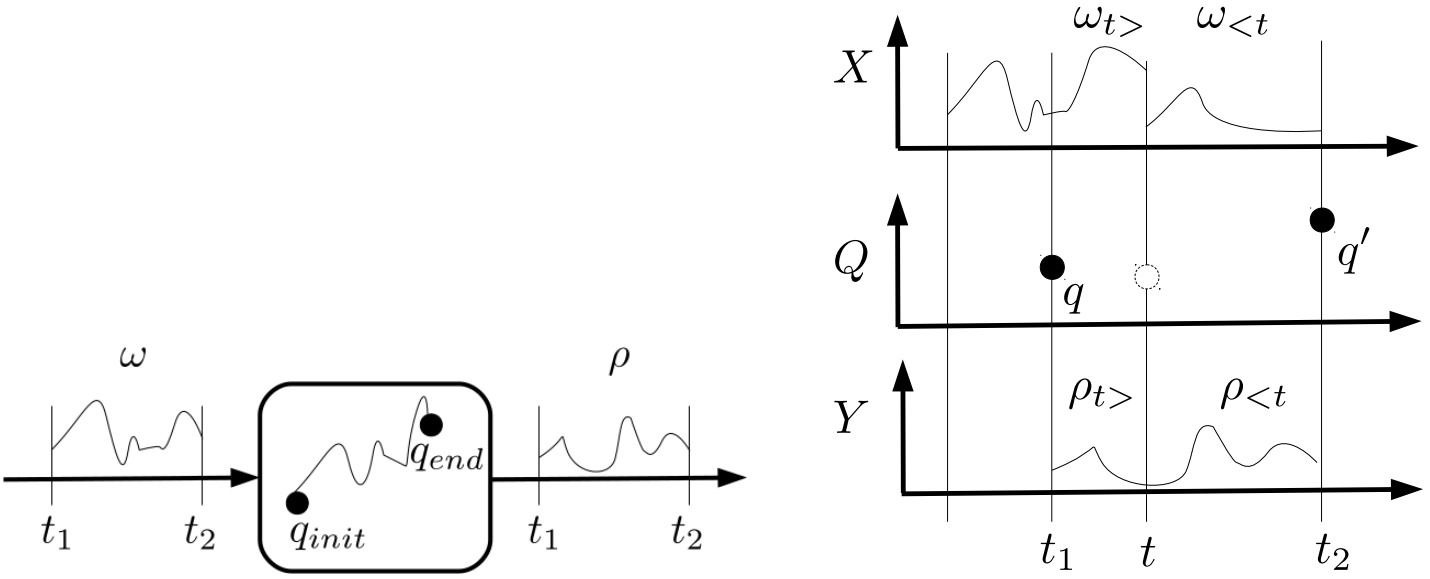
[5]A piecewise continuous input segment is a map from each time point $t \in< t_1, t_2 >$ (with t_1 and t_2 not fixed) to a corresponding input value $x \in X$.

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

[7]The set of states can be defined as $Q = \Pi_{i=1}^{n} V_i$, where i is the index (name) of a variable and V_i the range of the variable i. Then, every state consists in the vector of variable values at a particular time.

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

For one input segment $\omega \in \Omega$ defined over an interval $< t_1, t_2 >$, with t_1 and t_2 not fixed[8], 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 2b): $\delta(q, \omega) = \delta(\delta(q, \omega_{t>}), \omega_{<t})$, 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 $\rho_{<t}$. The *dynamics* of the system in input, state and output consists of the composition of input/output segments (called *generators* in an abstract algebra context) and state transitions. More information can be found in (Muzy, Zeigler, and Grammont 2017; Zeigler, Muzy, and Kofman 2018) about the proof of such compositions of generators. Using such generators, notice that as shown in Figure 2b, segments can easily have different lengths (the *length of a segment* is noted $l(\omega_{t>}) = t_1 - t$) as well as discontinuities from one segment to another, e.g., $\omega_{t]}(t) \neq \omega_{[t}(t)$ or not, e.g., $\rho_{t]}(t) = \rho_{[t}(t)$. Segments do not share necessarily the same domain. This allows systems to interact over a hybrid time base. For example, an input segment $\omega \in \Omega$ can be defined over a continuous time base ($T_c = dom(\omega) =< 0, t_f > \cap \mathbb{R}^+$) while an output segment $\rho \in P$ can be defined over a discrete time base ($T_d = dom(\rho) =< 0, t_f > \cap \mathbb{N}$). A resulting hybrid time base then consists of $T = T_c \times T_d$. The domain of each segment is then obtained by projecting the time base.



(a) 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 $\rho \in P$.

(b) Composition of segments.

Figure 2: System behavior and computations.

The *current state* is the minimal information to deterministically compute the *next state* in a very large state space. The system is assumed to be 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 (finitely many) previous state transitions. Notice also that the system holds inputs and outputs, which is a more general and convenient principle for

[8]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).

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 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 $q \in Q$, $\delta(q, \omega) = \delta(q, TRANS_\tau(\omega))$.

3.1.2 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* $\mathbb{R}$,

$\delta : Q \times \Omega \rightarrow Q$ is the *transition function*[9], where sets are the same than in Definition 1 and $\delta(q, \omega) = qe^{Al(\omega)} + \int_0^{l(\omega)} e^{A(l(\omega)-\tau)} B\omega(\tau)d\tau$, with $\omega \in \Omega$ a *piecewise bounded continuous input segment* that can be segmented for computations as shown in Figure 2b (where $\omega_{t>}(t) = \omega_{<t}(t)$)),

$\lambda : Q \rightarrow Y$ is the *output function* where sets are the same than in Definition 1 and $\lambda(q) = Cq$ is the *output function*,

$A : Q \rightarrow Q$, $B : X \rightarrow Q$ and $C : Q \rightarrow Y$ 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. $\delta(q_1 + q_2, \omega_{1,t>} + \omega_{2,t>}) = \delta(q_1, \omega_{1,t>}) + \delta(q_2, \omega_{2,t>})$, with $dom(\omega_{1,t>}) = dom(\omega_{2,t>})$ so $l(\omega_{1,t>}) = l(\omega_{2,t>})$.
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. $\delta(q_1 + q_2, \omega_{1,t>} + \omega_{2,t>}) = \delta(q_1, \omega_{1,t>}) + \delta(q_2, \omega_{2,t>})$

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

Proof. Based on matrix representation,

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

□

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,

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

□

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

Remark 1. Although by definition two input segments of linear systems received in parallel need to have the same length, i.e., $l(\omega_{1,t>})=l(\omega_{2,t>})$, still the sequential computation (or composition) of segments follows the general system definition 1, i.e., segments can have different lengths. This can be used to model piecewise linear systems (Coombes et al. 2018) where the different lengths of segments can be simulated based on a Discrete Event System Specification (DEVS)(Zeigler, Muzy, and Kofman 2018). Besides, even being out of the scope of the present contribution, disposing of linear systems with inputs and outputs, allows them to receive discontinuous input events leading to hybrid system specification (cf. (Zeigler, Muzy, and Kofman 2018) for more details). Moreover, as previously defined, segments can be either defined over a continuous or a discrete time base.

Remark 2. The composition property of linear systems can be conceived as a Koopman operator (Mauroy and Mezic 2020). The state set of such systems has no structure and is partially observable. Beyond segment flexibility, the other difference with LSYS structure is that Koopman operator does not allow inputs nor outputs for connecting systems together. Besides, while homomorphisms (discussed after) can be studied using a Koopman operator, again, morphisms are defined to account for inputs/outputs.

Remark 3. Notice that LSYS can be a hybrid system with segments of different lengths. Each segment can lead to a fixed point corresponding to mean field conditions. However, discontinuous inputs (discrete events) can occur to go from one fixed point to the other.

3.1.3 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. While *homomorphisms* are current for linear systems (Mauroy and Mezic 2020) and abstract algebraic machines (Hartmanis 1966) at state level, accounting for ("flexible") segments and inputs/outputs requires developing *morphisms* (more details in (Zeigler, Muzy, and Kofman 2018; Muzy, Zeigler, and Grammont 2017)).

Definition 5. A *system morphism* or generalized homomorphism[10], 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 $q \in \overline{Q}$, $\omega' \in \Omega'$,$h(\delta(q, \omega)) = \delta'(h(q), \omega')$ (transition function preservation)
4. $k : P \to P'$, is the output mapping.

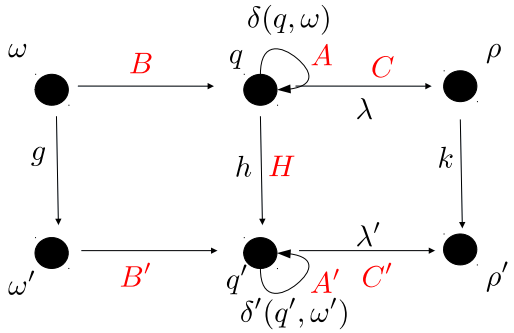


Figure 3: Commutative diagram of morphism mappings from a big system SYS to a small system SYS'. In red are indicated the matrix representations to be discussed for network representation.

3.2 Network of systems

3.2.1 Mathematical structures and lumping

Figure 4 shows a base network, a lumped network and the morphism between.

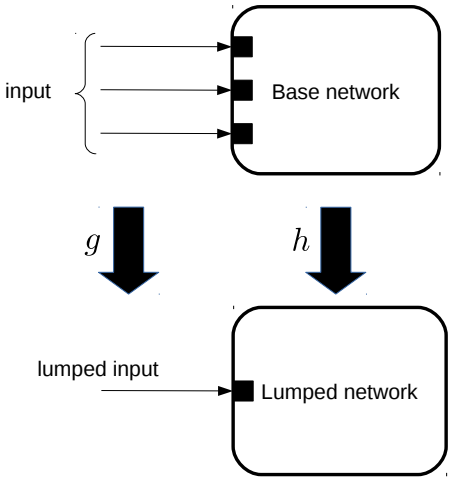


Figure 4: Morphism between base and lumped networks.

Let's define now more precisely the base and lumped model structures (see (Zeigler, Muzy, and Kofman 2018) for details). This definition concerns general systems as presented in Definition 1 but we will see later how to apply this general definition to linear systems. Notice also that it is here a non-modular definition where each component is not a mathematical structure. Rather, network dynamics relies on a global transition upon component (system) states.

[10]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.

Definition 6. A *base network of systems* consists of

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

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,

D_{int} is the set of indexes of systems inside the network,

$\{S_{i \in D_{ext} \cup D_{int}}\} = \{S_i^{ext} \cup S_i^{int}\}$ is the *set of sending systems j to a receiving system i, $S_i = (\delta_i, \lambda_i)$ (cf. Definition 1), to/inside the network* such that $S_i = \{j \mid \omega_i = \lambda_j(\delta_j(q_j, \omega_j)\}$ is the *set of sending components j external, i.e., $j \in D_{ext}$, or internal, i.e., $j \in D_{int}$,* to the network connected to the input, $\omega_i \in \Omega_i$, of *a receiving* component i.

$\{Z_{i \in D_{int} \cup \{\eta\}}\}$ is the *set of coupling maps* such that for each receiving component $i \in D_{int} \cup \{\eta\}$: $Z_{i \in D_{int} \cup \{\eta\}} = \{(j, i) \mid j \in S_{i \in D_{ext} \cup D_{int}}\}$.

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

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

Where

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

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

D'_{int} is the set of indexes of systems inside the network,

$\{S'_{i \in D'_{ext} \cup D'_{int}}\}$ is the *set of senders to/inside the network* such that $S'_i = \{j \mid \omega' = \lambda'_j(\delta'_j(q'_j, \omega'_j)\}$ is the *set of sending components j* to the network, connected to the input $\omega' \in \Omega'$.

$\{Z'_{i \in D'_{int} \cup \{\eta'\}}\}$ is the *set of coupling maps* such that for each receiving component $i \in D'_{int} \cup \{\eta'\}$: $Z'_{i \in D'_{int} \cup \{\eta'\}} = \{(j, i) \mid j \in S'_{i \in D'_{ext} \cup D'_{int}}\}$.

In a *base network,* a *block (or vector) 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 block of states in the base network matches a single state in corresponding lumped network. The mapping (or homomorphism) between state blocks to single states can be achieved (instead of a simple average or summation) as a *census of states*, i.e., as counting the number of components in a particular state.

The method for modeling and lumping networks consists of the following independent steps: either (1) specify the base network model (using Definition 6) or (2) specify the lumped network model (using Definition 7), and possibly (3) specify the morphism triple (g, h, k) (using Definition 5). Notice that this method works for any I/O systems.

3.2.2 Exact morphisms

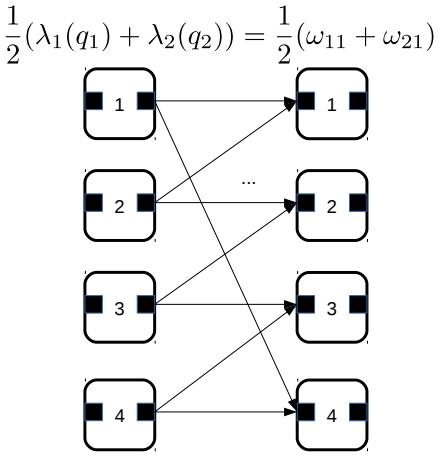
Exact morphisms[11] lead to no error prediction of the base network state by the lumped network. At each transition in both networks there should be a perfect match of states. This requires both dynamics and structure homogeneous conditions in the base network. *Dynamics homogeneous condition* consists of having all components in the base network having the same transition/output functions and parameters. *Structure homogeneous condition* consists of having a permutation-based coupling of components in the base network.

A permutation-based network connectivity is shown in Figure 5a. At connectivity level, each receiver gets the same values and has the same number of senders. Think of permutations as global constraints on the connectivity of the network. Both examples in Figures 5b and 5c 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. Represent the connectivity between sender and receiver indexes like this: $(0, x)(1, y)(2, z), ...$, 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 is how the number of permutations is obtained: $n! = n(n-1)(n-2) \times ... \times 2 \times 1$. Combinations (vs. permutations) are made if replacement is made each time we make a choice. So then we get $n^n = n \times n \times n \times .... \times n$ or $3 \times 3 \times 3 = 27$ 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 Figure 5c, the component 8 is hit twice - once for each map.

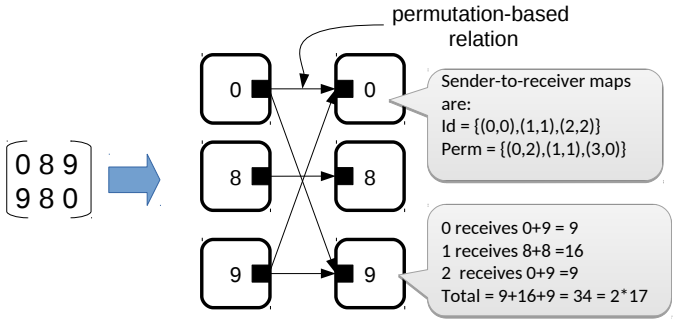
In a 2-D cellular space, the identity and each direction can be a map - so there are 5 maps: identity, North, South, East, West (2 directions in each dimension). Every cell gets one input from its northern neighbor in a finite space (torroidal wrapping makes cyclic permutation); similarly, for south, 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 $N \times M$ 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.

By examples in Figures 5c and 5b, 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.

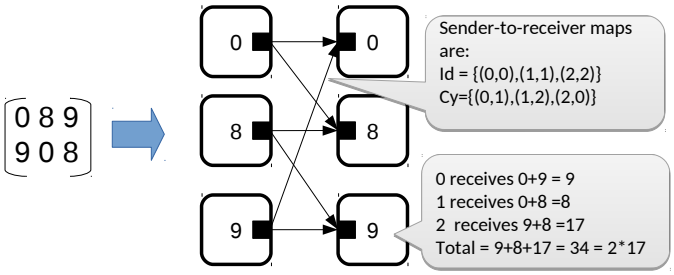
[11] Contrarily, *approximate morphisms* introduce errors at each transition of the lumped network (cf. Figure 17).



(a) 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.



(b) Connectivity example of components in a network with identity mapping Id and non-cyclic permutation mapping $Perm$.



(c) Connectivity example of components in a network with identity mapping Id and cyclic permutation mapping $Perm$.

Figure 5: Network connectivity and state representations.

In deterministic couplings, we are interested in maps. To get no error all-to-all is sufficient but not necessary. *Identity or permutation coupling maps are sufficient* (because they lead to a full sampling).

In conclusion, based on both dynamics and structure homogeneous conditions, *applying the lumping method steps to exact morphisms* consists of:

1. Specifying both transition and output functions of the base network of systems (using Definition 6) $\eta = (\delta, \lambda, D, \{S_i\}, \{Z_i\})$: All transition/output functions of components are the same, 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. Couplings $\{Z_i\}$ in the network are permutation-based (*cf.* Theorem 1 after).

2. Specifying both transition and output functions of the lumped network (using Definition 7) $\eta' = (\delta', \lambda', D', \{S'_i\}, \{Z'_i\})$: At each transition of the vector of states of the components of the base network, a corresponding state of the lumped network should match (cf. Appendix A), based on the congruence relation induced by the morphism over the state set of the base network.

3. Specifying the morphism (g, h) (using Definition 5 and with no output mapping for simplification): $g(..., \omega_i, ...)$ is the input mapping 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 with q_i the *component states* in the *base network*. Finally, the lumped transition function thus consisting of $\delta'(q', \omega') = \delta'(h(\times_i q), g(\times_i \omega_i))$.

The general proof of existence of an homomorphism between a base network of a lumped network of general systems has been proposed in (Zeigler, Muzy, and Kofman 2018) (page 420):

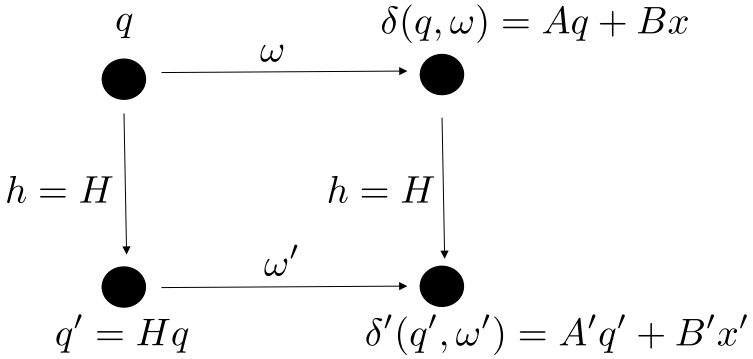
Theorem 1. *(adapted from (Zeigler, Muzy, and Kofman 2018)) Let there be a partition of systems of base network for which: 1) blocks (cf. Appendix A for a description of blocks) have homogeneous systems, 2) the couplings between systems in blocks is permutation invariant, and 3) the base network output is computable from block network census counting (cf. Section 6.2). Then a lumped network exists for which there is a homomorphism.*

As described in Section 6.2, the census counting is a linear operation, which holds for linear systems (*cf.* Definition 4). Therefore, this theorem remains true for linear systems. Besides, as we will see, the permutation-based coupling requirement is relaxed here to all-to-all, one-to-one, and many-to-one couplings.

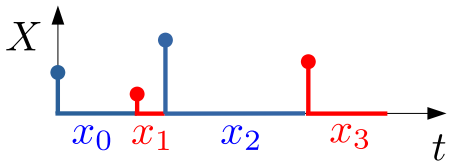
3.3 Matrix expression of linear network homomorphisms

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 be fulfilled by such homomorphism to hold for system networks.

Figure 6a depicts a general system morphism. For simplicity, the duration of input segments and corresponding transitions are not indicated, as well as output functions and output segments. Nevertheless, each input segment $\omega \in \Omega$ has a length $l(\omega) = t$, i.e., $\omega = \omega_{t>}$ and $\omega' = \omega'_{t>}$. Considering networks of linear systems, this means that the transition function of the big system consists of $\delta(q, \omega) = qe^{Al(\omega)} + \int_0^{l(\omega)} e^{A(l(\omega)-\tau)} B\omega(\tau)d\tau$. Corresponding matrix expression consists of $\delta(q, \omega) = Aq + Bx$, with ω defined over an *interval* $< 0, t >$, q a vector of *states* and x a *vector of input values*. Figure 6b depicts an example of simple discrete event input trajectory composed event segments. To each segment interval corresponds a single value. Notice that following theorems concern particular segments ω, ω'. However, if theorems are true for segments, it can be shown that they are true for any composition of these segments and any general systems (Zeigler, Muzy, and Kofman 2018).



(a) Commutative diagram of the matrix representation of a linear system morphism.



(b) Example of composition of discrete event input segments of different lengths.

Figure 6: Morphism between from a big general system SYS to a small one, SYS', indicating linear matrix correspondence and input segments.

The following theorem provides the conditions to be fulfilled by a homomorphism to hold for linear system networks with I/O. As shown in Figure 6b, system segments are defined over intervals while matrices have

values corresponding to such intervals. As far as we know there is no such general proof of existence of morphism between I/O linear system networks, for any segments of different lengths.

Theorem 2. *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 network of linear systems to a small network of linear systems, if the following conditions hold (cf. Figure 3):*

1. *$A'H = HA$, 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" but here we are taking $X' = X$ and the input encoding is the identity for simplicity,*
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" but here we are taking $Y' = Y$ and the output encoding is the identity for simplicity.*

Proof. We will prove that linear homomorphic conditions are *sufficient* and *necessary*. Proving the *sufficiency* of homomorphic conditions consists of proving that if they are true for the specific matrix expression, they are true for corresponding general system. Conversely, the *necessity* of homomorphic conditions consists of proving that if they are true for any general system, they are true for the matrix expression.

Proving that a morphism (*cf.* Definition 5) holds for general systems, consists of two steps:

1. For all $q \in Q$ and accounting that $x' = x$ and $\omega = \omega'$, for all $\omega \in \Omega$, $\delta'(h(q), \omega) = h(\delta(q, \omega))$, and
2. For all $q \in Q$, $\lambda'(h(q)) = \lambda(q)$.

We will show now the sufficiency of homomorphism conditions 1-3. Considering the commutative diagram of Figure 6a: $\begin{cases} h(\delta(q,\omega)) &= HAq + HBx \\ \delta'(h(q),\omega') &= A'q' + B'x' = A'Hq + B'x \end{cases}$

So if $HA = A'H$ and $HB = B'$, then $\delta'(h(q), \omega') = h(\delta(q, \omega))$. For output transitions, $\lambda'(h(q)) = C'h(q) = C'Hq$. Thus, if $C'H = C$ then $\lambda(q) = Cq = \lambda'(h(q))$.

To prove the *necessity* of homomorphism conditions 1-3 for any inputs and states, we consider that $\delta'(h(q), \omega') = h(\delta(q, \omega))$, which in matrix form consists of $HAq + HBx = A'Hq + B'x$. To prove conditions 1 and 2, let consider two separate cases:

1. **If the state is zero**, *then* $\cancel{HAq} + HBx = \cancel{A'Hq} + B'x$ *and* $HB = B'$, for all $x \in X$.
2. **If the input is zero**, *then* $HAq + \cancel{HBx} = A'Hq + \cancel{B'x}$ *and* $HA = A'H$ for all $q \in Q$.

Similarly, to prove condition 3, it can be proved that $C'H = C$, for all $q \in Q$.

□

Example 1. Homomorphism matrix form H from a big to small network of 1-D linear components (cf. Example 3).

The matrix form of the linear operators consists of $A = \begin{bmatrix} a & 0 \\ 0 & a \end{bmatrix}$, $Q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$, $B = \begin{bmatrix} b & 0 \\ 0 & b \end{bmatrix}$, $X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, $C = \begin{bmatrix} c \\ c \end{bmatrix}$ and $Y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$. Taking homomorphism $H = \begin{bmatrix} 1 & 1 \end{bmatrix}$, $A' = [a]$ and applying state linear mapping conditi‘on, 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, $HB = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} b & 0 \\ 0 & b \end{bmatrix} = \begin{bmatrix} b & b \end{bmatrix} = B'$. 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: $A' = [a]$, $B' = \begin{bmatrix} b & b \end{bmatrix}$, and $C' = [c]$.

Figure 7 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.

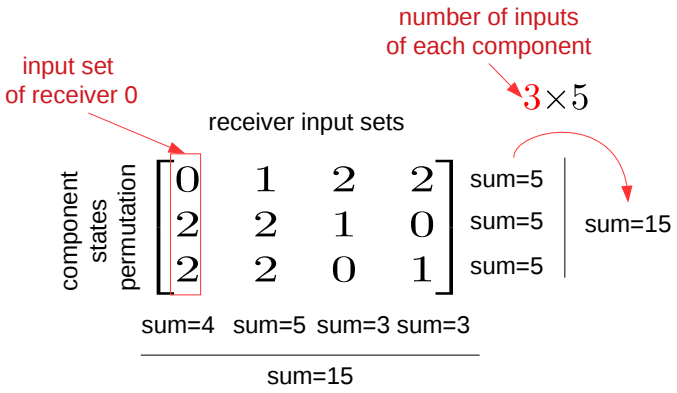


Figure 7: Matrix representation of a connectivity permutation: Each cell consists of a component state with values $\{0, 1, 2\}$, each column of the inputs received from a sender and each row consists of the global state of the network composed of four components (s_1, s_2, s_3, s_4). The first row represents the identity mapping in which each component receives its own state as input.

4 Lumping networks based on particular mean field conditions

4.1 Mean field based morphism

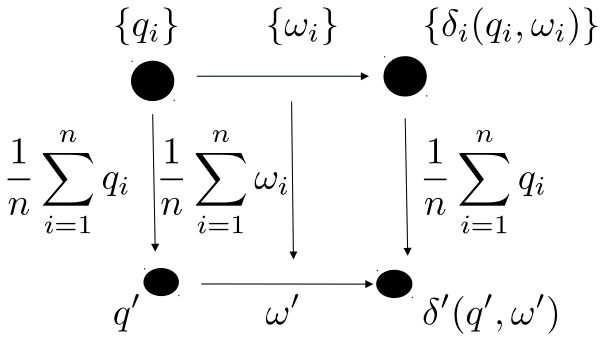
As introduced in Section 2, many mean field conditions exist according to the models explored. We choose here two particular/significant conditions.

Definition 8. The mean field conditions investigated here consist of *two conditions* for abstracting linear networks:

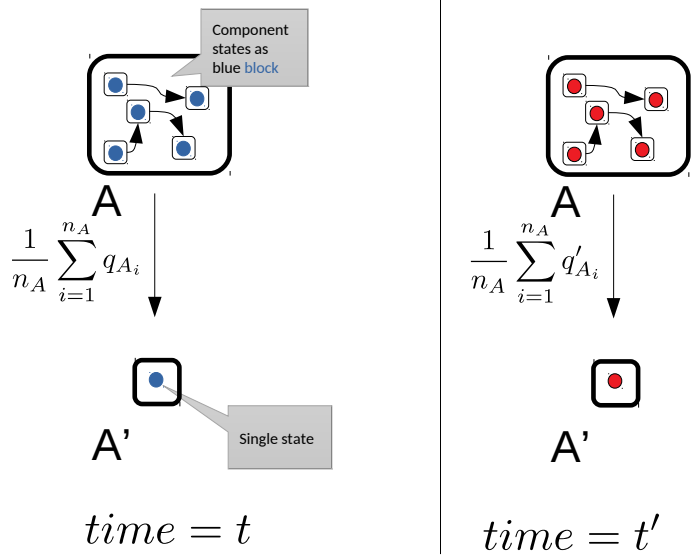
1. Homogeneity: All the components of the network have the same state transition/output functions and parameters and the network connectivity is permutation-based,

2. Fixed-point: The transition function of the network resultant system admits a fixed-point.

Mean field morphism between a base network model and a lumped network model is presented in Figure 8a.



(a) Commutative diagram of mean field network morphism.



(b) Base and lumped model transitions at times t and t' based on a homomorphism h between a base network of components A and a lumped network of components A'. At time t, all the states of components of network A correspond to blue block. At next time t', all the states of components of network A go to block red block.

Figure 8: State transition and lumping of a network of components based on mean field conditions.

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 8), the lumping method steps described in Section 3.2.2 can be applied considering: (1) A base network of linear systems (using Definition 6) $\eta_{linear} = (\delta_{linear}, \lambda_{linear}, \{S_{linear_i}\})$, with homogeneous dynamics and structure conditions, (2) A lumped network of linear systems (using Definition 7) $\eta'_{linear} = (\delta'_{linear}, \lambda'_{linear}, \{S'_{linear}\})$, where transition/output functions of components in the base network are the same than in the lumped network, $\delta' \stackrel{def}{=} \delta_i$, 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 (3) Specifying 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 4) 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 4) with q_i the *component states* in the *base network*.

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 in the detailed network. 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. This sampling is based on a permutation of the global states of the network.

Example 2. Let's detail the state lumping achieved in a network of components A (cf. Figure 8a). At each transition, the vector of component states of the base network A corresponds to a block of states. At each transition, all the component states can change to another block or remain in the same block. Following the commutative diagrams, the homomorphism requirement for all states $q_A \in Q_A$ is $\delta'(h(q_A)) = h(\delta(q_A))$, for $h(q_A) = \frac{1}{n_A}\sum_{i=1}^{n_A} q_{A_i}$, i.e., taking the sum of the states of the n_A components in network A. Each block (or vector) of component states thus corresponds to the average of component states (or a lumped state). At each time, the state of the lumped network A', $q'_A \in Q'_A$ should match the vector of states of the base network A, $q_A \in Q_A$.We will discuss in the next sections the properties and examples of such linear networks.

4.2 Lumping a network based on homogeneous conditions

When applied to a network, a morphism is called a *network 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 3. A network of two 1-D linear components with identical structure (following homogeneity condition of mean field theory (cf. Definition 8)) 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 fixed-point condition of mean field theory (cf. Definition 8), 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 fixed-point conditions, different structures lead to the same lumped network dynamics.

Remark 4. This simple example shows that $Q = 0$ is the fixed point of the lumped network and its stability depends on the sign of parameters $a + bc$: if $a + bc > 0$, the system grows exponentially, thus being unstable, while for $a + bc < 0$, the system is stable. Following Remark 5, analysis shows that the base model can be unstable with $a > 0$, while sufficiently positive feedback (i.e., $bc > 0$) can result in a stable lumped model.

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 the homomorphism requirement holds for all-to-all, one-to-one, and many-to-one couplings. Also, in the computational context, the number n of components needs not be infinite.

4.3 Fixed point morphism preservation

Theorem 3. *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 $q \in Q$, $\omega' \in \Omega'$,$h(\delta(q, g(\omega))) = \delta'(h(q), \omega')$,
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 $\omega' = g(\omega)$.

□

Remark 5. While fixed points are preserved from base to lumped models (i.e., downward), the reverse direction, (i.e. upward) is more problematic (Sierocki 1986). (Foo 1977) showed that for bounded linear systems upward preservation does indeed hold.

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

Theorem 4. *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 3, 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 4, 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:

$$
\begin{array}{rcll}
\omega & = & \lambda(\delta(q, \omega)) & \\
& = & \lambda(..., \delta_i(q_i, \omega_i), ...) & \text{based on base network structure (cf. Definition 6)} \\
& = & \lambda'(\frac{1}{n}\sum_{i=1}^{n} \delta_i(q_i, \omega_i)) & \text{based on lumped network structure (cf. Definition 7)} \\
& = & \lambda'(\delta_i(\frac{1}{n}\sum_{i=1}^{n} q_i, \frac{1}{n}\sum_{i=1}^{n} \omega_i)) & \text{based on linear system properties (cf. Definition 4)} \\
& = & \lambda'(\delta'(q', \omega')) &
\end{array}
$$

□

Remark 6. 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 Lumping connected networks

5.1 State transition matching between base and lumped networks

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 in the detailed network. 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 the couplings between linear components. This sampling is based on a permutation of the global states of the network.

Example 4. Let's detail the state lumping achieved in a network of networks A and B (cf. Figure 9). At each transition, the vector of component states of a base network, A or B, corresponds to a block of states. At each transition, all the component states can change to another block (e.g., for network A) or remain in the same block (e.g., for network B). Following the 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 $h(q_A, q_B) = (\frac{1}{n_A}\sum_{i=1}^{n_A} q_{A_i}, \frac{1}{n_B}\sum_{i=1}^{n_B} q_{B_i})$, i.e., taking the sum of the states of the n_A (resp. n_B) components in network A (resp. B). Each block (or vector) of component states thus corresponds to the sum of component states (or a lumped state). 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 computed 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.

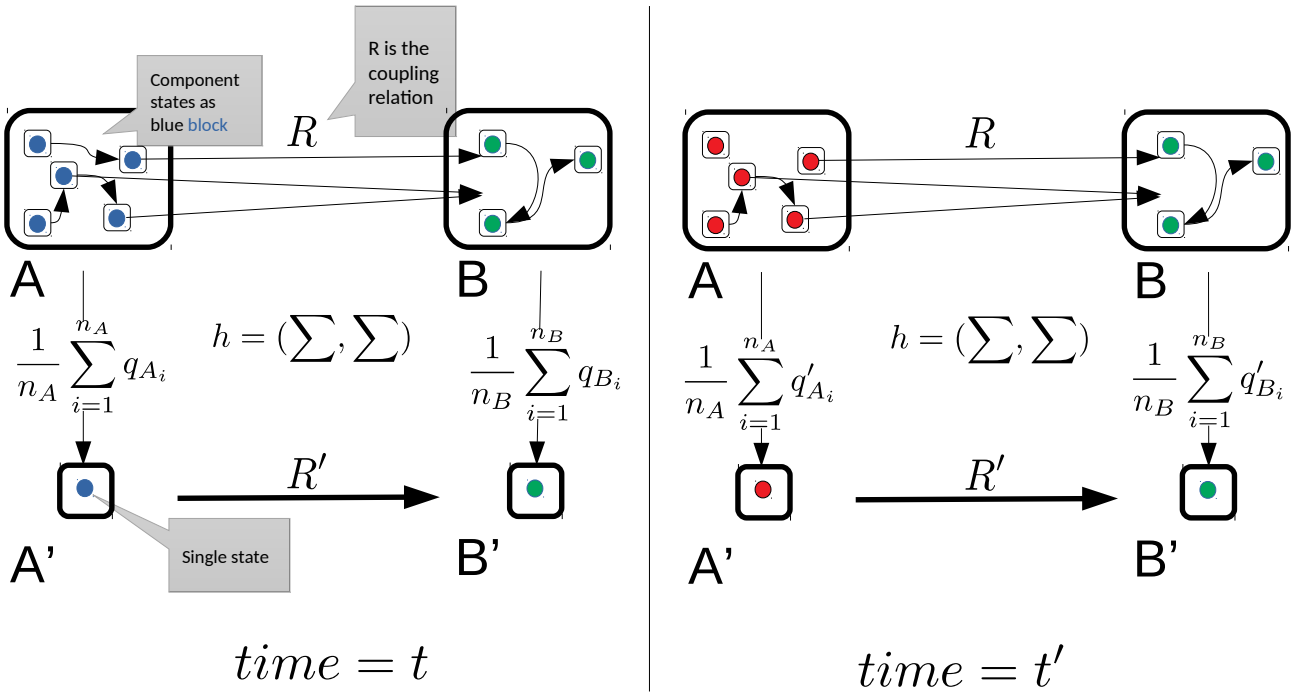


Figure 9: Base and lumped model transitions at times t and t' based on a homomorphism h between a base network of networks A and B and a lumped network of networks A' and B'. In the base network (resp. lumped network), components in network A (resp. A') influence components in network B (resp. B'). At the base and lumped levels, relations R and R' shows the couplings between the components. At time t, all the states of components of network A correspond to blue block and all the states of components of network B correspond to block green block. At time t', all the states of components of network A go to block red block and all the states of components of network B remain in block green block .

5.2 Lumping large coupled networks into small coupled networks

Figure 10 shows the morphism between two base networks and two lumped networks.

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

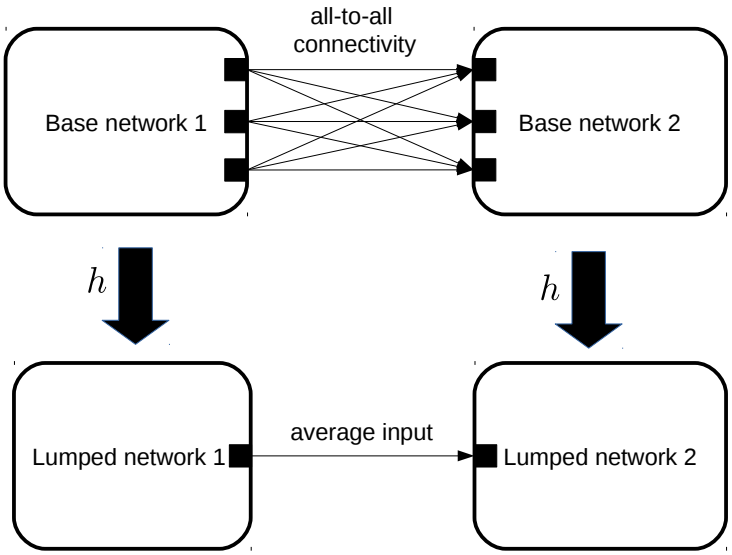
Theorem 5. *The set of coupled base networks of linear systems is closed under morphism.*

Proof. Figure 10b 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:

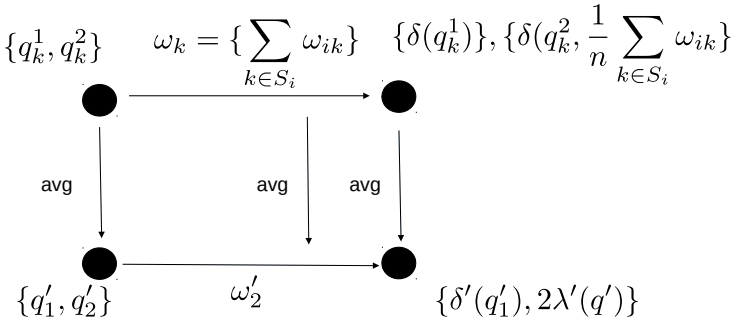
$$\begin{aligned}
\delta'(q'_1), \delta'(q'_2, \omega'_2) &= \delta'(\tfrac{1}{n}\textstyle\sum_{k=1}^{n} q_k^1), \delta'(\tfrac{1}{n}\sum_{k=1}^{n} q_k^2, \tfrac{1}{n}\sum_{k=1}^{n} \omega_k) \\
&= \delta'(\tfrac{1}{n}\textstyle\sum_{k=1}^{n} q_k^1), \delta'(\tfrac{1}{n}\sum_{k=1}^{n} q_k^2, \tfrac{1}{n}\sum_{k=1}^{n}\sum_{k\in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

□

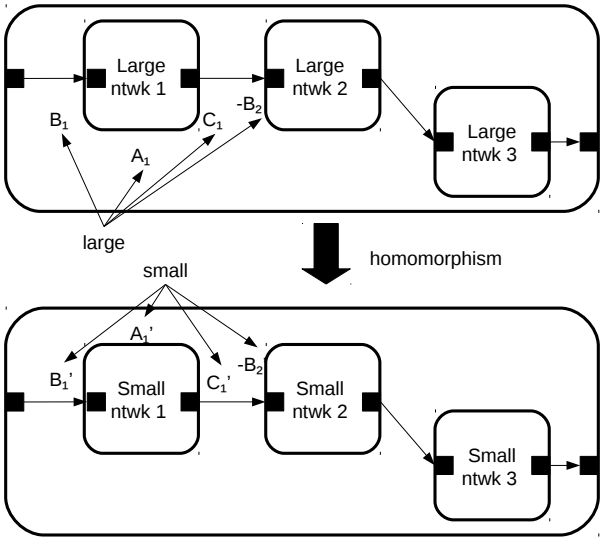
Figure 10c summarizes the large to small correspondence between the structures of base and lumped networks of networks.



(a) Morphism between two base networks and two lumped networks, an example of all-to-all coupling between the base networks.



(b) Commutative diagram of the morphism between two base networks and two lumped networks.



(c) 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.

Figure 10: Coupling and lumping of networks from large to small.

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

5.3 Simple discrete formulation

Figure 11 describes the lumping of two connected pools of components with no internal couplings. The state change from time t to time t' (notice that t' is not necessarily discrete thus equal to $t+1$ reflecting the possibly continuous-time nature of systems) of a neuron $i = 1, ..., n$ consists of

$$q_i(t') = q_i(t) + I_i(t)$$

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:

$$Q(t') = Q(t) + I(t)$$

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

$$Q^2(t') = Q^2(t) + Q^1(t)$$

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

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

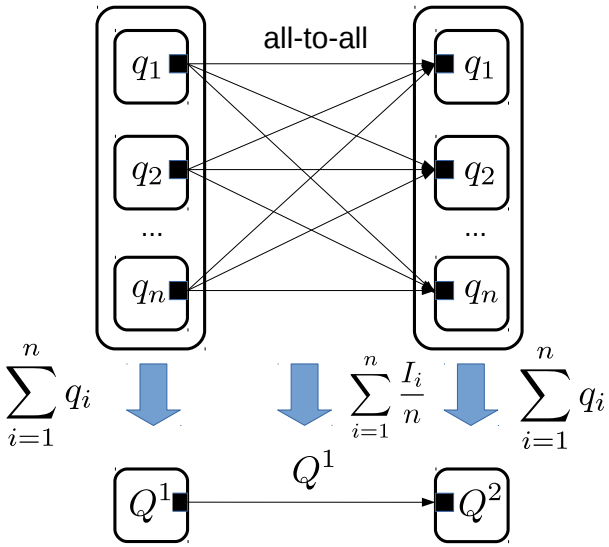


Figure 11: Simple discrete formulation of the lumping of two fully coupled networks.

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 $q_i(t') = q_i(t) + I_i(t) = q_i(t) + \sum_{i=1}^{n} \frac{Q_i(t)}{n}$. Applying h to both $\{q_i(t)\}$ and $\{q_i(t')\}$, we have $Q(t')$ and $Q(t) + \sum_{i=1}^{n} \sum_{i=1}^{n} \frac{Q_i(t)}{n} = 2Q(t)$ which match the lumped model transition $Q(t') = 2Q(t)$.

The solution of the lumped model is

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

Remark 7. In neuronal pools, averaging the inputs can be conceived as averaging the firing rates of neurons when taking their inputs as firing rates.

5.4 Example of Cascade networks

A dedicated software for setting base and lumped networks of linear systems has been developed using MS4Me IDE (Seo et al. 2013). The software allows connecting and modeling networks in a modular way as well as observing the trajectories of the components of the base networks and corresponding lumped networks. Figure 12a 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 12b 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. 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 "In1" consists of a_1. The transition function components consists after using coupling of:

$$\begin{cases} q_{first}(ID0)(t') = a_0 q_{first}(ID2)(t) + a_1 q_{first}(ID1)(t) \\ q_{first}(ID1)(t') = a_0 q_{first}(ID0)(t) + a_1 q_{first}(ID2)(t) \\ q_{first}(ID2)(t') = a_0 q_{first}(ID1)(t) + a_1 q_{first}(ID0)(t) \end{cases}$$

Where, e.g., $q_{first}(ID0)(t')$ notation represents the state of the first component "*ID0*", in the *first* network, at time t'.

- Corresponding first lumped network:

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

With:

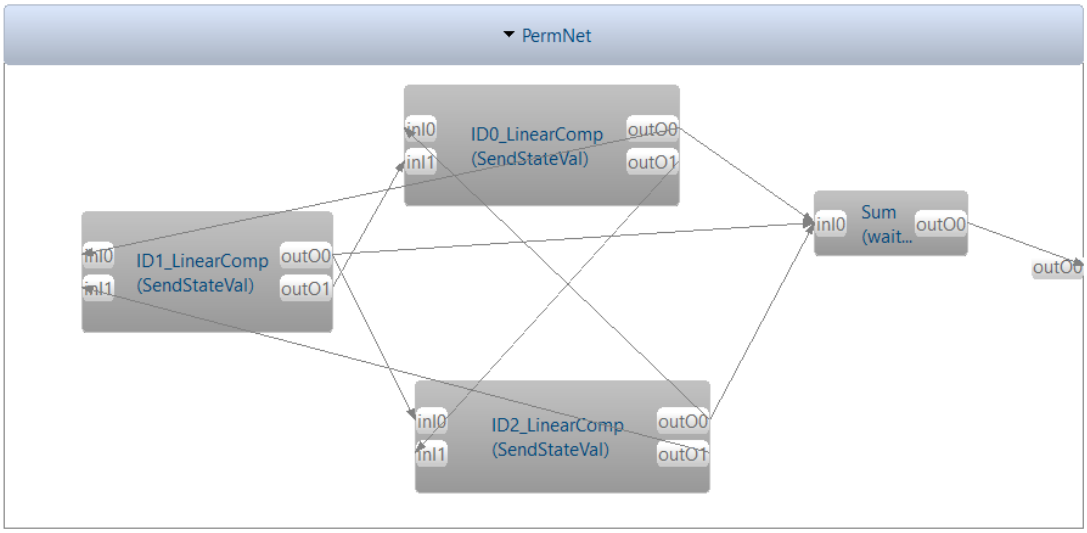
$$\begin{array}{rcl} Q_{First}(t') & = & q_{first}(ID0)(t') + q_{first}(ID1)(t') + q_{first}(ID2)(t') \\ & = & a_0 Q_{First}(t') + a_1 Q_{First}(t') \\ & = & (a_0 + a_1) Q_{First}(t') \end{array} ,$$

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

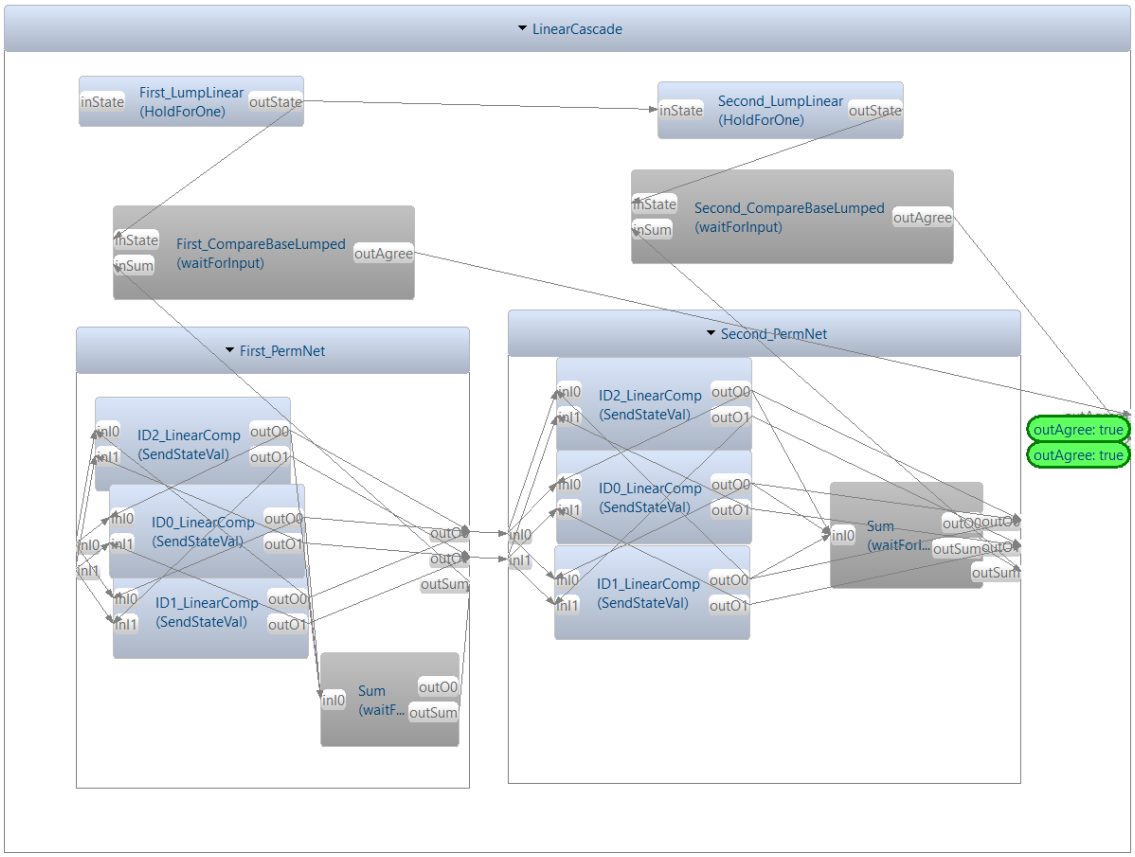
- First Base Component to Second Base Component Coupling is all-to-all for both couplings:

$$\begin{cases} q_{second}(ID0)(t') = a_0 q_{second}(ID2)(t) + a_1 q_{second}(ID1)(t) + a_0 Q_{First} + a_1 Q_{First} \\ q_{second}(ID1)(t') = a_0 q_{second}(ID0)(t) + a1 q_{second}(ID2)(t) + a_0 Q_{First} + a_1 Q_{First} \\ q_{second}(ID2)(t') = a_0 q_{second}(ID1)(t) + a_1 q_{second}(ID0)(t) + a_0 Q_{First} + a1 Q_{First} \end{cases}$$

- Corresponding second lumped network: $Q_{Second}(t') = a(Q_{Second}(t) + bI(t))$, with $I(t) = bQ_{First}$ and h parameter mapping: $a = (a_0 + a_1)$, $b = 3$. Then, $Q_{Second}(t') = (a_0 + a_1)(Q_{Second}(t) + 3 * Q_{First}(t))$ and $Q_{Second}(t') = (a_0 + a_1)Q_{Second}(t) + 3 * (a_0 + a_1)Q_{First}(t)$.



(a) 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.



(b) 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.

Figure 12: Network structure and coupling of networks using the software developed based on MS4Me IDE (Seo et al. 2013).

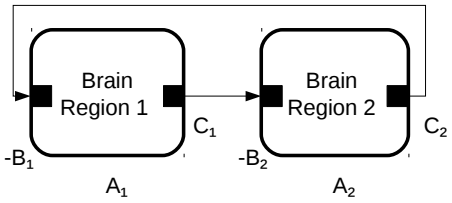
6 Application to neuronal networks

6.1 Amari-Wilson-Cowan network model

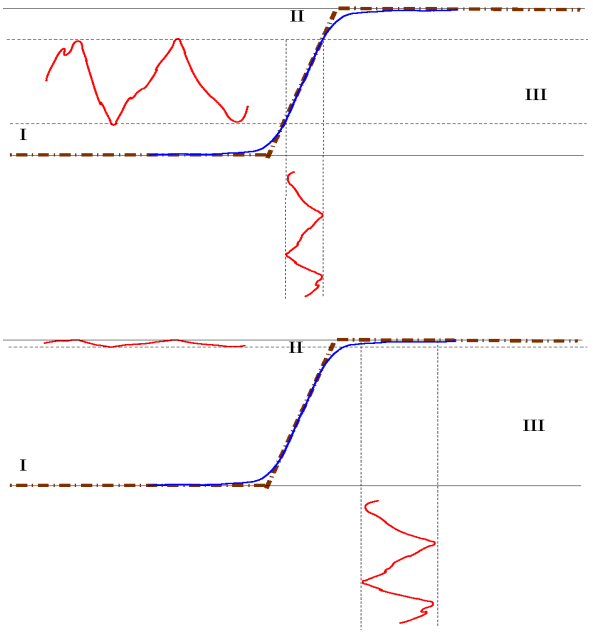
A well known model of neuron dynamics is the the Amari-Wilson-Cowan (AWC) model (Amari 1972; Wilson and Cowan 1972; Wilson and Cowan 1973), slightly modified by taking external input voltage to the network as external inputs from another network:

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

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 13b). "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).



(a) 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 (and conversely $C_2 - B_1$ from region 2 to region 1).



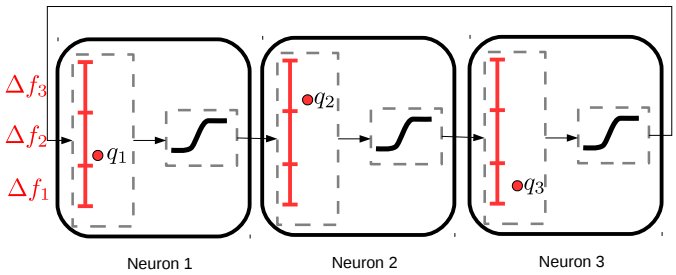
(b) The sigmoidal shape of the function f and its effects on voltage fluctuations. Top: When neurons voltage fluctuates around the inflection point of the sigmoid, and if the gain g is large enough fluctuations are amplified. Bottom: When the neurons voltage fluctuates in the flat parts of the sigmoid (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).

Figure 13: Simulation of brain regions based on lumping simulatability, matrix equivalence and linear transition function of neurons.

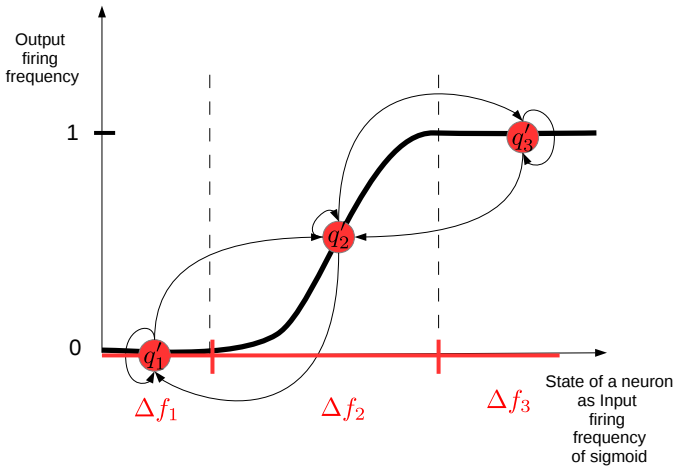
Parameters of Equation 2 consist of: c is the *signal attenuation on the axon*, b is the *pulse attenuation on synapses and dendrites*.

6.2 Census-based lumping

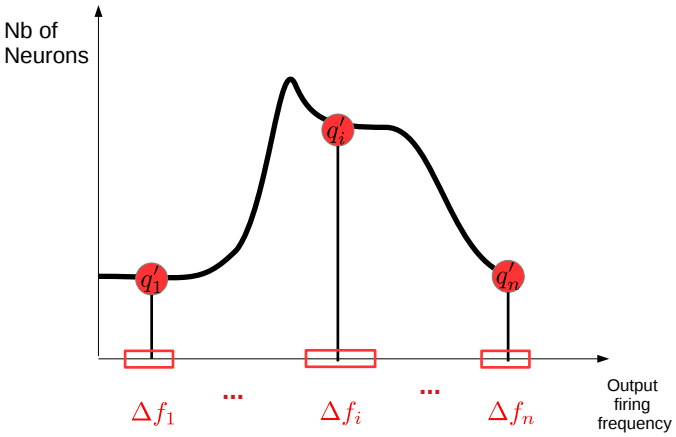
The census-based lumping approach (Zeigler, Muzy, and Kofman 2018) is based on the counting of the different states the components of a network can be found in. Figure 14a shows an example of 3 neurons with corresponding states q_1, q_2, q_3. Each of these states can be in a census cell range. For these continuous states, a census cell is an interval Δf_i, where the intervals are disjoint and together cover the range of firing frequencies. For discrete states, a census cell is simply a state. In Figure 14b, the sigmoid function of base neuron models is shown, conceived as a transfer function where a neuron receives an input frequency and outputs a frequency. Considering Figure 13b the problem can be reduced to a single region consisting of neurons with sigmoid output functions and a matrix of weights, A, that linearly transforms the outputs back to the inputs. The problem then becomes: how to reflect the combined feedback transform in the lumped model. Let's consider now that to each region I, II, III of the sigmoid as described in Figure 13b, corresponds a census cell Δf_1, Δf_2, and Δf_3 and a lumped model census vector (q'_1, q'_2, q'_3) representing the count of base model neurons in the respective census cells. There is a homomorphism if for each census cell, all neurons in a cell behaves the same, i.e., go to the same census cell under the action of the weight matrix A. So all base neuron states corresponding to a lumped state in $\{q'_1, q'_2, q'_3\}$ can all remain in the same state or all go to the same next state (cf. transition arrows). As we saw in the previous counter example, neurons in the same region can behave differently (not based on a sigmoid global function) even with linear feedback and sigmoid, when there is only one census cell. This is the case when one cell, Δf_1 becomes the whole firing range as is true when the average of firing frequencies (i.e., effectively the sum) is used for the putative homomorphic mapping. We have shown that under homogeneity, the homomorphism is exact when all neurons are in region II and remain in this region after the combined feedback transformation. This is why the census approach can be used to count the number of neurons in each cell and to attempt to track the cell to cell transitions through regions I, II, and III. The morphism is then approximate, neurons being in the same state being able to go to two different states because of census cell sizes thus breaking basic homomorphism requirement (cf. Appendix A). In this approach, the sizes of the census cells represent the resolution of the homomorphic mapping. Having enough cells it is maybe expected that the approximate morphism turns to be exact. The distribution of neuron states can then be described (cf. Figure 14c). For an infinite number of neurons and infinitely small cells, continuous distributions are obtained. The question then is to prove that the lumped model respects the state distribution at each transition (Hartmanis 1966). Notice that the distribution is not necessarily preserved but it must be "respected", i.e., two states in the same interval must go to the same (possibly other) interval. Figure 14c shows the difference between a base model using the linear part C of the sigmoid and a base model using the sigmoid (or its linear approximation). In both models, the feedback relies on the weights B. However, lumping the linear model based on mean field assumption preserves the average firing rate and can be predicted as depicted above, considering the in/decreasing factor of the linear slope. Note that in the linear case, a single census cell covering the whole range of neuron states is sufficient. In this case, all neurons remain in the same cell under any transition with the sum (or average) of their states in the cell employed in the homomorphism. Using the (linear approximation of the) sigmoid C', all the neuron states (e.g., not firing (corresponding to the 0 output of the sigmoid, saturating (corresponding to the 1 output of the sigmoid), or in between (corresponding to the $(0, 1)$ region output of the sigmoid)) are shuffled from time t to $t + 1$. Also, a minimum of three census cells, $\Delta f_1, \Delta f_2, \Delta f_3$, is necessary. Notice that for the morphism to hold each block of neuron states (a vector of states) at time t should go to the same (possibly other) block at time $t + 1$ (cf. Appendix A). Two blocks can go to the same other block, etc. However, this distribution mechanism needs to be proven. *Characterizing such approximate homomorphisms is left for future research.* However, it shows that the census based approach can be used as a simple method to lump constructively networks of (linear approximation) of sigmoid based neurons.



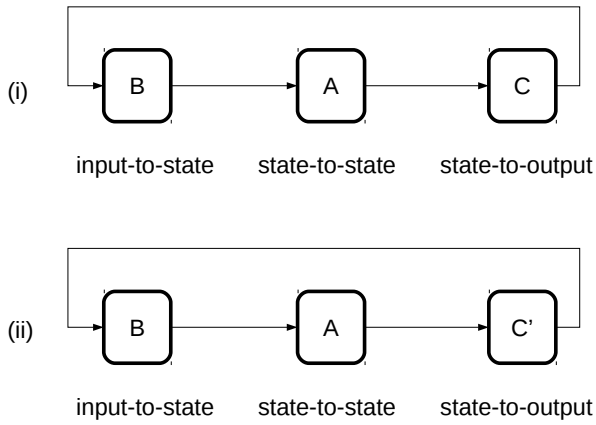
(a) Example where each neuron gets its state belonging to one of the three census cells $\Delta f_1, \Delta f_2, \Delta f_3$.



(b) Sigmoid at lumped level, where each state q'_1, q'_2, q'_3 on the sigmoid represents a set of base neurons in states belonging to corresponding census cell.



(c) State distribution of neurons at time t.



(d) Difference between linear part of the sigmoid (i) and (linear approximation of the) sigmoid (ii) feedback consists just in changing matrix C (linear function) to C' (sigmoid).

Figure 14: Census based approach.

7 Conclusion and perspectives

This article proves the existence of morphisms for lumping finite-size networks of linear systems. Morphisms proved to constitute a constructive and general method for lumping such systems. Using permutation-based and basic mean field conditions, no error is introduced at each transition using exact morphisms. Simple discrete formulations have been defined and can be used to model linear neurons. Thanks to a flexible definition of input/output segments, piecewise linear systems (Coombes et al. 2018) can be simulated efficiently by discrete events scheduling the duration of piecewise linear segments (although predicting the length of such segments is a a difficult modeling question). The framework can be used to study further hybrid systems allowing discontinuous inputs for piecewise linear systems. Also we showed why basic linear mean field assumptions cannot be used in some cases to lump sigmoid-based neurons. Also, non-homogenous conditions of component parameters and connectivity have been introduced in Appendix B through approximate morphisms. At each step the error showed to be computationable. Finally, the census-based approach has been discussed as a solid perspective to account for non-linear models of neurons through approximate morphisms.

This whole approach provides a computational method for abstracting finite-size networks of general linear

systems. Several direct different application areas should benefit from these results. In electronics, neuromorphic networks (Mayr, Hoeppner, and Furber 2019) and Binary Neural Networks (BNN) (Courbariaux, Bengio, and David 2015) can be studied using our approach. For neuromorphic networks, lumping networks leads to reducing the number of electronic neurons and computations, so to more efficient solutions while preserving the network behavior. BNNs are easier to model because they only depend on step heaviside transition functions with no memory (or previous state dependence). Here again, reducing the number of artificial neurons (for the same learning capabilities and less computations) is a strong advantage. In cognitive neurosciences, the permutation-based connectivity shown here can be investigated to reduce the complexity of the experimental evidence of information computation by neural cliques (Xie et al. 2016; Tsien 2015). More recently, such evidences has been shown to apply to neuromorphic circuits based on a multilinear logics (Selesnick 2019).

The main formal perspective consists of investigating deeper approximate morphisms (Zeigler, Muzy, and Kofman 2018) and lumpability (Atay et al. 2016) based on the computational error induced at each transition, using a probabilistic modeling approach, e.g., for neural networks (El Boustani and Destexhe 2009; Zeigler 1975). In neurosciences, only a finite number of electrodes are used thus constituting an interesting application perspective for this new approach. Furthermore, deterministic models showed to be able to reproduce *in vitro* experiments (Kass et al. 2018). These models are based on Leaky Integrate and Fire (LIF) models that can be investigated based on general system theory (Muzy, Zeigler, and Grammont 2017) and thus again morphisms. Concerning Deep Neural Networks (DNN) (LeCun, Bengio, and Hinton 2015), previous section discusses the census-based approach for modeling usual sigmoid functions. It will then be worth investigating the abstraction of memory-based DNNs engaged in non feed forward structures (as for example permutation based network structures) (Paugam-Moisy and Bohte 2012). There is also a need for approximate morphisms to support the mapping of Recurrent Neural Nets (RNN) into Spiking Neural Nets (SNN) to run on low power neuromorphic chips (Diehl et al. 2016).

Acknowledgments

We would like to thank Bruno Cessac, for very constructive comments to improve the quality of this article concerning mean field theory and linear systems. Also we thank Andy Barto for his comments on morphisms, which helped a lot to improve the understandability of the theory.

References

Amari, S-I (1971). "Characteristics of randomly connected threshold-element networks and network systems". In: *Proceedings of the IEEE* 59.1, pp. 35–47.

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*.

Atay, Fatihcan M et al. (2016). "Perspectives on Multi-Level Dynamics". In: *Discontinuity, Nonlinearity, and Complexity* 5.3, pp. 313–339.

Ben Arous, G and O Zeitouni (1999). "Increasing propagation of chaos for mean field models". In: *Annales de l'IHP Probabilités et statistiques*. Vol. 35. 1, pp. 85–102.

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.

Coombes, Stephen et al. (2018). "Networks of piecewise linear neural mass models". In: *European Journal of Applied Mathematics* 29.5, pp. 869–890.

Courbariaux, Matthieu, Yoshua Bengio, and Jean-Pierre David (2015). "Binaryconnect: Training deep neural networks with binary weights during propagations". In: *Advances in neural information processing systems*, pp. 3123–3131.

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

Diehl, Peter U et al. (2016). "Conversion of artificial recurrent neural networks to spiking neural networks for low-power neuromorphic hardware". In: *2016 IEEE International Conference on Rebooting Computing (ICRC)*. IEEE, pp. 1–8.

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.

Foo, N . (1977). "Stability preservation under homomorphisms". In: *IEEE Trans SMC* 7, pp. 750–754.

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

Hartmanis, Juris (1966). *Algebraic Structure Theory of Sequential Machines (Prentice-Hall International Series in Applied Mathematics)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc. ISBN: B0006BNWTE.

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

Hong, Guosong and Charles M Lieber (2019). "Novel electrode technologies for neural recordings". In: *Nature Reviews Neuroscience* 20.6, pp. 330–345.

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.

Kass, Robert E et al. (2018). "Computational neuroscience: Mathematical and statistical perspectives". In: *Annual review of statistics and its application* 5, pp. 183–214.

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

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning". In: *nature* 521.7553, pp. 436–444.

Mauroy, Y Susuki and I Mezic (2020). *The Koopman Operator in Systems and Control*.

Mayr, Christian, Sebastian Hoeppner, and Steve Furber (2019). "SpiNNaker 2: A 10 Million Core Processor System for Brain Simulation and Machine Learning". In: *arXiv preprint arXiv:1911.02385*.

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

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

Muzy, Alexandre, Bernard P Zeigler, and Franck Grammont (2017). "Iterative specification as a modeling and simulation formalism for i/o general systems". In: *IEEE Systems Journal* 12.3, pp. 2982–2993.

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.

Paugam-Moisy, Hélene and Sander M Bohte (2012). "Computing with spiking neuron networks." In: *Handbook of natural computing* 1, pp. 1–47.

Selesnick, Stephen (Nov. 2019). "Tsien's Power-of-Two Law in a Neuromorphic Network Model Suitable for Artificial Intelligence". In: *IfCoLog Journal of Logics and their Applications* 6, pp. 1223 –1251.

Seo, Chungman et al. (2013). "DEVS modeling and simulation methodology with ms4me software". In: *Symposium on Theory of Modeling and Simulation-DEVS (TMS/DEVS)*. Vol. 25.

Sierocki, I. (1986). "A note on structural inference in systems theory". In: *Int J Gen SystÂ* 13, pp. 17–22. DOI: 10.1080/03081078608934951.

Sompolinsky, Haim, Andrea Crisanti, and Hans-Jurgen Sommers (1988). "Chaos in random neural networks". In: *Physical review letters* 61.3, p. 259.

Touboul, Jonathan, Geoffroy Hermann, and Olivier Faugeras (2012). "Noise-induced behaviors in neural mean field dynamics". In: *SIAM Journal on Applied Dynamical Systems* 11.1, pp. 49–81.

Tsien, Joe Z (2015). "A postulate on the brain's basic wiring logic". In: *Trends in neurosciences* 38.11, pp. 669–671.

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.

Xie, Kun et al. (2016). "Brain Computation Is Organized via Power-of-Two-Based Permutation Logic". In: *Frontiers in Systems Neuroscience* 10, p. 95. ISSN: 1662-5137. DOI: 10.3389/fnsys.2016.00095. URL: https://www.frontiersin.org/article/10.3389/fnsys.2016.00095.

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. (1975). "Statistical Simplification of Neural Nets". In: *International Journal of Man-Machine Studies* 7.3, pp. 371–393. DOI: 10.1016/S0020-7373(75)80018-6. URL: http://dx.doi.org/10.1016/S0020-7373(75)80018-6.

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

A Simple state homomorphism between two models

Figure 15 shows a homomorphism from a base autonomous model (δ), with $\delta : Q \to Q$, the *state transition*, Q the *state set* to a lumped model (δ'), with $\delta' : Q' \to Q'$, the *state transition*, Q' the *state set*. Between the two models, a homomorphism consist of a map $h : Q \to^{onto} Q'$, with the size of Q' being smaller than the one of Q. The map h induces a partition of the base state set Q such that $h : \{B_i\} \to Q'$, or:

- Q is partitioned into a set of subsets or *blocks*, with $B_1 \cup B_2 = Q$ and $B_1 \cap B_2 = \emptyset$, for any block B_i of the partition which does not contain empty set $\emptyset$,
- Equivalent classes consist of $[q] = \{q \mid q \sim q' \wedge q, q' \in Q\}$,
- The set of all equivalent classes is $Q/\sim = \{[q] \mid q \in Q\}$.

The equivalence relation is said congruent because it is on the algebraic structure (δ) that should be preserved by state transitions and considering the homomorphism h: $q_1 \sim q_2$ if $h(q_1) = h(q_2)$, for $q_1, q_2 \in Q$, which means that any congruent states in a block are mapped to the same state $q' \in Q'$ in the lumped model. Notice that each state transition in a block of the base model maps a state transition in the lumped model.

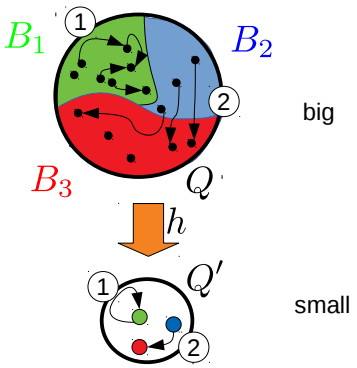


Figure 15: Partition of state set Q into three blocks B_1, B_2 and B_3. All the states in B_1 remain in the block after a transition ①. All the states in B_2 go to B_3 after transition ②. All the states B_3 remain the same after a transition. The lumped state set consists of $Q' = \{q'_1, q'_2, q'_3\}$, where each lumped state q'_i corresponds to a block B_i.

Example 5. Simple example of non-homomorphism

Figure 16 shows a simple example of two counters: one counting from 0 to 3 and the other from 0 to 1. The latter is the homomorphic image of the former. Notice a homomorphism of this type exists between any non-prime number and one of its factors.. Also, the choice between the base and the lumped model depends on the experimental frame or observation of the system. For example, the lumped model could be the minimal state set model for any even numbered counter (base model).

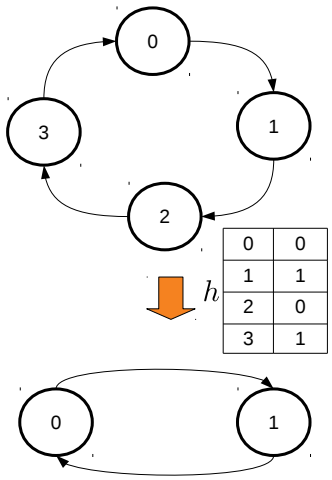


Figure 16: On the top a 4 counter is lumped into a 2 counter based on a map h. The latter induces a partition with blocks $B_1 = \{0, 2\}$ and $B_1 = \{1, 3\}$.

B Approximate model and simulation

B.1 Computational approximation, dynamics and stability

Figure 17 describes the error dynamics in the lumping model. Contrarily to exact morphisms, approximate morphisms, as we will see, can quantify the error introduced at each transition.

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 transition function,
2. Positive influence of predecessors, a_{ji},
3. Random sampling couplings instead of a uniform one. 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.

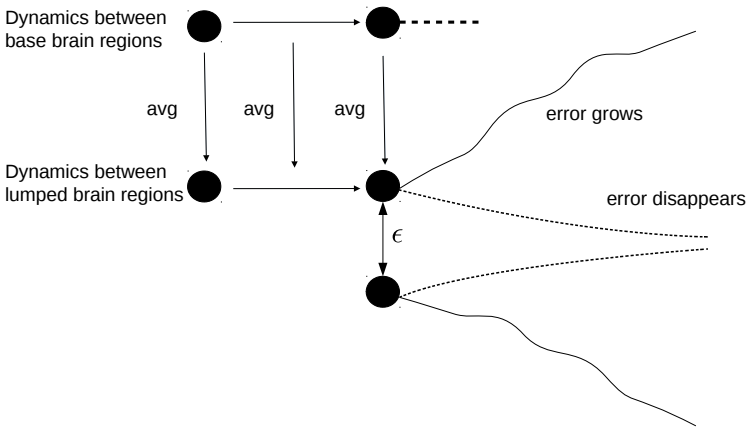


Figure 17: The error depends on the dynamics between base and lumped brain regions: Usually, the brain is a stable system where the error of the lumped model used for study should disappear. Sometimes, the brain turns unstable (as during epileptic crises).

B.2 Non-homogeneous components in a network

An example of **exact homomorphism** with two homogeneous components consists of:

$$\begin{cases} q(t') = aq(t) \\ r(t') = ar(t) \end{cases}$$

So $q(t') + r(t') \ = aq(t) + ar(t) = a(q(t) + r(t))$. If $z = q + r$ then $z(t') = az(t)$.
In formal terms, the *base model* transition function consists of:

$$\begin{cases} \delta : \mathbb{R} \times \mathbb{R} \to \mathbb{R} \times \mathbb{R} \\ with\ \delta(q, r) = (aq, ar) \end{cases}$$

The *lumped model* consists of:

$$\begin{cases} \delta' : \mathbb{R} \to \mathbb{R} \\ with\ \delta'(z) = (az) \end{cases}$$

The homomorphic map consists of:

$$\begin{cases} h : \mathbb{R} \times \mathbb{R} \to \mathbb{R} \\ with\ h(q, r) = q + r \end{cases}$$

The proof of homomorphism consists of two steps:

1. $\delta'(h(q, r)) = \delta'(q + r) = a(q + r) = aq + ar$, and
2. $h(\delta(q, r)) = h(aq, ar) = aq + qr = \delta'(h(q, r))$.

An example of **approximate homomorphism** with two non-homogeneous components:

$$\begin{cases} q(t') = aq(t) \\ r(t') = br(t) \end{cases}$$

The sum of both states consists of $q(t') + r(t') \ = aq(t) + br(t)$.
Let's define now parameters a and b as:

$$\begin{cases} a = avg + \alpha \\ b = avg + \beta \end{cases}$$

The sum now consists of $q(t') + r(t') = (avg + \alpha)q(t) + (avg + \beta)r(t) = avg \times (q(t) + r(t)) + \alpha q(t) + \beta r(t)$. Note that if $a = b$ then $\alpha = \beta = 0$ so if $z = q + r$ then,

$$z(t') = avg \times z(t) + error(t)$$

Where, $error(t) = \alpha q(t) + \beta r(t)$.

But note we do not know q and r having only z so we cannot compute the error knowing only the lumped model. This is the usual situation.

In formal terms, the *base model* transition function consists of:

$$\begin{cases} \delta : \mathbb{R} \times \mathbb{R} \to \mathbb{R} \times \mathbb{R} \\ with\ \delta(q, r) = (aq, br) \end{cases}$$

The *lumped model* consists of:

$$\begin{cases} \delta' : \mathbb{R} \to \mathbb{R} \\ with\ \delta'(z) = avg \times z \end{cases}$$

Theorem 6. *The homomorphic map $h : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is an* approximate homomorphism *$h(q, r) = q + r$.*

Proof. The proof of homomorphism consists of two steps:

1. $\delta'(h(q, r)) = \delta'(q + r) = avg \times (q + r) + error$
2. $h(\delta(q, r)) = h(aq, br) = aq + br \ = \delta'(h(q, r)) + error$

□

B.3 Non-homogeneous connectivity of a network

Based on the simple discrete example in Sub-section 5.3, 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 18 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.

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

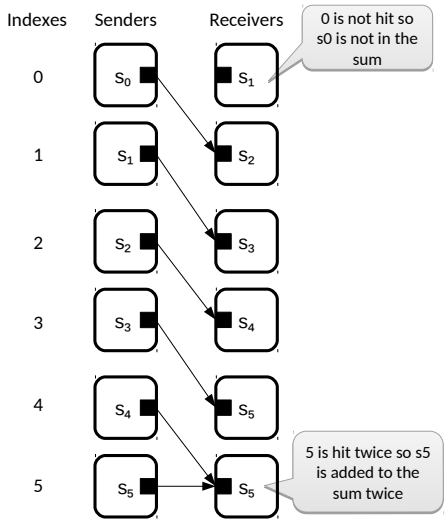


Figure 18: Error in sum due to permutation deficiency. The map D is not a permutation and consists of $D = \{(0,1),(1,2),(2,3),(3,4),(4,5),(5,5)\}$.

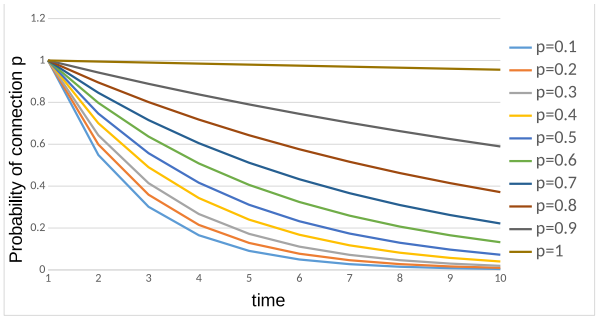
Let the component states, s_i, be either 0 or 1. Then, there is zero error when a missed component has a 0 value. Also a component that receives more than one input may contribute several ones to the sum. So the error in the sum may be zero even when the map is not a permutation. The error due to a permutation 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.

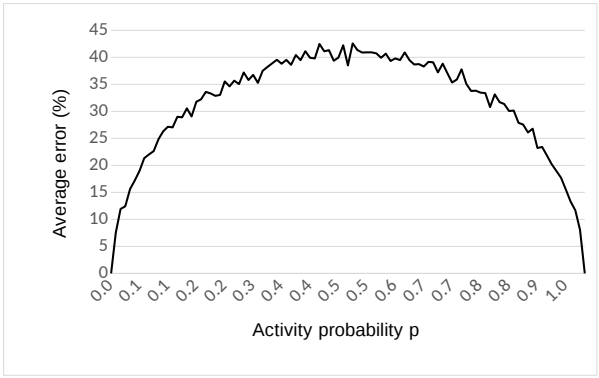
The computational experiment generates maps at random and takes the average of their errors as the values plotted in Figure 19b. The error turns out to be max at $p = 0.5$. The average number of 1's (the true number) increases directly with p so the relative error decreases with p. This result can be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

Figure 19c shows that increasing the network activity probability, the average relative error goes to zero.

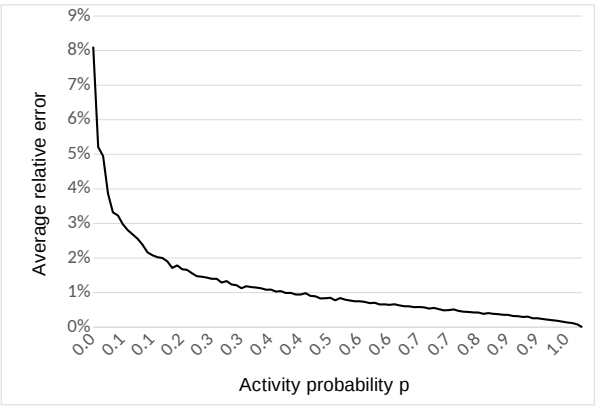
Figure 19d shows the average relative error with respect to the number of components. It can be seen that increasing the number of neurons (and notice that it is possible to simulate easily $4\,000\,000$ components), the error goes to zero. The connectivity is fixed at $p = 0.5$ for this experiment since it shows the maximum error. Note that the plot uses the negative logarithm to show decrease in error over several decades.



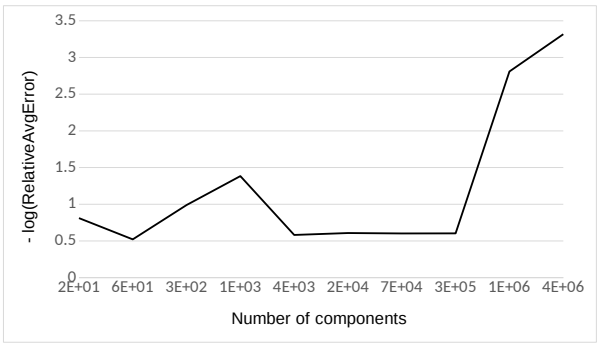
(a) Lumping approximation as the ratio $\frac{Q_{sim}(t)}{Q(t)}$. Simulation are obtained according to the probability of connection p between the component pools, and with a number of components $n = 100$. Parameters of Equation1 are $a = 1$ and $bc = -0.1$, so the exact state consists of $Q(t) = Q(0)0.9^t$, thus exhibiting a decreasing stable behavior.



(b) 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.



(c) 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.



(d) Negative log plot of average relative error with respect to the number of components up to 4 000 000 of components.

Figure 19: Error computations.

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-Rényi, which is uniform so good to average error): A full sampling is 100% and a deficient sampling is less percentage.