Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon

Research article

Identifying steady state in the network dynamics of spiking neural networks

Vivek Kurien George [a,b], Arkin Gupta [c,b], Gabriel A. Silva [d,b,*]

[a] Department of Bioengineering, University of California San Diego, La Jolla, CA, 92037, United States of America
[b] Center for Natural and Engineered Intelligence, University of California San Diego, United States of America
[c] Department of Computer and Electrical Engineering, University of California San Diego, United States of America
[d] Departments of Bioengineering and Neurosciences, University of California San Diego, United States of America

ARTICLE INFO

Keywords:
Complex networks
Network dynamics
Network evolution
Periodicity detection
Spiking neural networks
Representation of network dynamics
Symbolic dynamics

ABSTRACT

Analysis of the dynamics of complex networks can provide valuable information. For example, the dynamics can be used to characterize and differentiate between different network inputs and configurations. However, without quantitatively delineating the network's dynamic regimes, analysis of the network's dynamics is based on heuristics and qualitative signatures of transient or steady-state regimes. This is not ideal because interesting phenomena can occur during the transient regime, steady-state regime, or at the transition between the two dynamic regimes. Moreover, for simulated and observed systems, precise knowledge of the network's dynamical regime is imperative when considering metrics on minimal mathematical descriptions of the dynamics, otherwise either too much or too little data is analyzed. Here, we develop quantitative methods to ascertain the starting point and period of steady-state network activity. Using the precise knowledge of the network's dynamic regimes, we build minimal representations of the network dynamics that form the basis for future work. We show applications of our techniques on idealized signals and on the dynamics of a biologically inspired spiking neural network.

1. Introduction

Modeling physical and information systems as networks is a powerful approach, especially for systems such as biological neural networks with multiple elements interacting in non-trivial and complex ways. Network abstractions are used pervasively in science and engineering [1–6], and both the structure and dynamics of complex networks are well studied [2,7–9]. Some examples of dynamic processes on networks are oscillator synchronization, diffusion processes, rumour spreading, and epidemic propagation [10–14].

While there are many methods to compare network structures–for example, graph edit distance, distances based on node degree distributions, comparing the spectral properties of the graph Laplacian–there are comparatively fewer methods capable of mathematically describing network dynamics in a way that allows for normalized quantitative comparison of structurally and dynamically heterogeneous networks [15–19]. In other words, it has proven difficult to compare two structurally similar networks that dynamically evolve in their own respective ways, and it remains an open problem. The challenge is that the structural connectivity of the network cannot fully account for the temporal evolution of the network dynamics [20]. One representation

* Corresponding author at: Departments of Bioengineering and Neurosciences, University of California San Diego, United States of America.
E-mail addresses: vivek.kurien.george@gmail.com (V.K. George), arkin.gupta@gmail.com (A. Gupta), gsilva@health.ucsd.edu (G.A. Silva).

https://doi.org/10.1016/j.heliyon.2023.e13913
Received 21 August 2022; Received in revised form 15 December 2022; Accepted 15 February 2023

Available online 1 March 2023

of the evolving dynamical topology of a network consists of the vertices and edges derived from causal signaling paths [18], and it is a subset of the network's structural connectivity. In most cases, a necessary early step in comparing dynamic topologies is to create a mathematical abstraction–in our case, a graph–in a defined space that allows for the computation of different metrics. However, this is difficult to achieve because the dynamics of the networks evolve toward some steady-state configuration over different periods of time and in different ways. Knowledge of the steady-state starting point and periods is necessary to ensure that the appropriate dynamic regimes of network activity are present during comparison.

In this work, we address this problem by solving a number of technical issues that impede normalized quantitative comparisons of network dynamics. First, we propose methods to find the precise transient and steady-state regimes of network dynamics using an efficient algorithm in a novel way. We then use information about the dynamic regimes of network activity to construct mathematical abstractions from which we can systematically extract the paths and patterns of the dynamics of structurally heterogeneous networks.

To start we construct a mathematical description of the network dynamics. Next, we propose methods to find the Steady-State Statistics (SS-S) of the network's dynamics. For the network dynamics we chose to use dynamical rules developed by Silva in [21] which describes in generality the signaling dynamics of biological spiking neural networks (Section 3 contains the details). The SS-S consists of a steady-state starting point and the steady-state period. The steady-state starting point is the time point when the network's dynamics enter a periodic orbit, that is, start to repeat, and the steady-state period is the time interval, or the number of time steps in one period of steady-state activity. To determine the SS-S, we propose a signal summation based algorithm of complexity $O(n^3)$ and a string repetition search based algorithm of complexity $O(n)$, where n is the number of time samples under consideration. Finally, upon determining what part of the system's dynamical regime to capture, we construct a network representation of the dynamic topology. In this article we focus our techniques on the analysis of neural dynamics [22] and creating representations for machine learning [19].

The rest paper is organized as follows. In Section 2, we discuss related work. In Section 3, we describe the construction of our complex network model (a biologically inspired spiking neural network). Following that, in Section 4, we show how to transform the network dynamics into a symbolic description. Then in Section 5 we formally define the components of the SS-S. In Section 6, we introduce methods to find the SS-S of network dynamics. In Section 7, we show some applications of our methods: we apply the summation method to an idealized signal (Section 7.1), a random signal (Section 7.2), and to the dynamics of a small spiking neural network (Section 7.4). Then, we use the string search method the derive the SS-S of the small spiking neural network (Section 7.4). Using the SS-S, we show how to construct a spatial-temporal network representation of the dynamics (Section 7.5). This representation is a starting point for further study. To conclude, we put this work in a broader context and discuss some future directions.

2. Related work

In neuroscience there are many mathematical descriptions of neural network dynamics, each of which lend themselves to different insights. Some of the most commonly used descriptions are dynamical systems descriptions [23], network science descriptions [24, 25], topological descriptions [26], and symbolic descriptions [27]. Generally, underlying each of these descriptions is the membrane voltage dynamics of individual neurons or voltage readings from neural regions. The activity of neurons is commonly summarized by a so called "raster plot." It consists of a set of sequences of node activations, each sequence corresponding to the membrane dynamics of a particular neuron. In [27], Cessac rigorously showed that there is one-to-one correspondence between the membrane dynamics and raster plot.

To build compact mathematical descriptions of the neuronal dynamics such as in [17,18,22–24], one must detect when the network of spiking neurons enters a periodic, steady-state domain of activity. In the earliest work we found [28], the authors sought to detect periodicity and study the steady-state dynamics in computational models of biological neural networks with the aim of understanding the circumstances under which neural networks could maintain ongoing activity. They estimated periodicity through visual observation of aggregate network activity. In some simplified models of biological neural networks, it is possible to determine fixed-points and periodic attractors analytically [16], but for more complex neural models, it is generally not, which thereby requires the use of numerical methods to approximate solutions [29,30].

In computational modeling settings where it is difficult to accurately determine the SS-S analytically [31,32], even without the presence of noise and where the dimensions as well as the number of data points are large, we must use efficient algorithms. While the data mining community has developed a vast array of periodicity detection methods for time series databases [33–35], and their techniques can be used in a wide variety of applications where observations and data generation processes are both noisy, their methods assume that the collected data is in the periodic regime. In our case modeling a biological spiking neural networks–and in most modeling cases–periodicity can not be assumed because there are transient and steady-state regimes of the network dynamics. Because the existence of a transient regime violates the underlying periodicity assumption for the data input to the methods, and because we are interested in explicitly delineating the transient and steady-state regimes of network activity for further analysis, we can not use data mining periodicity detection techniques. Instead, we use linear time algorithms from the string analysis community to find the precise SS-S of the network dynamics.

To detect the SS-S of network dynamics, we recast the network dynamics into a form which can be analyzed by string processing algorithms [36,37], more specifically, string run-finding algorithms. A string is a linear sequence of symbols drawn from some alphabet, and it is a common form of data storage [38]. The earliest of these algorithms was developed by Main [39], and many other algorithms were developed to improve computation time and space complexity [40–42]. Kolpakov and Kucherov [43,44] conjectured O(n) complexity run-finding algorithms, and Bannai etal. [45] proved the conjecture. Thorough benchmarking of commonly used string run-finding algorithms was done in [46]. We take advantage of the linear time algorithms [45] developed by the string analysis community to determine the SS-S of the dynamics resulting from a biological spiking neural network model.

3. Network description

A biological spiking neural network is a type of complex network, that is, a dynamical system composed of interacting elements. We model the topology of such a network as a directed graph [47]. We define a directed graph G as the tuple $G = (V, E)$, where the set of nodes $V = \{v_1, v_2, \dots, v_n\}$, n is the number of nodes in the network, E is a set of directed edges $E \subset V \times V$. Nodes in the graph represent neurons, and edges represent axons.

The dynamics of each node in the network follows that of a so called geometric dynamic perceptron [21]. Full details and mathematical proofs for the node and network dynamics can be found in [21]. Briefly, this model captures the competing interactions of signals from upstream nodes incident on a target downstream node, along directed edges. The model takes into account how temporal latencies produce offsets in the timing of the summation of incoming discrete events due to the physical geometry of the edges as well as the network's structural connectivity. The dynamics that result from the weighted summation process are then responsible for the activation of the target node. At the core of this model is the interplay between the incoming signals and the target node's refractory period–an unresponsive period which can be construed as the time for internal processing at the individual node level. Built on this theoretical and conceptual framework, the geometric dynamic perceptron model is an extension of the classical perceptron model [48] and a generalization of the integrate-and-fire neuron model [49].

In our model, the i^{th} neuron takes on a binary state $\omega_i(t) = \{0, 1\}$ at time t. A neuron's state is determined by its underlying membrane potential. An axon/directed edge from neuron j to neuron i is given by $e_{ji} = (v_j, v_i)$, where $e_{ji} \in E$. Due to the geometric embedding of the complex network, every edge in G has an associated delay, τ_{ji}, which is the time it takes a signal originating at neuron j to reach neuron i. In general, τ_{ji} can be a function of several variables, but in this work, the delays were initialized using a uniform distribution then kept constant.

4. Symbolic description of network dynamics

Here we construct a mathematical bridge from a dynamical system description to a symbolic coding description following the work of Cessac etal. [50,51]. We let $V_i(t)$ be the membrane potential of neuron $i \in 1 \dots n$ at time t. Then we construct a vector representation of all the node membrane potentials $\boldsymbol{V}(t) = [V_i(t)]_{i=1}^{n}$ at a given time t. The dynamics begin at $\boldsymbol{V}(0)$ which is the initial condition of the system. The range of values of each node's membrane potential is in the closed interval $[V_{min}, V_{max}]$. A node activates when its membrane potential reaches or exceeds the threshold for activation $\Theta \in [V_{min}, V_{max}]$. For a trajectory $\boldsymbol{V}$, the activation times of neuron i are given by:

$$t_i^{(m)}(\boldsymbol{V}) = min\left\{ t | t > t_i^{(m-1)}(\boldsymbol{V}), V_i(t) \geq \Theta \right\}. \tag{1}$$

In other words, $t_i^{(m)}$ is the time of the m^{th} activation of neuron i, which we denote as the minimum of the set of activations after the $(m-1)^{th}$ activation when the membrane voltage reaches the threshold from below. As an initial condition for the set of activation times, we set $t_i^{(0)} = -\infty$. After activation, the neuron's membrane potential resets to some value $V_{reset} \in [V_{min}, V_{max}]$:

$$V_i(t) \geq \Theta \implies \lim_{\Delta \to 0^+} V_i(t + \Delta) = V_{reset}. \tag{2}$$

Next, we generate a symbolic coding description of the system's dynamics by formalizing the notion of a spike raster plot. A raster plot is a sequence $\{\boldsymbol{\omega}(t)\}_{t=0}^{+\infty}$ of vectors $\boldsymbol{\omega}(t)$, where $\boldsymbol{\omega}(t) = [\omega_i(t)]_{i=1}^{n}$. If neuron i activates at time t (given by eq. (1)), $\omega_i(t) = 1$, otherwise neuron i is quiescent and $\omega_i(t) = 0$.

$$\omega_i(t) := \begin{cases} 1 & \text{if } V_i(t) \geq \Theta, \\ 0 & \text{if } V_i(t) < \Theta. \end{cases} \tag{3}$$

The dynamic evolution of the membrane potential of neurons (nodes) in the system is given by:

$$\boldsymbol{V}(t + \Delta) = \boldsymbol{F}(\boldsymbol{V}(t)) \tag{4}$$

Where $\boldsymbol{F} = [F_i]_{i=1}^{n}$. In the non-refractory period of the neuron, the model evolves as follows:

$$F_i(V(t)) = \gamma V_i(t)(1 - \omega_i(t)) + \sum_{j \in \alpha} s_{ji} \omega_j(t - \tau_{ji}), \tag{5}$$

where $0 < \gamma \leq 1$ is the membrane voltage decay constant, $V_i(t)$ is the membrane voltage of node i, α is the set of signals that arrived at node i at time t, s_{ji} is the synaptic weight between neuron j and i, τ_{ji} is the delay between neurons j and i, and $\omega_j(t - \tau_{ji})$ indicates the activation of node j at time $t - \tau_{ji}$. In words, at any given time t, either the current membrane potential, $V_i(t)$, decays until V_i approaches V_{min} or the current membrane potential is increased by the contribution of a new arriving signal multiplied by the synaptic weight. Finally, we note that the refractory period is not explicitly discussed in the above description; it is a period of time where the node is unresponsive to incoming signals and the membrane potential is held at V_{min}.

5. Defining the steady-state statistics

The period of steady-state activity of the network dynamics is defined as the repeating or periodic set of network states that the dynamics settle to after some transient activity. Given our network construction, the regimes of network activity are persistent activity and neural death, i.e., cessation of all activity in the network. Although neural death is a type of steady state, it is the trivial case in this work. Persistent activity can either eventually enter a periodic orbit of network states or continue to exhibit non-periodic activity. In situations where it is difficult to determine a network's steady-state behavior analytically, we forward calculate the network activity to computationally tractable limits and analyze the resulting dynamics to determine the network's SS-S.

Given that the state of the network is $\boldsymbol{\varpi}(t)$, we say that the network is in a steady state if for $\forall t \geq t_{ss}$

$$\boldsymbol{\varpi}(t+c) = \boldsymbol{\varpi}(t+c+T) \tag{6}$$

where t_{ss} is the steady-state starting point, T is the steady-state period, and c is some arbitrary positive time shift.

The transient regime of the network's dynamics is the evolution of $\boldsymbol{\varpi}(t)$ until it enters the steady state regime. Therefore, t_{ss} also marks the end of the transient regime. Once the network activity has entered a periodic orbit, we define the period of steady-state activity as the interval of time/time-indices it takes for the activity to repeat. In networks with many nodes or with some of feedback mechanism that modifies the network parameters as a result of network activity, it is possible that the forward computation will not have been calculated for a long enough data window for the entire network to exhibit steady-state behavior. The methods presented in this paper can only determine the SS-S up-to the extent of the forward calculation, with the longest period equal to half the total simulation time interval.

6. Methods for finding network steady-state statistics

6.1. Summation method

We develop the summation method using the simulation data set $\mathcal{M}$:

$$\mathcal{M} = \left\{ \boldsymbol{\varpi}(t_0), \boldsymbol{\varpi}(t_1), \boldsymbol{\varpi}(t_2) \dots \right\}. \tag{7}$$

In words, $\mathcal{M}$ is a set composed of vectors $[\omega_i(t)]_{i=1}^{n}$, where n is the number of nodes in the network. For simplicity, one can imagine $\mathcal{M}$ as a matrix whose rows represent nodes and columns represent time. The state of node i is given by $\omega_i(t)$ eq. (3), where $\omega_i(t) \in \{0,1\}$. We assume the dynamics are sampled appropriately such that a node only activates once per time point at most. Each $\omega_i(t)$ is defined over the interval $t \in [t_0, t_f]$, where t_0 is initial observation time and t_f is the final observation time. The signal $\omega_i(t)$ is comprised of two parts: the transient part $g_i(t)$ which is defined over the interval $t \in [t_0, t_{SS_i})$, and the periodic part $h_i(t)$ which is defined over the interval $[t_{SS_i}, t_f]$. The network's activity can be described by the following set of equations:

$$\begin{aligned}
\omega_1(t) &= g_1(t) + h_1(t), \\
&\dots \\
\omega_i(t) &= g_i(t) + h_i(t), \\
&\dots \\
\omega_n(t) &= g_n(t) + h_n(t).
\end{aligned} \tag{8}$$

Since $h_i(t)$ is periodic and given the form of $\omega_i(t)$ in eq. (8), we can describe it using a sequence of delta functions:

$$h_i(t) = \sum_{m=1}^{r} (\delta(t - mb_1) + \dots + \delta(t - mb_q)). \tag{9}$$

In eq. (9) there are a total of q activations per period and r total periods. Each element of the sum is the time of activation in some m^{th} period, where m is a multiple of the fundamental period. Note, during implementation, care must be taken for the final period in the data window because it is likely to be a fraction of a period.

To find the SS-S of the system, we start with the SS-S of each $\omega_i(t)$ (SS-S-i). SS-S-i is comprised of the steady-state start point and steady-state period. After all the steady-state start point are determined for each $\omega_i(t)$, the latest steady-state start point across all $\omega_i(t)$ is used as the steady-state start point of the system. We define the steady-state period for the system as the longest period across all $\omega_i(t)$.

To determine the SS-S-i for the i^{th} node in the network we use $z_i(k,T)$:

$$z_i(k,T) = \int_{k}^{k+T} \omega_i(t)\, dt, \tag{10}$$

where k is the candidate steady-state start point of ω_i and T is the candidate steady-state period of ω_i. The longest period sets the upper-bound for T, while the lower bound of T is fixed to two consecutive time points. In the simulation time interval $[t_0, t_f]$, we define $t_i \in [t_0, t_f]$ to be the time of steady-state start point of $\omega_i(t)$, which is the start of $h_i(t)$.

For all $t \geq t_i$, the signal $\omega_i(t)$ is in the steady-state regime. Let $\psi_i \in [0, \frac{t_f - t_0}{2}]$ be the length of the period of $\omega_i(t)$'s steady state. We evaluate $z_i(k,T)$ over the candidate periods $T \in [0, \frac{t_f - t_0}{2}]$ and candidate starting points k. Our goal is to find the correct ψ_i and t_i. The regimes of operation where we evaluate eq. (10) are

1) $k \in [t_0, t_i)$, $T \in [0, \frac{t_f - t_0}{2}]$
2) $k \in [t_i, t_f]$, $T \in [0, \frac{t_f - t_0}{2}]$
   a) $k \in [t_i, t_f]$, $T \in \{\psi_i, 2\psi_i, \dots n\psi_i\}$
   b) $k \in [t_i, t_f]$, $T \notin \{\psi_i, 2\psi_i, \dots n\psi_i\}$

We define some function $f(x)$ to be uniformly zero starting at some constant c when the function stabilizes at the value of zero after some point c, i.e.,

$$\forall x \geq c\,, f(x) = 0 \tag{11}$$

We will use the notion of uniformly zero to analyze the results of eq. (10).

To determine the SS-S-i, we find the smallest k and smallest T in A, where

$$A = \left\{ (k,T) \mid \forall k > c, \frac{\partial z_i(k,T)}{\partial k} = 0 \right\}. \tag{12}$$

Note that A is a set of ordered pairs where each element is a candidate start time and candidate period.

To find the final (k,T) for node i, first, we determine the steady-state start point of $\omega_i(t)$ (i.e., t_i) by finding the ordered pairs containing the minimum k in the set A:

$$t_i = \left\{ k \mid \min_k A \right\}. \tag{13}$$

We now consider a subset of A, where the first elements in the ordered pairs–the candidate start time–are the same, but the second elements–the candidate periods–are different due to the presence of harmonics of the fundamental period. The steady-state period of $\omega_i(t)$ (i.e., ψ_i) is

$$\psi_i = \left\{ T \mid \min_T A\Big|_{k=t_i} \right\}. \tag{14}$$

Once we determine the final t_i and ψ_i for each node, the maximum values across all nodes are selected for the SS-S of the system.

A limitation of using uniform sampling is its computational cost, which is a function of the sampling rate. To apply the summation method in the non-uniform sampling setting we define $\tilde{\mathcal{M}}$:

$$\tilde{\mathcal{M}} = \left\{ \boldsymbol{\omega}(t) | \exists \omega_i(t) = 1 \right\}. \tag{15}$$

One can imagine $\tilde{\mathcal{M}}$ as a matrix whose columns contain at least one node activation, that is, every column has at least one row with a value of 1. We use $\tilde{\mathcal{M}}$ in Section 7.3 to find the SS-S in an asynchronous, event-based, non-uniformly sampled simulation framework. Generally, $\tilde{\mathcal{M}}$ can be used in settings where the data is not guaranteed to be uniformly [52] or optimally sampled. For example, we may not be able to identify the optimal sampling rate a priori for networks whose dynamics are affected by evolving network parameters that affect the sampling rate. However, under the non-uniform sampling setting eq. (15), the summation method may not capture the true SS-S, and this is a limitation. Incorrect SS-S can result when there is a contraction or expansion of timescales during the time interval of node activations, or when the relative sequence of node activations does not change. But in some applications, approaches like the summation method are satisfactory, for example in biological neural systems where a global clock does not exist [53].

6.2. String search method

To apply the efficient string search method described later in Section 6.2.3, we must convert the dynamic evolution of the system (Section 4) into a string description. We use either $\mathcal{M}$ and $\tilde{\mathcal{M}}$ as the input to the string search methods. To construct a string representation from a sequence of activations, we define a labeling function that maps activations to an alphabet of fixed length strings of size s. The alphabet is denoted as Σ_s. $\mathcal{L} : \boldsymbol{\omega} \to \{\Sigma_s, \epsilon\}$, that is, $\mathcal{L}$ maps activations of neurons to fixed length strings Σ_s and quiescent states to the empty string ϵ, resulting in an alternate description of $\tilde{\mathcal{M}}$ that was the non-uniform sampling description defined in eq. (15). We let σ_{s,ω_i} be the symbol associated with the activation of the i^{th} node. For brevity, we will drop the subscript s and assume a fixed length string in place of the symbol. Note that in the string we do not distinguish the time of activation; the string only contains information about the set of nodes which activated. We define $\mathcal{S}$ as the set of strings representing the evolution of the network. The ordering of this set is based upon the ordering of $\tilde{\mathcal{M}}$. The input to the string run-finding algorithm is

$$\mathcal{S} = \left\{ \sigma_{\boldsymbol{\omega}(0)}, \sigma_{\boldsymbol{\omega}(1)} \cdots \right\}. \tag{16}$$

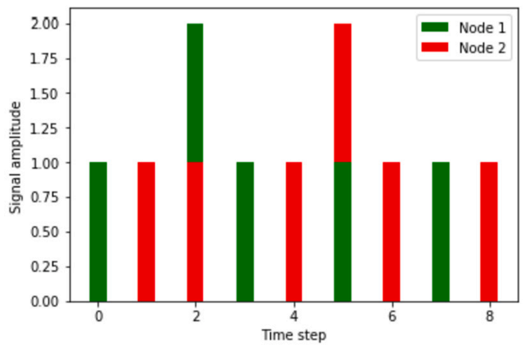


Fig. 1. Graphical representation of a set of ordered node activations that we will convert to a string of symbols. The activity of two nodes is presented. The y-axis represents the state of the nodes. *12a12a212* is a possible encoding of the signal.

To consider uniform sampling, the only change required is for $\mathcal{L}$ to assign a string to the quiescent states of $\boldsymbol{\omega}(t)$ rather than mapping those states to ϵ, thereby we construct an alternate to $\mathcal{M}$ (eq. (7)). Uniform sampling increases run time of the string search methods due to the increase in the number of time samples.

6.2.1. An example string transformation

In this section we present a simple technique to convert a given discrete signal $x(t)$ into a string. We begin by assuming that a system has n nodes that can activate. We then enumerate each node arbitrarily and keep it fixed over the course of the analysis. We start with an empty string, and, since our system has information about which node activated, if node i activated at time $t = 1$, we simply append i to the string. As such, we append each subsequent node activation to the string. In the case of multiple node activations at the same time, we generate a new enumeration and append it to the string. Consider a network with two nodes whose activation pattern is depicted in Fig. 1: Node 1 activates at t_1, node 2 activates at t_2, both activate at t_3, and so on. Now we enumerate the event of the two nodes activating simultaneously with an arbitrary character (here "a") and return the following output: *12a12a212*. By doing so, we convert our signal into a string and reduce the problem from finding the steady state in a complex signal to finding a repeating substring. Note that when encountering events such as simultaneous activation, we use an unused character to represent it and store the mapping in a hash table.

6.2.2. Naive string search

In this section we present a naive string approach for periodicity detection, it is a prelude to the string run-finding algorithm discussed in Section 6.2.3. The input to the string search uses the construction from Section 6.2. Here we do a brute force search to identify a repeating, terminating substring in our signal-converted string. To do so, we loop over the period of the expected steady-state pattern, and for each period, we loop over the start time, essentially testing all $n(n+1)/2$ substrings as the candidate steady-state pattern. Formally, given a string of length n, we find the pattern using Algorithm 6.2.2. The search terminates when the algorithm finds the shortest substring that repeats until the terminus of the input string. The outputs of the procedure are the SS-S. A basic analysis of the pseudo-code yields an $O(n^3)$ time complexity, which is improved upon in the next section.

6.2.3. Efficient string search

The efficient algorithm uses the input $\mathcal{S}$ (eq. (16)). We let $\sigma[t]$ denote the t^{th} location in $\mathcal{S}$. In stringology [37,54,55], a string of length n has period p if $\sigma[t] = \sigma[t+p]$ for any t, such that $1 \leq t \leq n-p$. If we were to consider the case where a string can be decomposed into sub-strings u, v, and w, such that $uv^k w$, then string u is considered the prefix of input string, string v is the period repeated k times, and the string w is considered the suffix of the input string.

Due to our interest in finding the starting time and period of the steady state, we are concerned with strings that end with repetition. More specifically, strings that end at some location in the period v during the course of a run [39]. In stringology, a primitive string is a string that is the fundamental period of the repetition (and can not be further compressed into the form v^k). The leftmost primitive string is called maximal, that is, if the primitive string can not be shifted anymore to the left. If the primitive string could be shifted further left, a new prefix and an addition to the k^{th} repetition would result, and that point would be the leftmost maximal. A run is defined as the maximal, fractional, and primitively-rooted repetition (example: $12312312312 = (123)^{3\frac{2}{3}}$, where the run is 123). Fractional runs take into account the stop of the forward calculation at some fraction of the period. Now we have the stringology terminology to describe our needs. We are interested in strings which end in runs, i.e., periodic sequences. Moreover, we are interested in the leftmost primitively-rooted repetition of that last run because that is the starting point of steady state.

The problem of finding runs in strings has been solved using O(n) time algorithms [43], where n is the length of the string. Modern run-finding algorithms [41,46,56,57] rely on building and analyzing complex data structures that are amenable to finding repetitions in strings. Algorithms which detect runs follow a general procedure. First, a suffix array is built [58,59]. Then, the Longest Common Prefix [60] is calculated. Next, the leftmost runs are found using Lempel-Ziv Factorization [61,62]. Finally, all maximal repetitions are found using Kolpakov and Kucherov's [43] linear time algorithm.

Algorithm 1 Naive Find Steady-State Statistics.

```
    Input signal
    Output repeating-signal, steady state: start & period
 1: procedure FindSteadyState
 2:     str ← signal
 3:     strlen ← length of signal
 4:     perlen ← length of candidate period length
 5:     index ← length of current position in string
 6:     for perlen = 1 to strlen do:
 7:         for index = 0 to strlen-1 do:
 8:             substring = str[index:index+perlen]
 9:             if CheckTerm(substr, str, index, perlen) then
10:                 repeating-signal ← substring
11:                 start-steady-state ← index
12:                 period ← perlen
13:                 return repeating-signal, start-steady-state, period

14: procedure CheckTerm
15:     str ← signal
16:     substring ← candidate steady state
17:     index ← candidate start point
18:     perlen ← length of candidate period length
19:     while i = index to strlen-perlen do:
20:         if substring != str[i:i+perlen] then
21:             return False
22:         return True
```

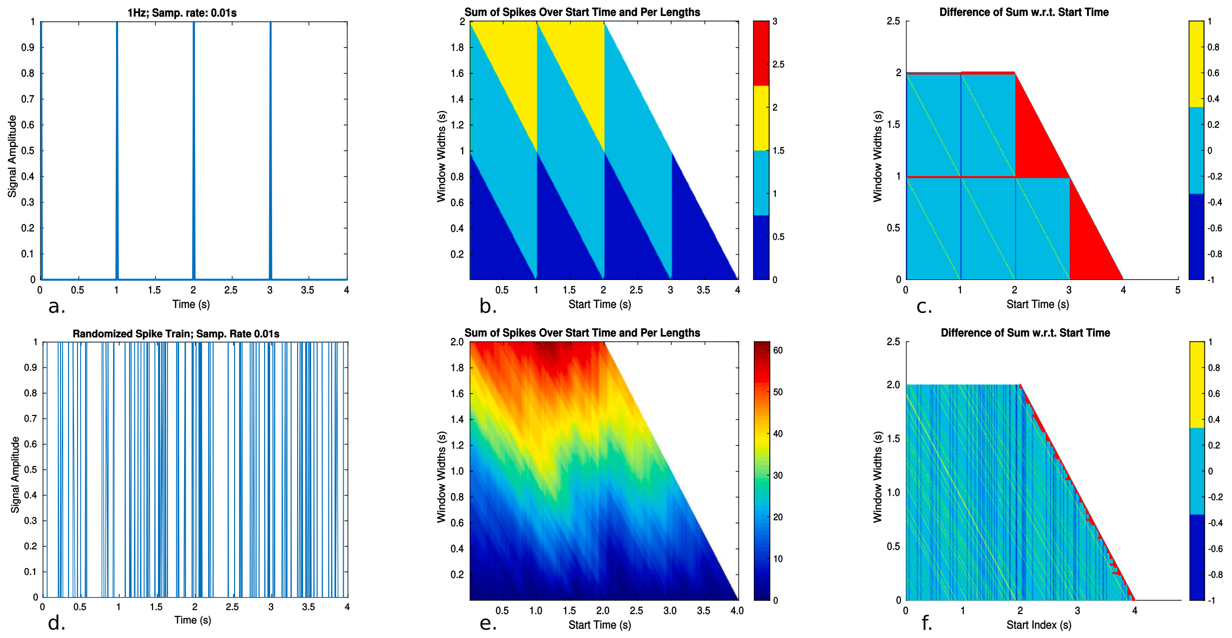


Fig. 2. An illustration of the summation method. (a) An idealized $1Hz$ signal, sampled at $100Hz$, comprised of Dirac impulses. The duration of the signal is $4s$. (b) The sum of pulses over the candidate starting points and candidate periods, $z_i(k,T)$, eq. (10). The values of the heat map are defined over the set of possible starting points and periods, whereas, the white-space corresponds to the starting points and periods that can not be determined given the data. For example, we can not determine periods longer than half the data because we do not observe a repetition. (c) A plot of the difference of the sum of spikes with respect to starting position, $\frac{\partial z_i}{\partial k}$ (the condition of eq. (12). The two leftmost horizontal red lines refer to the period and the harmonics of the original signal. The red horizontal lines indicate when the difference is zero from its leftmost end, the start time, to the end of the data. At the leftmost position of the red line is the start time of steady state. The heat map indicates the values of the differences. (d) A Poisson distributed sequence of Dirac impulses sampled at $100Hz$. (e) Its sum of spikes, eq. (10). Note that the symmetries present in panel (b) are absent. (f) A plot of the difference of the sum of spikes with respect to starting position, $\frac{\partial z_i}{\partial k}$ (the condition of eq. (12). As indicated by the lack of long contiguous red horizontal lines, there is no dominant start time and harmonic.

Once the maximal runs have been found, we store the runs that end at the input string's final character position, and we output the run which has the minimum starting position. The primitively-rooted repetition is the period of steady-state activity. We detected runs using runFinder, which is an implementation courtesy of Dr. Hideo Bannai of Tokyo Medical and Dental University in Japan.

6.3. Methodological limitations

Both the summation and string search periodicity detection methods have limitations. For example, we only considered the binary node states, but other network observables may be more appropriate indicators of steady state, such as the ordered list of node activators (that is upstream nodes whose signals cause a target node to activate). While the string search method can accommodate an activators list, the summation method can not without post-processing.

The summation method uses the latest start time and the longest period across all nodes; therefore, we are assured that the system is in a steady state. But while the system is in a steady state, the outputted period of the steady state may include non-integer multiples of the periods of node activity. That is, nodes with shorter periods would terminate midway through their n^{th} period, here n is a real number greater than or equal to 1, at the time point when the node with the longest period ends. If the requirement is to capture integer multiples of the periods of all nodes dynamics, then the greatest common divisor of all their periods is required. The resulting value may exceed the feasible observation time of the network's dynamics.

Non-uniform sampling causes a few issues with both the summation and string search methods. Situations can arise where there is a contraction or a dilution of time-scales, but the order of network activity is unchanged. This can only be detected when the system is observed through uniform sampling. Any drift in network activity without causing a reordering in $\tilde{\mathcal{M}}$ (eq. (15)) will be treated the same by both the summation and string search methods.

Another non-uniform sampling issue that afflicts the summation method is presence of an incorrect initial set of activations in the first period. This issue does not occur in the uniform sampling setting. A node activation that is not actually part of the steady-state set of node activations may erroneously be included. This happens when the number of time indices between the erroneously included activation and the actual steady-state starting point is less than the number of time indices between the spacing of any of the node activations of the actual steady-state period. Outside the first period, this issue will not arise because of the strict requirement that the number of spikes in the smallest period from the start to finish be maintained between all consecutive starting points. This issue can be corrected by post-processing the periodic network activity by comparing the contents of the detected periods. When an erroneous spike is detected, one solution is to remove the erroneous spike from consideration by flipping the corresponding bit, rerunning the algorithm, and rechecking for alignment.

7. Applications

7.1. Summation method: 1 Hz signal

We illustrate the summation method using as input the waveform in Fig. 2a, which is an ideal $1Hz$ impulse train sampled at $100Hz$. The signal consists of a sequence of zeros and ones. In Fig. 2b, we evaluate $z_i(k,T)$, eq. (10). In Fig. 2c, we show $\frac{\partial z_i}{\partial k} = 0$ a condition in eq. (12). The red horizontal lines in Fig. 2c indicate the candidate periods and starting points which correspond to eq. (12).

The candidate periods fall into two categories. Category 1 encompasses the two red horizontal lines which share a start time of 0 seconds, and category 2 encompasses all the other red horizontal lines which start at start time 1 second, 2 seconds, and 3 seconds. For the signal's SS-S, we choose the earliest start time per eq. (13) and the shortest period per eq. (14). The period of 1 second explains the data from a start time of 0 seconds onward. The other red horizontal lines are a result of the envelopes for some range of (k,T) in eq. (10) as noted in Section 6.1, meeting the condition of eq. (11) but explaining less of the data.

7.2. Summation method: randomized impulse train

Here we input a Poisson distributed sequence of impulses as shown in Fig. 2d to the summation method. As a result of applying eq. (10), we observe in Fig. 2f that there are bands of values of start times and period lengths which have the same number of spikes. However, the bands of start time are not horizontally contiguous, which is our requirement for the SS-S. One way in which we can handle noise is by relaxing the criteria for the change in the number of spikes in adjacent starting points.

Given the stochastic nature of input signal, we observe that Fig. 2i no longer exhibits the kind symmetry found in Fig. 2c. Therefore, the uniformly zero condition eq. (11) is only satisfied in eq. (12) for k at the terminus of the data window. In other words, all the zero differences (time points meeting the uniformly zero condition) are relegated to a few starting points at the end of the data-window, and there are no periods of non-negligible size, or easily discerned harmonics thereof. In such cases, when the system's dynamics has not entered the regime of periodic activity, we may still end up with candidate periods and starting points due to an unchanging number of spikes in the last few starting points, k, where the uniformly zero criteria is met. Therefore, one must make sure that the forward calculations are done for long enough to ensure that the system is actually in steady state. A simple empirical workaround is to ensure that the steady-state starting point occurs for at least some fraction of the forward-calculated data-window's length.

7.3. Summation method: non-uniformly sampled network dynamics

Here, we apply the summation method to detect the SS-S of the dynamics on a biological spiking neural network as described in Sections 3 and 4. The network under consideration is illustrated in Fig. 3a. The complexity of the dynamics in this network arises from the interplay of edge delays and node refractory periods. We introduce an image of a hand written digit from the MNIST

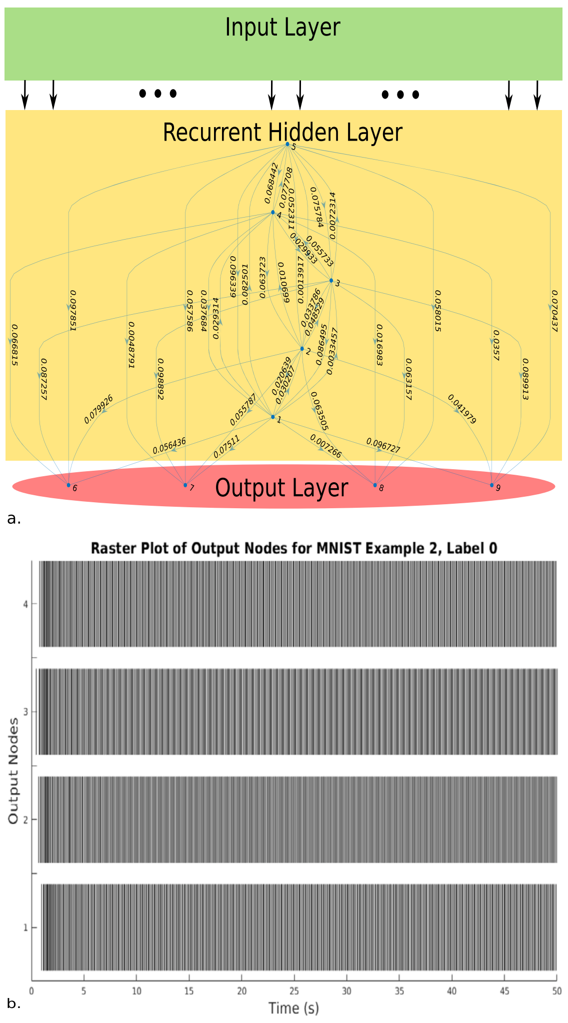


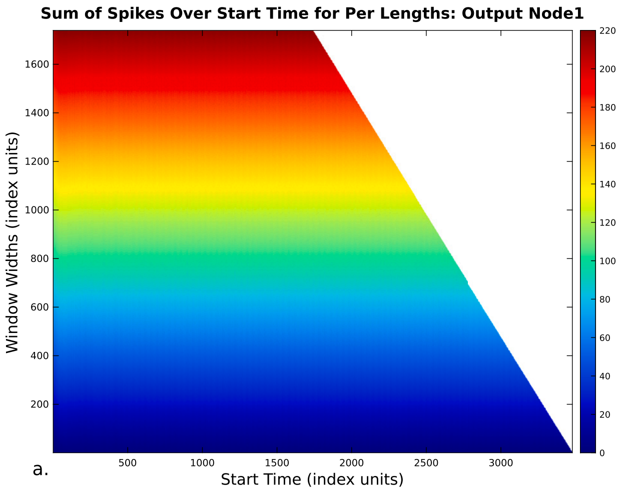
Fig. 3. (a) A sample network diagram. The Input layer consists of 784 nodes. The Recurrent/Hidden layer contains 5 nodes. The Output layer contains 4 nodes. The values of the Input layer are set based upon the MNIST images. If the pixel contains a non-zero value, then the corresponding input node is activated and is part of the initial condition of the network. The network's connectivity is as follows: Feed-forward connections from the Input layer to the Hidden layer, recurrent connections within the Hidden layer, and feed-forward connections from the Hidden layer to the Output layer. A sample of the edge delays are shown. We used a node refractory period of $1/2^4 s$, a node activation threshold of $\Theta = 2$, a membrane decay constant of $\gamma = e^{-2\Delta t}$, where Δt is the current time minus the signal arrival time. We set the synaptic weights and the outgoing signal amplitudes to unity. (b) The resulting network activity of the nodes in the Output layer resulting from activity of the Input layer. The network activity is shown as a raster plot, eq. (3) We forward calculate the network dynamics using eq. (5) to $50s$. We chose this length of simulation time for two reasons. First, the length was long enough to make sure that most initial conditions lead to steady-state activity for our purposes. Second, that we are able to observe multiple periods of steady-state activity to make visualization of the SS-S easier.

database [63] as binary vector at time $t = 0$ to the input layer nodes of the network and analyze the resulting dynamics, eq. (5). In order to get a sense of the data we wish to analyze, we created a raster plot of the output nodes' activity (Fig. 3b).

We construct the input to the summation method using $\tilde{\mathcal{M}}$ (eq. (15)); this puts us in the non-uniform sampling setting. We show the start time index and the size of the periods in units of time indices in Figs. 4a, 4b. Note that each iteration through the method is done on a per node basis, and the SS-S of the system is based on the latest start time and the longest period among all the analyzed nodes.

As with the previous examples, the smallest value on the abscissa, where the red horizontal line begins, is the start of the steady-state behavior. Each successive red line (from the set of red lines with the same minimum start time) on the ordinate represents the harmonics of the fundamental period of steady-state activity.

In Fig. 4 we switched from using seconds to using time indices. While the start time index corresponds to a specific absolute time value (seconds) in the non-uniform setting, the number of time indices which make up a period corresponds to a range of absolute time values. Fig. 5a illustrates the variation in the absolute time for a fixed period as a function of the start time index. By evaluating eq. (10) for a fixed period, in units of time indices, and across start time indices, we observe the fluctuations in period length measured in absolute time between start times. This is due to non-uniform sampling. The absolute time between adjacent activations varies; therefore, moving one time index does not correspond to a move of one absolute time unit.



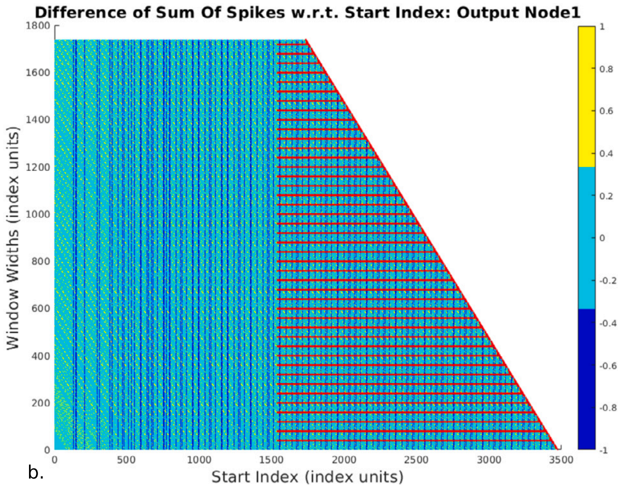


Fig. 4. (a) The sum of spikes, eq. (10), over candidate periods and starting indices for output node 1. Given our approach to the summation method in the non-uniform sampling setting, it no longer necessary to discuss units in terms of absolute time (seconds). In the non-uniform sampling setting, the candidate periods and starting points are selected and swept along the sampling points determined by the system's node activations. For example, a period of length y index units may refer to a range of absolute times. As such, the candidate SS-S are determined based on indices rather than an absolute time measure. The system's SS-S is determined from analysis of all nodes in the network and from selecting the latest starting point and longest period amongst all nodes. As the size of the period increases, more node activations are captured, resulting in higher values for the evaluated sums. (b) The first finite difference in index units, $\frac{\partial z_i}{\partial k}$ (the condition of eq. (12). The result of the difference is given by values of the heat map. The red horizontal lines denote where the uniformly zero condition is met. The starting point of steady state is the minimum of the start indices of the red horizontal lines. The fundamental period of network activity is given by the lowest valued red horizontal line with the minimum starting index. Each harmonic of the fundamental period is given by the incrementally higher horizontal red lines with the same starting indices. The red horizontal lines at the end of the data range of starting points correspond to consecutive candidate periods with the same number of spikes. In this situation, the candidates are easily filtered using the minimum starting point criteria. Light blue values correspond to adjacent starting points whose difference in the number of spikes is 0.

Each horizontal line in Fig. 5a corresponds to a harmonic of the period. The fluctuations in a particular horizontal line in Fig. 5a correspond to the aforementioned absolute time between the start of the period and end of the period. Fig. 5b shows the number of spikes which are present in each harmonic of the period. There is a one-to-one correspondence between Figs. 5a and 5b in terms of bottom to top ordering of curves. As expected, and required by eq. (6), once the system is in steady state, the number of spikes in a period stabilizes to the steady-state number of spikes in a period. The number of spikes in a period is unchanging over a period for $t \geq t_{SS}$. Each vertically successive horizontal curve in Fig. 5b shows the number of spikes in each of the corresponding harmonics of the period.

7.4. String search method: non-uniformly sampled network dynamics

We apply the efficient string search method described in Section 6.2.3 to detect the SS-S of the network dynamics considered in Section 7.3. We use eq. (16) to construct the input to the run-finding algorithm. Since the network has a total of 793 nodes (784 Input layer nodes associated with the pixels in each image, 5 Hidden layer nodes, and 4 Output layer nodes), each node is associated

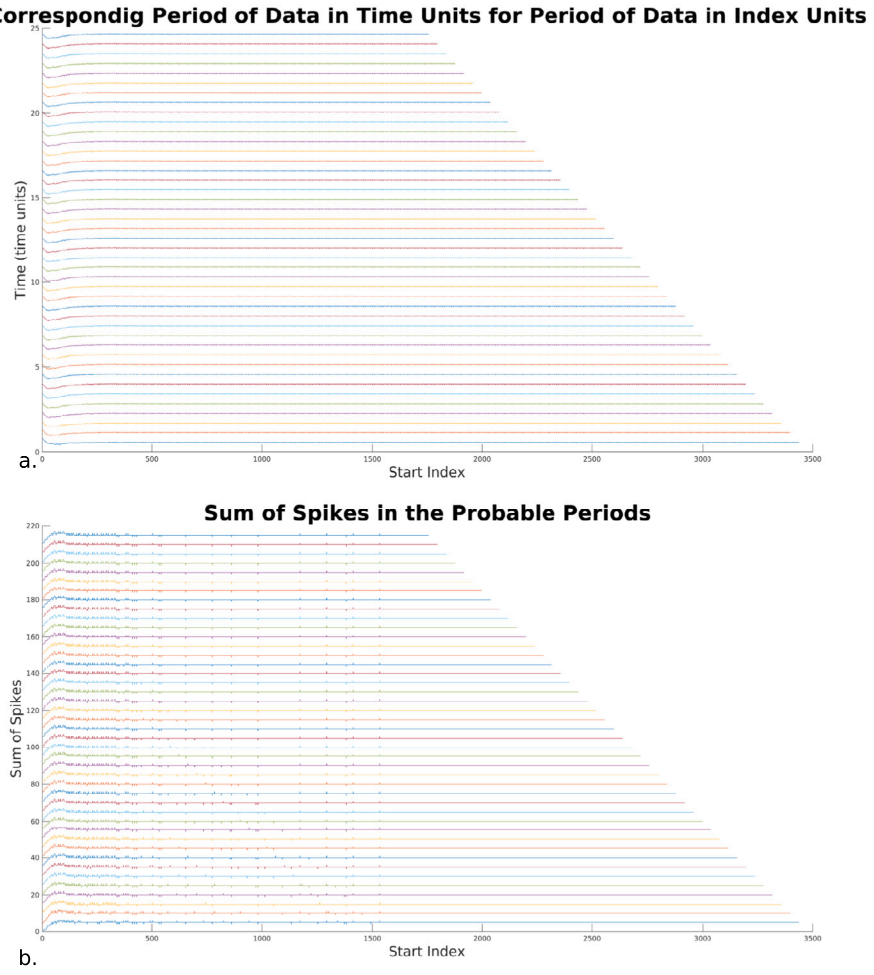


Fig. 5. We show the effects of the summation method on non-uniformly sampled network dynamics (following from Fig. 4) as it pertains to both time indices of the detected period and the number of spikes in the detected period, as well as how the two relate to absolute time. In (a), the abscissa shows the candidate start indices and the ordinate shows the length of the period in time units of seconds. Each horizontal line in the plot corresponds to a previously determined harmonic–the horizontal red lines from Fig. 4b. The bottom-most horizontal line corresponds to the fundamental period, and each subsequent one corresponds to the harmonics of the fundamental period. Fluctuations of each horizontal line correspond to the length of the period in seconds when counting the fixed number of time indices from the start index to the end index of a given period. Because of non-uniform sampling, the fluctuations remain. This is in contrast to (b), where the sum of spikes in each period during steady state is constant. In (b), we count the number of spikes in each period as a function of the start index (eq. (10)), and we can clearly see that after the inception of steady-state dynamics, the number of spikes stabilizes to the steady-state value, that is, a constant value for the sum of the spikes. The ordinate is a count of the number of spikes. The abscissa is the start index. And each horizontal line corresponds to a harmonic, the bottom-most one corresponding to the fundamental period and each subsequent one a harmonic.

with a string of 3 symbols. Node 1 will have a label 001, node 2 will have label 002, and so on up to node 793 with label 793. We do not delimit the symbols with commas or other delimiters as those symbols increase the processing time.

Table 1 shows a sample output for illustration. For a string of length 8586 symbols, which translates to 2862 node activations, the run-finding algorithm finds a string of length 81 symbols, which is 27 node activations in a steady-state period and a run of length 6090 symbols, which translates to 2030 steady-state node activations. We use the precise SS-S to construct a graph of activations based on the set of causal relationships between node activations up to the point where the network dynamics stop producing new information; this occurs after the system enters steady state.

7.5. Constructing spatial-temporal graph representation of network dynamics

Using the causal node activity and knowledge of the network dynamics' SS-S, we construct a finite graph representing the dynamics of the network. The nodes of the graph refer to (v_i, t_k), where v_i is the neuron which activates at some time t_k. The edges of the graph represent the causal activators [21] of (v_i, t_k). Every time a neuron is activated, a new node, (v_{i+1}, t_{k+1}), is added to the graph. As such, the child nodes are the subsequent nodes that are causally activated. A set of root nodes represents the initial

Table 1
Sample outputs for the SS-S of the system using the efficient string search method. Each of the entries is in either string character units or refers to specific node numbers in a sequence. String character units refer to the index (rows 1 and 2) or interval (row 3) of a symbol or symbols in a string sequence. When mapping from String Character Units to network node activations, care must be taken to account for the length of the node's symbolic description. For example, because we are using a network with 793 total nodes and using the decimal counting system, we use three symbols to describe every node, and each of the values in string character units is divisible by three. Row 4 shows a sequence of node numbers which represents the sequence of node activations in one period in the steady-state regime. We inserted commas in row 4 only to make the nodes involved in the sample period more readable; inserting extraneous symbols into the string finding algorithm increases the run time and may not have enough benefit.

Sample Start Index (String Character Units)	2496
Sample Stop Index (String Character Units)	8586
Sample Period Length (String Character Units)	81
Sample Period (Node Numbers)	786, 788, 789, 785, 793, 791, 792, 790, 787, 786, 789, 788, 793, 785, 791, 792, 790, 787, 786, 789, 788, 785, 793, 791, 792, 790, 787

conditions and external perturbations to the network. By construction, we are able to distinguish repeated node activations, while still being able to construct subgraphs of neuron-neuron interactions.

Given this network representation of the dynamics, we are able to use network tools to do comparisons. Without the SS-S, it is difficult to determine the which portions of the network dynamics are relevant. Moreover, for situations where network dynamics do not taper off into quiescence, the SS-S provides a bound on the construction of the spatial-temporal graph. Finally, to do a comparison of representations from different sets of network dynamics, knowledge of the dynamical regime (i.e., the transient or the steady-state portions) is important.

8. Conclusion and future work

In this paper we propose methods to analyze network dynamics and show applications of those methods. Without these methods, at one extreme in a sense, analysis of a network's dynamics is based on an unknown portion of its transient activity because there is no way to know whether the dynamics have entered the steady-state regime. This results in the omission of information produced by the dynamic evolution of the model. At the other extreme, analysis of a network's dynamics is based on data that over-represents the steady-state regime because of the lack of precise knowledge about when the steady-state regime began and the number of periods present in the data. This results in a waste of computational resources and an imbalance when making quantitative comparisons of different sets of dynamics. At worst, the analysis of a network's dynamics is made without any knowledge at all of the dynamic regimes being analyzing. The determination of whether a system is in the steady-state or transient regime implicates different mathematical tools and considerations which come to bare during the analysis of the system's response.

This paper makes three important contributions toward the identification and analysis of normalized comparisons of network dynamics. First, we developed methods to calculate the precise SS-S of network dynamics using efficient algorithms, with each SS-S finding method providing different advantages. The summation method can be extended to a noisy setting by relaxing the uniformly zero criteria while maintaining the computational complexity. And the string search method utilizes a more efficient but less flexible algorithm because its computational complexity increases when considering noise. In addition, it can only handle limited noise types (insertion and deletion).

Second, we successfully applied our methods to idealized signals and the dynamics of biological spiking neural networks. Our methods detected the SS-S of a spiking neural network with complex dynamics resulting from the interplay between edge signaling delays, node refractory periods, and synaptic weights. Further comparison between network topologies and steady-state start time is left to future works.

Third, we showed how to transform network dynamics into finite spatial-temporal graph representations using the SS-S to bound the construction. The graph representation we constructed reflects one of several representations to describe the dynamics [17,18,22].

While the methods in this paper have application to a number of systems, we suspect that larger and/or more complex networks–including networks with variations in parameters over long time scales–still pose practical challenges to finding the SS-S. For example, in some models of biological spiking neural networks forward calculation of the network dynamics until stead state is reached may be computationally infeasible because the inception of the steady state regime has an exponential dependence on the network's size [27,50]. This limitation can be partly mitigated if one is only interested in transient stability, which in this context we define as the periodic network dynamics over some time interval and of some subset of the entire network. Our proposed methods for finding the SS-S can easily be extended to detect transient stability by defining observational sets of interest.

A potentially fruitful future direction for this work is to automate the stopping criteria for forward computation of complex networks. If steady state detection could be performed in parallel with forward computation, periodicity could be used to stop the forward computation of network dynamics. This might be accomplished with an online, efficient algorithm for string repetition detection, that is, one that does not require recomputing large data structures to incorporate newly appended symbols to the string [64].

Another extension of the string search algorithm is to incorporate additional observed network states and parameters. These could be, for example, causal node activators, weighted synaptic signal contributions, times between activations, etc. This can be done by extending the library of symbols included in the string and paying a relatively small computational penalty for increasing the complexity to $O(kn)$, where k is the symbol length. For example, in the case of adding causal node activators, let us assume there are, on average, three activators required per activation. If so, we would extend the size of the string by a multiple of four. Any parameter that can be translated into a symbolic representation can be included in the string representation for analysis. It is up to the user to determine the appropriate trade-offs between the parameters for inclusion and the computational costs associated with a longer string.

Although the graph representation of the network activity is agnostic to the steady state finding process, the number of observations that need to be included to build a finite and compact graph needs to be determined before the method can be applied. By bounding the network dynamics, various network models and parameters can be evaluated. Using the techniques developed in this work, we can quantitatively differentiate between network activity regimes and represent network dynamics in a normalized manner.

Funding statement

This work was supported by unrestricted funds to the Center for Engineered Natural Intelligence at the University of California San Diego.

CRediT authorship contribution statement

Vivek Kurien George (VKG); Arkin Gupta (AG); and Gabriel A. Silva (GAS): Analyzed and interpreted the data and wrote the paper.
Vivek Kurien George (VKG); Arkin Gupta (AG): Performed the experiments. Contributed reagents, materials, analysis tools or data.
Vivek Kurien George (VKG): Conceived and designed the experiments.

Declaration of competing interest

The authors declare no competing interests.

Data availability

No data was used for the research described in the article.

Acknowledgements

The code for runFinder is courtesy of Dr. Hideo Bannai (https://github.com/hdbn/runfinder).

References

[1] E. Bullmore, O. Sporns, Complex brain networks: graph theoretical analysis of structural and functional systems, Nat. Rev. Neurosci. 10 (3) (2009) 186–198.
[2] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, D.-U. Hwang, Complex networks: structure and dynamics, Phys. Rep. 424 (4–5) (2006) 175–308.
[3] S. Wasserman, K. Faust, Social Network Analysis: Methods and Applications, Social Network Analysis: Methods and Applications, Cambridge University Press, 1994.
[4] D.S. Bassett, A.N. Khambhati, S.T. Grafton, Emerging frontiers of neuroengineering: a network science of brain connectivity, Annu. Rev. Biomed. Eng. 19 (1) (2017) 327–352.
[5] N. Deo, Graph theory with applications to engineering and computer science, Networks 5 (3) (1975) 299–300.
[6] L.d.F. Costa, O.N. Oliveira Jr., G. Travieso, F.A. Rodrigues, P.R.V. Boas, L. Antiqueira, M.P. Viana, L.E.C. Rocha, Analyzing and modeling real-world phenomena with complex networks: a survey of applications, Adv. Phys. 60 (3) (Jun. 2011) 329–412.
[7] C. Donnat, S. Holmes, Tracking network dynamics: a survey using graph distances, Ann. Appl. Stat. 12 (2) (Jun. 2018) 971–1012.
[8] M. Salehi, R. Sharma, M. Marzolla, M. Magnani, P. Siyari, D. Montesi, Spreading processes in multilayer networks, IEEE Trans. Netw. Sci. Eng. 2 (2) (Apr. 2015) 65–83.
[9] M.E. Newman, Spread of epidemic disease on networks, Phys. Rev. E 66 (1) (2002) 016128.
[10] M. Kivela, A. Arenas, M. Barthelemy, J.P. Gleeson, Y. Moreno, M.A. Porter, Multilayer networks, J. Complex Netw. 2 (3) (Sep. 2014) 203–271.
[11] S. Boccaletti, G. Bianconi, R. Herrero, C. Genio, J. Gómez-Gardeñes, M. Romance, I. Sendiña-Nadal, Z. Wang, M. Zanin, The structure and dynamics of multilayer networks, Phys. Rep. 544 (Nov. 2014) 1–122.
[12] Z. Wang, C. Xia, Co-evolution spreading of multiple information and epidemics on two-layered networks under the influence of mass media, Nonlinear Dyn. 102 (4) (2020) 3039–3052.
[13] Z. Wang, C. Xia, Z. Chen, G. Chen, Epidemic propagation with positive and negative preventive information in multiplex networks, IEEE Trans. Cybern. 51 (3) (2020) 1454–1462.
[14] Q. Yin, Z. Wang, C. Xia, M. Dehmer, F. Emmert-Streib, Z. Jin, A novel epidemic model considering demographics and intercity commuting on complex dynamical networks, Appl. Math. Comput. 386 (2020) 125517.
[15] C. Brennan, A. Proekt, A quantitative model of conserved macroscopic dynamics predicts future motor commands, eLife 8 (Jul. 2019) e46814.
[16] K. Morrison, C. Curto, Predicting neural network dynamics via graphical analysis, in: Algebraic and Combinatorial Computational Biology, Elsevier, 2019, pp. 241–277.
[17] J.M. Roldan, S.P. G, V.K. George, G.A. Silva, Construction of edge-ordered multidirected graphlets for comparing dynamics of spatial temporal neural networks, arXiv:2006.15971, 2020.
[18] V.K. George, F. Puppo, G.A. Silva, Computing temporal sequences associated with dynamic patterns on the *C. elegans* connectome, Front. Syst. Neurosci. 15 (2021) 15.
[19] P.Y. Wang, S. Sapra, V.K. George, G.A. Silva, Generalizable machine learning in neuroscience using graph neural networks, Front. Artif. Intell. 4 (2021) 4.

[20] M. Buibas, G.A. Silva, A framework for simulating and estimating the state and functional topology of complex dynamic geometric networks, Neural Comput. 23 (1) (2011) 183–214.
[21] G.A. Silva, The effect of signaling latencies and node refractory states on the dynamics of networks, Neural Comput. 31 (12) (2019) 2492–2522.
[22] C. Curto, What can topology tell us about the neural code?, Bull. Am. Math. Soc. 54 (1) (Sep. 2016) 63–78.
[23] H. Ju, D.S. Bassett, Dynamic representations in networked neural systems, Nat. Neurosci. 23 (8) (2020) 908–917.
[24] M.D. Humphries, Dynamical networks: finding, measuring, and tracking neural population activity using network science, Netw. Neurosci. 1 (4) (2017) 324–338.
[25] O. Sporns, Graph theory methods: applications in brain networks, Dialogues Clin. Neurosci. 20 (2) (2018) 111.
[26] C. Giusti, R. Ghrist, D.S. Bassett, Two's company, three (or more) is a simplex, J. Comput. Neurosci. 41 (1) (2016) 1–14.
[27] B. Cessac, A discrete time neural network model with spiking neurons: rigorous results on the spontaneous dynamics, J. Math. Biol. 56 (3) (Nov. 2007) 311–345 [Online]. Available: http://link.springer.com/10.1007/s00285-007-0117-3.
[28] P.A. Anninos, Cyclic modes in artificial neural nets, Kybernetik 11 (1) (Jul. 1972) 5–14.
[29] E.N. Brown, R.E. Kass, P.P. Mitra, Multiple neural spike train data analysis: state-of-the-art and future challenges, Nat. Neurosci. 7 (5) (May 2004) 456–461.
[30] T. Donoghue, M. Haller, E.J. Peterson, P. Varma, P. Sebastian, R. Gao, T. Noto, A.H. Lara, J.D. Wallis, R.T. Knight, et al., Parameterizing neural power spectra into periodic and aperiodic components, Nat. Neurosci. 23 (12) (2020) 1655–1665.
[31] M.A. McClarnon, Detection of steady state in discrete event dynamic systems: an analysis of heuristics, Ph.D. dissertation, University of Virginia, 1990.
[32] A.M. Law, W.D. Kelton, W.D. Kelton, Simulation Modeling and Analysis, vol. 3, McGraw-Hill, New York, 2000.
[33] M. Patel, N. Modi, A comprehensive study on periodicity mining algorithms, in: 2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication, ICGTSPICC, IEEE, 2016, pp. 567–575.
[34] M.G. Elfeky, W.G. Aref, A.K. Elmagarmid, Periodicity detection in time series databases, IEEE Trans. Knowl. Data Eng. 17 (7) (2005) 875–887.
[35] F. Rasheed, M. Alshalalfa, R. Alhajj, Efficient periodicity mining in time series databases using suffix trees, IEEE Trans. Knowl. Data Eng. 23 (1) (2010) 79–94.
[36] W.F. Smyth, Computing regularities in strings: a survey, Eur. J. Comb. 34 (1) (Jan. 2013) 3–14.
[37] M. Crochemore, C. Hancart, T. Lecroq, Algorithms on Strings, Cambridge University Press, 2007.
[38] H. Koponen, N. Mhaskar, W. Smyth, An overview of string processing applications to data analytics, in: 2021 Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge, RDAAPS, IEEE, 2021, pp. 1–8.
[39] M.G. Main, Detecting leftmost maximal periodicities, Discrete Appl. Math. 25 (1) (Oct. 1989) 145–153.
[40] F. Franek, W. Smyth, X. Xiao, A note on crochemore's repetitions algorithm-a fast space-efficient approach, Nord. J. Comput. 10 (1) (2003) 21–28.
[41] M. Crochemore, C.S. Iliopoulos, M. Kubica, J. Radoszewski, W. Rytter, K. Stencel, T. Waleń, New simple efficient algorithms computing powers and runs in strings, Discrete Appl. Math. 163 (Jan. 2014) 258–267.
[42] D. Kosolobov, Computing runs on a general alphabet, Inf. Process. Lett. 116 (3) (Mar. 2016) 241–244.
[43] R. Kolpakov, G. Kucherov, Finding maximal repetitions in a word in linear time, in: 40th Annual Symposium on Foundations of Computer Science, Cat. No. 99CB37039, IEEE Comput. Soc., New York City, NY, USA, 1999, pp. 596–604.
[44] M. Crochemore, L. Ilie, L. Tinta, The "runs" conjecture, Theor. Comput. Sci. 412 (27) (2011) 2931–2941.
[45] H. Bannai, T. I, S. Inenaga, Y. Nakashima, M. Takeda, K. Tsuruta, The "runs" theorem, SIAM J. Comput. 46 (5) (Jan. 2017) 1501–1514.
[46] R.C. Fuller, Performance comparisons of various runs algorithms, Ph.D. dissertation, McMaster University, Ontario, 2012.
[47] M.E. Newman, The structure and function of complex networks, SIAM Rev. 45 (2) (2003) 167–256.
[48] F. Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain, Psychol. Rev. 65 (6) (1958) 386–408.
[49] E. Izhikevich, Which model to use for cortical spiking neurons?, IEEE Trans. Neural Netw. 15 (5) (Sep. 2004) 1063–1070.
[50] B. Cessac, T. Viéville, On dynamics of integrate-and-fire neural networks with conductance based synapses, Front. Comput. Neurosci. 2 (Jul. 2008).
[51] B. Cessac, H. Paugam-Moisy, T. Viéville, Overview of facts and issues about neural coding by spikes, J. Physiol. (Paris) 104 (1–2) (Jan. 2010) 5–18.
[52] R. Brette, M. Rudolph, T. Carnevale, M. Hines, D. Beeman, J.M. Bower, M. Diesmann, A. Morrison, P.H. Goodman, F.C. Harris, M. Zirpe, T. Natschläger, D. Pecevski, B. Ermentrout, M. Djurfeldt, A. Lansner, O. Rochel, T. Vieville, E. Muller, A.P. Davison, S. El Boustani, A. Destexhe, Simulation of networks of spiking neurons: a review of tools and strategies, J. Comput. Neurosci. 23 (3) (Dec. 2007) 349–398.
[53] S. Panzeri, R.A. Ince, M.E. Diamond, C. Kayser, Reading spike timing without a clock: intrinsic decoding of spike trains, Philos. Trans. R. Soc. Lond. B, Biol. Sci. 369 (1637) (2014) 20120467.
[54] C.-C. Weng, Implementing efficient algorithms for computing runs, Ph.D. dissertation, McMaster University, Ontario, 2011.
[55] D. Gusfield, Algorithms on strings, trees, and sequences by dan gusfield [Online]. Available: /core/books/algorithms-on-strings-trees-and-sequences/F0B095049C7E6EF5356F0A26686C20D3, May 1997.
[56] G. Chen, S.J. Puglisi, W.F. Smyth, Fast and practical algorithms for computing all the runs in a string, in: B. Ma, K. Zhang (Eds.), Combinatorial Pattern Matching, in: Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2007, pp. 307–315.
[57] S. Puglisi, W. Smyth, M. Yusufu, Fast optimal algorithms for computing all the repeats in a string, in: Prague Stringology Conference 2008, 2008.
[58] U. Manber, G. Myers, Suffix arrays: a new method for on-line string searches, SIAM J. Comput. 22 (5) (Oct. 1993) 935–948.
[59] Y. Mori, A lightweight suffix-sorting library. Contribute to y-256/libdivsufsort development by creating an account on GitHub, Aug. 2019, original-date: 2015-03-17T15:30:25Z.
[60] J. Kärkkäinen, P. Sanders, Simple linear work suffix array construction, in: International Colloquium on Automata, Languages, and Programming, Springer, 2003, pp. 943–955.
[61] J. Ziv, A. Lempel, Compression of individual sequences via variable-rate coding, IEEE Trans. Inf. Theory 24 (5) (Sep. 1978) 530–536.
[62] M. Crochemore, L. Ilie, W.F. Smyth, A simple algorithm for computing the Lempel Ziv factorization, in: Data Compression Conference, dcc 2008, Mar. 2008, pp. 482–488.
[63] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[64] J.-J. Hong, G.-H. Chen, Efficient on-line repetition detection, Theor. Comput. Sci. 407 (1–3) (2008) 554–563.

Vivek Kurien George received his B.Sc. degrees in Electrical Engineering and Mathematics from the Florida Institute of Technology and a Ph.D. in Bioengineering with a specialization in Computational Neuroscience from the University of California at San Diego. He is currently a Postdoctoral Fellow at the Center for Natural and Engineered Intelligence at the University of California San Diego. His current research interests include developing artificial intelligence and machine learning algorithms using neurobiological principles, graph theory, and combinatorics.

Arkin Gupta received his B.Sc. in Mathematics and Computer Science from University of California at San Diego. During his time at UCSD he worked at the Center for Engineered Natural Intelligence where he focused on developing and analyzing neurobiological machine learning models. Arkin currently works as a Quantitative Researcher for a systematic hedge fund.

Gabriel A. Silva is a Professor in the Department of Bioengineering and the Department of Neurosciences at the University of California San Diego. He holds a Jacobs Family Scholar in Engineering Endowed Chair, is the Founding Director of the Center for Engineered Natural Intelligence, and Associate Director of the Kavli Institute for Brain and Mind. He received an Hon.B.Sc. in Human Physiology and a B.Sc. in Biophysics from the University of Toronto, Canada, followed by an M.Sc. in Neuroscience also from the University of Toronto. He then did his Ph.D. in Bioengineering and Neurophysiology at the University of Illinois at Chicago, followed by a postdoctoral fellowship in the Institute for BioNanotechnology and Medicine (IBNAM) and the Department of Neurology at Northwestern University. He joined the faculty at the University of California, San Diego in 2003.