Computational Elements of Circuits

Johannes Leugering, Pascal Nieters, and Gordon Pipa

Abstract

Information processing in the brain is implemented across several temporal and spatial scales by populations of neurons. This chapter addresses how single neurons, small network motifs, and larger networks, in which emergent dynamics are largely shaped by the connectivity of the system, contribute to this processing of information. Computation is defined as a semantic mapping; that is, it is the process by which representations of external (e.g., stimulus-driven) or internal (e.g., memories) information change. A feature specific to neuronal computation is that mappings are mostly local, constrained by connectivity patterns between neurons. This implies that complex mappings from local information onto representations that are highly relational and abstracted, and which rely on information between distant parts of the system, require mechanisms that can bridge, bind, and integrate pieces of information across large scales. An overview of this process in the nervous system is delineated: Local information processing is described at the level of individual neurons and small motifs. Emergent phenomena are addressed that implement information processing across large recurrent neuronal populations. Finally, an omnipresent but mostly ignored feature of neuronal systems, delay-coupled computation, is described.

Information Processing in Single Neurons and Populations

An understanding of how information is processed in neural systems begins with a consideration of how an individual neuron perceives and processes information, before extending this scope gradually to larger systems. Our goal in this chapter is to present a concise, abstract view of computation in neural systems, understood to be key to a meaningful change in the representation of information. In the interest of brevity, the biological complexity of neurons and networks (e.g., the role of specific ion channels or the potential influence of glia cells and neuromodulators) will not per se be addressed.

From "The Neocortex," edited by W. Singer, T. J. Sejnowski and P. Rakic.
Strüngmann Forum Reports, vol. 27, J. R. Lupp, series editor.
Cambridge, MA: MIT Press. ISBN 978-0-262-04324-3

A Stochastic Process Linear-Nonlinear Neuron Model

From the perspective of the linear-nonlinear (LN) model, a neuron is a computational unit that receives a multivariate time-varying input signal through its synaptic inputs and generates a univariate time-varying output signal. This mapping from input to output signals is near instantaneous (at least time-invariant), as the neuron itself is assumed to have, at most, a very limited internal memory[1] and be subject to noise.

In the mathematical framework of stochastic processes, a neuron can thus be concisely described as a nonlinear, causal, time-invariant operator that maps a multivariate stochastic process onto a univariate stochastic process. We make several simplifying assumptions that result in a convenient class of neuron models (Ostojic and Brunel 2011; see also Figure 11.1):

- The neuron's operation can be modeled as a leaky integrator or, even simpler, an instantaneous input-output mapping.
- It is composed of a linear operator, which reduces the multivariate input arriving at different synapses along the dendritic tree to a univariate input to the neuron's soma, followed by a nonlinear transformation.
- The linear operation is parameterized by synaptic weights, which can be positive or negative.
- The nonlinear transformation, which we refer to as the activation function or just nonlinearity, is a monotonically increasing, (locally) differentiable and bounded function.

While the activation could be further used in a spike generation process as an instantaneous firing rate, we treat it here as the neuron's continuous state or output. Each neuron in a population independently processes its own input (which may be correlated to other neurons' inputs), and its state provides one component of the entire population's multivariate state. The computation carried out by a population of neurons, mapping a multivariate input signal onto a multivariate state, must thus arise component-wise from the computations realized in the individual neurons. Each neuron, however, is limited to those operations which can be realized by a LN model under the above constraints.

To better understand the capabilities and limitations of this class of models, it helps to analyze them from a machine learning perspective, where such models commonly appear under different guises and names.

[1] The exception to this rule is found in plasticity mechanisms, which we assume to operate on a much slower, separate timescale than that of the output signal, and thus they can be treated as virtually constant in this context. The commonly made assumption of near instantaneous operation of the neuron further presumes that slower active dendritic processes do not substantially contribute to computation, which can be called into question and may turn out to be an overly simplistic perspective.

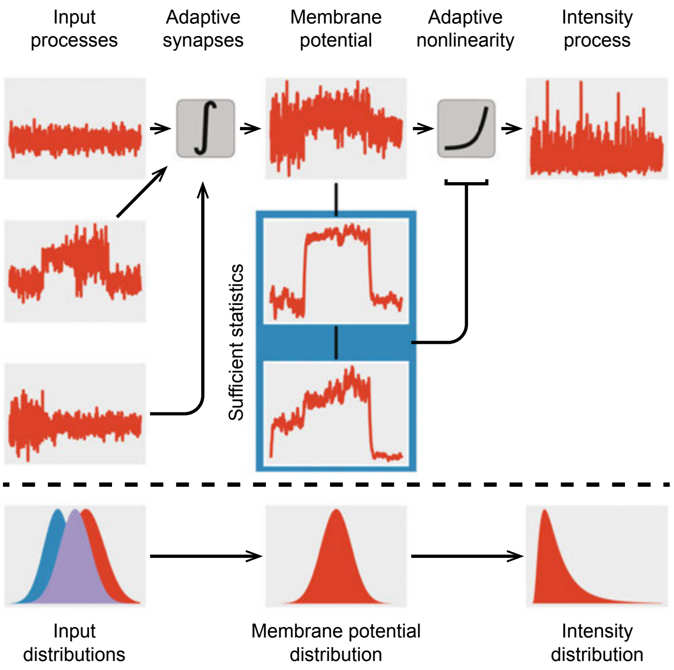

Figure 11.1 A model neuron receives a linear combination of multiple time-varying stochastic processes that are scaled by adaptive, synaptic weights and integrated into the neuron's membrane potential. By sensing some sufficient statistics of the membrane potential, the neuron's nonlinearity can be adjusted to achieve an activation (or intensity) with desirable statistical properties. Assuming stationarity of the input processes, the neuron's nonlinearity can be determined by the desired mapping from the univariate membrane potential distribution to an intensity distribution. Adapted from Leugering and Pipa (2018).

LN Models in Machine Learning

Using a Heaviside function for the nonlinearity, LN models appear in machine learning in the form of linear hard-margin classifiers, such as the classical perceptron (Rosenblatt 1958), linear support vector machines (Hearst et al. 1998), or depth-one decision trees (so-called "decision stumps"; Criminisi et al. 2012). With continuous nonlinearities, such as the logistic function, these models can be used as soft-margin classifiers and regressors, as in generalized linear models (GLMs) (McCullagh and Nelder 1989), where the nonlinearity is used to relate a linear combination of input features to the expected value of the (task-specific) label associated with the data. To improve performance, multiple instances of such models can be combined laterally to form an ensemble, used in a boosting procedure or stacked hierarchically, like the layers of an artificial neural network (Hopfield 1988) or the levels of a decision tree.

Computation in this context simply refers to the ability of the model to encode specific task-relevant information about its inputs into its output. The same claim has been made for individual biological neurons, as well as whole layers of neurons in deep networks under the “information bottleneck” principle. The deceptively simple argument is that each neuron (or each layer of a network, respectively) is presented with a high-dimensional input signal that carries task-relevant, as well as irrelevant, information, and, in a noisy environment with limited capacity to transmit information, ought to transform it into an informative low-dimensional output signal (Becker 1996).

Supervised Learning

In a supervised setting, where the desired output of the model is known at all times, the extraction and transmission of task-relevant information with simultaneous suppression of task-irrelevant “noise” represents a form of lossy compression. In multilayer networks, backpropagation can provide a supervised error signal for each layer and ultimately each neuron, thus allowing it to locally solve a lossy compression problem, which has been hypothesized as the theoretical mechanism underlying the surprising success of deep neural networks (Shwartz-Ziv and Tishby 2017).

Unsupervised Learning

The concept and potential mechanisms of error backpropagation in biological neural networks, however, are controversial, and the existence of supervised target signals may be called into question altogether. In the absence of supervision, the information bottleneck principle can be restated as the objective for each neuron to simply encode its inputs into its output in the most informative way possible, since it cannot distinguish task-relevant from irrelevant information.

The information encoding of the output signal is reflected in the firing statistics, with heavy-tailed firing rate distributions corresponding to sparse spiking codes, and narrowly peaked distributions corresponding to tonic firing or bursting codes. By driving synaptic plasticity, this can, in turn, shape the topology of synaptic connections and lead to the formation of specific motifs, thus allowing a population of neurons to implement task-relevant computation without supervision.

Under the biological constraints imposed on the neuron (e.g., bounded firing rates, energy limitations), the mutual information between input and output is bounded by the entropy attainable by the output distribution. A common objective is thus for the neuron to enforce a maximum entropy distribution of its outputs by appropriately adjusting its nonlinearity, while simultaneously tuning its synaptic connection weights to project the multidimensional input signal onto the most informative subspace. Equivalently, for a population of neurons, the information bottleneck objective is to realize a maximum entropy joint distribution, such that each marginal distribution of an individual neuron's output satisfies the biological constraints.

Unsupervised Learning Application: Independent Component Analysis

As it turns out, this objective fully determines a unique optimal choice of nonlinearity for a given family of input distributions and a desired output distribution. It also implies that the linear subspaces selected by the neurons' respective synaptic input weights should correspond to the main independent components. Consequently, this problem is also referred to as independent component analysis (ICA), a generalization of principle component analysis which can no longer be solved by linear methods (Hyvärinen and Oja 1998; Triesch 2007).

This intuition transfers seamlessly to a framework of stochastic processes (Leugering and Pipa 2018), where a population is tasked with mapping its (stationary) multivariate input process onto a multivariate output process, with a joint distribution composed of independent components with given marginal distributions. By factoring the population's joint distribution into its marginal distributions and a copula function, it becomes apparent that this objective can be achieved through the interaction of two distinct mechanisms:

1. The copula function captures all of the dependency structure present in the joint distribution and depends only on the choice of synaptic input weights of the population; thus it can be adjusted by synaptic plasticity.
2. The marginal distribution of each neuron's output can be enforced purely by an appropriate choice of nonlinearity; thus it can be adjusted by intrinsic plasticity.

Since all of the information required to solve the ICA problem is available locally to the neurons or their synapses, it can be solved by the LN model discussed above using simple, biologically plausible mechanisms of intrinsic and synaptic plasticity in a time-continuous, noisy setting.

Using motifs of several laterally inhibiting neurons, different independent components can be found, leading to a highly informative, multivariate output signal. As shown in Figure 11.2, such a structure can be used to learn, in an unsupervised fashion, to classify MNIST images with just a handful of neurons. For an in-depth discussion of this result, see Leugering and Pipa (2018).

Computation in Networks Using Emergent Properties

The cerebral cortex is a highly distributed system with reciprocal connections that shape neuronal activity through self-organizing and that can create

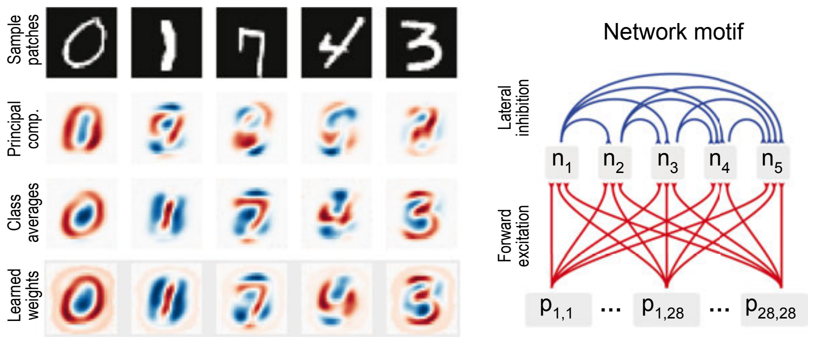

Figure 11.2 A small motif of five neurons receives feedforward excitation from 28 × 28 neurons, representing pixels of visual inputs. Images from the MNIST database are presented successively, while the synaptic weights and each neuron's nonlinearity are adjusted by local synaptic and intrinsic plasticity, respectively. Only the combination of both learning mechanisms leads to the (unsupervised) discovery of independent components in the input space, corresponding to the average input for each class. Lateral inhibition learns to decorrelate the neurons and ensures that different components are discovered, reducing redundancy and thus maximizing the information content of the motif's output. Results from Leugering and Pipa (2018).

coherent states able to encode representations of sensory objects, decisions, and programs for motor acts (Uhlhaas et al. 2009). The topology of the connectivity shares properties with small world networks having no singular center where all information converges (Gerhard et al. 2011). This raises questions of how the numerous computations on the level of single neurons are coordinated and bound together to give rise to coherent percepts and actions, and how relations between simultaneously represented contents can be encoded. One option is that neuronal synchrony can implement both features. Several mechanisms have been discussed—for instance mediated by inhibitory synapses, enhanced via gap junctions, induced by motifs of neuronal connectivity (Vicente et al. 2008; Pérez et al. 2011; Messé et al. 2018)—that can induce neuronal synchronous firing even despite long conduction delays. However, one of the central challenges that has not been sufficiently addressed is that the mechanism needs to enable the neurons to synchronize and desynchronize in a stimulus-specific fashion, and thereby to encode relationships. Noise-induced coherence is one such mechanism that was recently demonstrated to produce fast, stimulus-specific, and biologically plausible synchronization patterns.

First discussed in complex and excitable systems (Pikovsky and Kurths 1997), noise-induced coherence is a process that can structure and synchronize the activity of the system based on noisy or even unstructured input. The nature of noise-induced coherence is that the complex system (the dynamical elements, e.g., neurons, together with the network topology) defines patterns that exhibit enhanced coherence if the system is driven by a corresponding motif, neuron-specific optimal amplitude of unstructured noise. In other words, and in respect to neuronal networks, noise translates a pattern of neuron-specific firing rates into patterns of coherent and synchronized population responses (i.e., translation of a neuron-specific rate code to a population-based sync code). Importantly, this translation is network specific, which opens the possibility that the expression of synchronous events is not only driven by the stimulus-specific rate pattern but also by the network, and its structure is shaped by neuronal plasticity.

Transformation of Spike Rate Coding to Coherent Population Codes via Noise-Induced Coherence

To illustrate the mechanism and encoding based on noise-induced coherence, let us consider an example for the visual cortex V1. In general, it is known that network structure of cortical networks is at least partially shaped by the experience of past activation mediated by neuronal plasticity. For V1, this implies that the connection strength horizontal connections in V1 reflect the aggregate statistics of natural visual scenes (Onat et al. 2013); that is, V1 cells with nearby receptive fields are preferentially connected, and specifically when they select for similar visual stimuli. Figure 11.3a shows a network simplified to such a V1 prototypic connectivity pattern. The system receives stimulus-specific input described by neuron-specific retinal coordinates which match their cortical position (retinotopy) and have a particular angle (orientation tuning) presynaptic spike rates (i.e., uncorrelated and rate-modulated Poisson firing). To illustrate the effect of noise-induced coherence, we use two kinds of stimuli: one that is open and composed of two shorts blocks, and one that is closed and composed of a longer bar. Given the retinotopic mapping, this implies that the activation pattern, in comparison to the underlying network, results in different shortest path lengths between stimulus-driven cells. Only few of these cells will have direct connections, since horizontal connections preferably connect cells with nearby receptive fields. More generally, the network connectivity implies a metric for possible stimulus patterns. Given this metric, for V1, the shortest path between any two responding cells will likely be longer, on average, for a scattered stimulus than for a more compact stimulus. As a result, the same cells which receive a presynaptic input pattern matching the connectivity of the network (here, cells that are part of a continuous patch) exhibit stronger noise-induced coherence than others. Such mechanisms can be generalized to more complex encoding schemes, depending on the connectivity patterns of the network. For example, the well-known orientation tuning of cells in V1, in combination, and network motifs described by preferred connectivity across cells with similar orientation will result in enhanced coherence of neurons that encode chains of shorter line segments (see Figure 11.4). In general, noise-induced coherence is a mechanism that can measure the similarity between the network connectivity and the stimulus-induced spike rate pattern (Korndörfer et

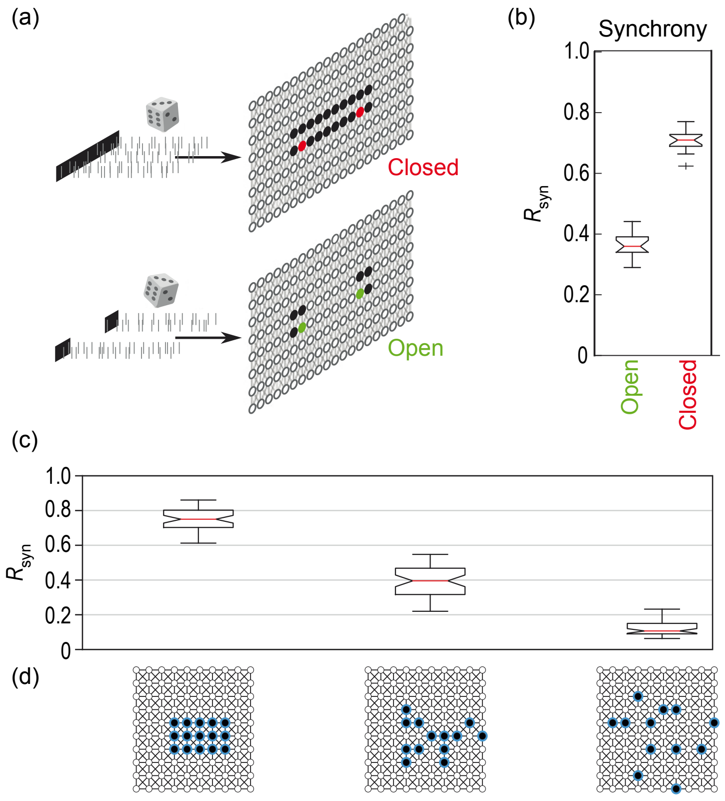

Figure 11.3 (a) Noise-induced coherence for two alternative presynaptic stimuli (red, a closed line; green, an open line segment). Coherence is measured between the green- or red-highlighted neurons. The only difference is the context given by the stimulus drive, which itself is composed of unstructured Poisson noise. The network topology is defined by nearest neighbor connection matches, and stimuli are matched using retinotopic mapping. (b) Synchrony measure of the pairs of neurons shown in (a) and for the two stimulus conditions. Coherence is higher for the compact closed line, since the shortest path length between stimulus-driven neurons is smaller for the closed contour. (d) This feature is generalized to the amount of scattering of a stimulus; that is, the greater the scattering, the larger the shortest path length between neurons, given the metric of the underlying network connectivity. The resulting stimulus-induced coherence (c) is the largest for the most compact, and the lowest, for the most scattered stimulus. Adapted from Korndörfer et al. (2017).

al. 2017). It can therefore be a measure of how well the stimulus matches a prior learned by neuronal plasticity and encoded in the network's connectivity. Here the stimulus-induced spike rate reflects a classical labeled line code. Thus, spike synchrony generated by noise-induced coherence carries synergistic information that reflects to which degree the current stimulus encoded by the spike rate is expected, given past stimulus experiences. Such a signal could be used early after input onset in a feedforward fashion, for instance,

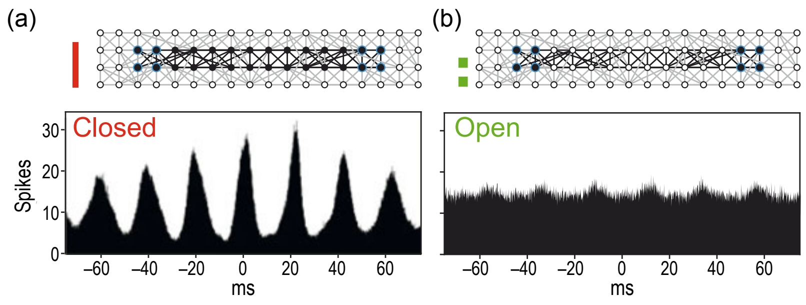

Figure 11.4 Average cross correlation of noise-induced coherence between two sets of neurons marked in blue for two different stimulus conditions: (a) closed line segment and (b) open line segment. The noise-induced coherence is stronger for the closed, compared to the open, condition. In the original publication (Korndörfer et al. 2017), it is shown that this increased coherence can be decoded as closed contour as early as a few spikes after stimulus onset (70 ms).

to guide attention toward stimuli composed of plausible parts. In contrast to many other types of synchronization, it also does not require, but can be improved by, inhibitory cells (Korndörfer et al. 2017), and it produces firing patterns that closely resemble *in vivo* recorded patterns (e.g., Gray et al. 1989; Uhlhaas et al. 2009).

Reservoir Computing

Most mechanisms discussed over the past decades for neuronal information processing require highly structured networks, specific types of dynamical processes, and very specific encoding schemes of information (e.g., rate code versus population spike codes). A frequently used feature of computational models is that they rely on attractor dynamics, which can be trained to implement specific computational features, such as associated memory in Hopfield networks (Hopfield 1982) or the winner-takes-all mechanism (Maass 2006) for decision making, for example.

Like the ICA network discussed above, all of these computational models implement a clearly defined information processing principle and rely on a very specific type of implementation, in terms of connectivity and dynamical elements. This is a strong advantage, since it allows us to study principle and well-defined behavior, and to reduce the computation to a minimal set of required properties. At the same time, this reductionism also renders the models biologically implausible, since biological systems are subject to noise on pretty much any property, such that neuronal networks are mostly random with some statistical preferences for certain motifs, and neurons are diverse in type and morphology.

Therefore, a strikingly different model for neuronal computation is *reservoir computing*, originally introduced as liquid state machines by Maass et al. (2002) or echo-state networks by Jaeger and Haas (2004). In contrast to most other computational principles, the recurrent network of a reservoir computer can be unstructured and random. This surprising property results from the simple insight that the distance between random mappings of states is growing fast, with increasing dimensionality of the mapping. In other words, implementing a certain computation does not require a dedicated network with specific connectivity tailored for the given task but, in principle, only a random network that implements a sufficiently high-dimensional random mapping. In the field of machine learning, this is known as feature expansion or kernel machines (Schölkopf and Smola 2002). Further, reservoir computing makes explicit use of the recurrence of neuronal networks to maintain an echo (i.e., memory capacity) of past inputs. This echo is mediated by reverberating activity, generated by the recurrent connectivity. Together, feature expansion and memory of the system can render a reservoir computer a universal computer (Buonomano and Maass 2009). The only task-specific element in reservoir computing is a task-specific mapping that can be learned by supervised, semi-supervised (Toutounji and Pipa 2014), or reinforcement learning algorithms (Aswolinskiy and Pipa 2015).

The remarkable insight of reservoir computing is that random recurrent networks can implement, in principle, any kind of computation if the networks are sufficiently complex. From a biological point of view, this implies that initially unstructured networks can bootstrap themselves, based on neuronal plasticity, to improve performance. Importantly, it can operate initially even without any structure.

Computation in Delay-Coupled Systems

When describing computation in the nervous system from the perspective of abstract single neurons or recurrent networks which show emergent behavior as a collective, a simplifying assumption is often made: interactions between neurons are instantaneous and not delayed. This is simply because delays in differential equations complicate the analysis of such systems significantly, and deriving theoretical results is a lot harder.

In biophysical reality, however, the brain is a network of nodes and wires that must be subject to transmission delays. For instance, conduction delays of tens of milliseconds occur in axonal transmission of spikes (Ringo et al. 1994). Interspike intervals, indicative of the timescales on which neurons compute outputs, have been found on the same scale in the motor system (Calvin and Stevens 1968) or in retinal ganglion cells (Levine and Shefner 1977). It is thus clear that delays play a role in the dynamics and computational properties of neural networks. A long-established example, where this role is well understood, is audio processing: transmission delays on delay lines are used to distinguish left ear input from right ear input, and interpolate the location of a sound source (London and Häusser 2005).

In cortical structures, network motifs or microcircuits have been found that circumvent transmission delays and lead to zero time lag synchronization (Vicente et al. 2008). On the other hand, in microcircuits where transmission delays are modeled, only very specific topologies allow for coherent spiking activity, which delays control phase differences between oscillatory neurons (Pérez et al. 2011). So, locally, transmission delays control phase transitions between in-phase and out-of-phase response, whereas, globally, axonal delays can stabilize coherent response-important phenomena in neural computation.

Even still, these examples only describe how delays can negatively impact behavior of microcircuits or stabilize existing behavior. Future work should investigate the degree to which the added complexity of delay-coupled systems can be exploited for computation.

A Single Node with Delayed Feedback

Stabilizing emergent phenomena may not be the only mechanism by which delays can aid computation in the nervous system. Instead, the benefit of delayed interactions can be illustrated theoretically by examining a single computational node with delayed feedback. This very simple setup is described by a delay differential equation:

$$dx(t) = f\left(x(t),\, x(t-\tau)\right)dt. \tag{11.1}$$

The equation can be solved by a trick known as the method of steps (Guo and Wu 2013), which is both intuitive and illustrative of the complexity of delayed interactions: Assume that the solution to Equation 11.1 on some interval, $[t_0 - \tau, t_0]$, is known and denote that solution ϕ_0. For the subsequent overlapping interval, $[t_0, t_0 + \tau]$, Equation 11.1 can then be rewritten as

$$dx(t) = f\left(x(t),\, \phi_0(t-\tau)\right)dt, \tag{11.2}$$

since for all $t \in [t_0, t_0 + \tau]$, it holds that $t - \tau \in [t_0 - \tau, t_0]$, where ϕ_0 is the solution. This is now an ordinary differential equation and can be solved using traditional methods. However, the starting condition for the new solution is now a tuple $(\phi_0, \phi_0(t_0))$ of a function, and the function is evaluated at t_0. Further, the solution on the interval $[t_0, t_0 + \tau]$ is again a function; let that function be ϕ_1. In the method of steps, this procedure is iterated with this new starting value for the next interval of length τ. In general, if $t_i = t_0 + i\tau$, then ϕ_i is the solution on the interval $[t_{i-1}, t_i]$.

Even though Equation 11.1 is a differential equation of a single, scalar variable, solving it involves mapping functions onto functions for each τ interval or cycle (Figure 11.5a) and is therefore infinitely dimensional. Delay differential equations are a subclass of partial differential equations whose state is described by functions, instead of finite-dimensional state vectors.

By introducing one simple, delayed feedback to a dynamical system, we elevate the complexity from one to infinitely many dimensions. This complexity

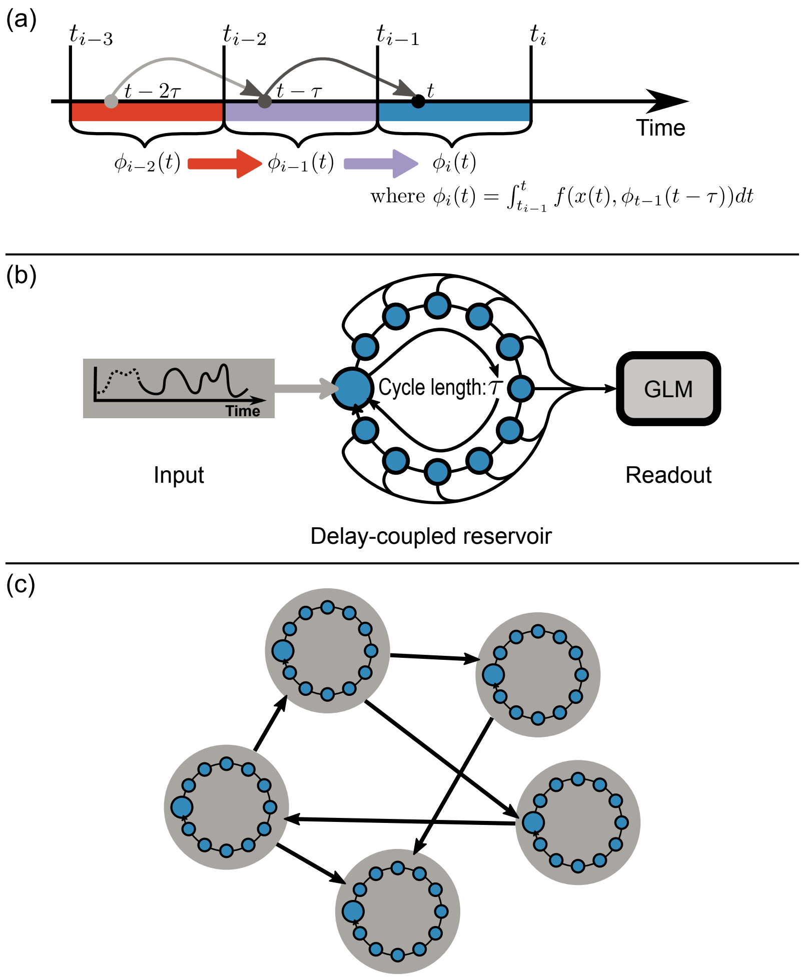

Figure 11.5 (a) Schema of solving delay differential equations with the method of steps. Functional solutions ϕ_i are mapped onto solutions ϕ_{i+1} via an integral over the original delay differential equations, where the delay dependency is replaced by a dependency on the last solution. (b) A delay-coupled reservoir utilizes the complexity of delay differential equations for computation by creating an input-driven dynamical system and feeding sampled activity during one τ cycle into a GLM readout trained to solve a specific task. (c) Networks of delay-coupled nodes can be understood as small recurrent systems inside a larger recurrent system. This model may be used to model the complexity of recurrence and delay coupling at the same time.

not only makes solving the mathematical problem more difficult, it also leads to dynamics that can be used for computation and inference.

The Delay-Coupled Reservoir

Introduced by Appeltant et al. (2011), the delay-coupled reservoir is a system described by a delay differential equation, such as Equation 11.1, but driven by an input $u(t)$:

$$dx(t) = -ax(t) + g(x(t-\tau), u(t)), \tag{11.3}$$

where g is a nonlinear function. The key insight from this work is that the activity in/of this simple one-node recurrent system can be sampled N-times during one delay cycle of length τ, and this sampled activity can be treated as the N-dimensional of a reservoir computer with N nodes. The input $u(t)$ is adapted to also change on the timescale of one τ-cycle, such that each τ-cycle associates one N-dimensional vector of activations with one input value. Following the reservoir computing procedure, this activation vector can then be used in a linear readout to learn a time-invariant, fading-memory function on the input (Figure 11.2).

The on-the-surface simplicity of delay differential equations leads to straightforward hardware implementations, where some nonlinear element is driven by input and self-coupled via a delay line. These simple building blocks have led to implementation based on standard electronic building blocks, but they also allow for the exploration of new computing devices, as in using delay-coupled lasers and photonics (Larger et al. 2012).

The hidden complexity of the system, however, allows it to be used in time-series forecasting, speech recognition, and even volatility prediction for financial markets (Appeltant et al. 2011; Grigoryeva et al. 2014).

This complexity, and the process of obtaining a vector of activity, can also be looked at theoretically using the method of steps. The iterative solution of the ordinary differential equation for subsequent τ-cycles, or intervals of length τ, can then be approximated analytically and written as a vector update equation for the N-relevant sample points directly (Schumacher et al. 2013). Thus, for computation within a reservoir computing setup, the infinite dimensionality of space of solutions to the delay differential equation reduces to N dimensions, a free parameter of the model. One can therefore profit from the potentially infinite dimensionality of a functional state. In practice, with a chosen decay rate α and a specific nonlinearity g, choosing arbitrarily large N does not benefit specific machine learning tasks above a task-dependent soft threshold. Nevertheless, researchers have seen benefits in expanding the dynamics of this simple, nonlinear, and delayed feedback-coupled node into an $N = 50$ up to an $N = 800$ dimensional state vector, as input to the linear regressor.

Instead of sampling the activity of the delay-coupled reservoir at *N* evenly spaced points, one can optimize the placement of these readout points. Here, it is useful to treat the *N* sampling points as a network of virtual nodes with a very particular connectivity structure: a lower diagonal exponential decay matrix. The distance from one sampling point, or virtual node, to the next can then be fine-tuned according to a homeostatic plasticity rule. It presupposes that good spatiotemporal computational performance is achieved when different virtual nodes are both sensitive to their inputs and as diverse as possible (Toutounji et al. 2015). The experiments in the study show that this rule does indeed lead to increased performance. From the point of view of the readout, the result permits a crude biological interpretation: a linear-nonlinear output neuron optimizes the locations along an axon, where it "reads" the activity of another neuron with complex time-dependent dynamics. Clearly, this interpretation is somewhat bold, but it highlights the potential of future research that uses delayed feedback models to encode and then decode information in temporal dynamics.

The delay-coupled reservoir can also serve as a model system to investigate how two different delays might interact. In a previous study, Nieters et al. (2017) highlighted strange dependencies that arise if Equation 11.3 is expanded to

$$dx(t) = -ax(t) + g\big(x(t-\tau_1), x(t-\tau_2), u(t)\big). \tag{11.4}$$

Delays that are close to simple rational, or even integer multiples of each other, lead to a poorly performing reservoir computer—how close is too close is controlled by the decay rate α of the exponential decay in the system. A too strong dependency of a sampling point onto its own history—the effect of choosing the $\tau_2 = 2\tau_1$—is detrimental. This delicate sensitivity to the choice of a second delayed feedback is reminiscent of the sensitivity to different delays in microcircuits mentioned earlier but is, of course, also an artifact of the discretized system used to model the activity at *N* sampling point with an analytic approximation and discretization. Future work must focus on a more realistic setting, where delays are distributed and continuous to investigate whether sharp transitions between well- and badly performing models also occur.

The takeaway from previous investigations into delay-coupled computation is that the added complexity can induce a complex temporal dynamics readout by an appropriate readout mechanism, which can benefit computation significantly. Reservoir computing is a compatible concept that embeds models, such as the delay-coupled reservoir, in the context of neural computation. More work is needed to connect the observed effects of delay coupling more closely with biological reality. Studies also highlight how complex neural networks may actually be that are subject to multiple delayed interaction effects. A possible perspective to study such systems abstractly is to connect single nodes with distributed delays recurrently in a reservoir, in the sense of a classical recurrent network. In such a network of networks, each node itself can be regarded as a simple recurrent system (Figure 11.3).

Discussion

In this chapter, we have discussed several computational principles at the level of individual neurons and networks of neurons, and addressed the implications of delayed communication. These principles ranged from specifically tuning single neurons to implement well-defined computational tasks (i.e., independent component analysis) to reservoir computing to implement computing based on randomly connected networks and random feature expansion. This diversity and wide range of functions can be viewed as either an overwhelming complexity that might just hide a key underlying unifying principle not yet uncovered, or a rich diversity used by the evolution as a large reservoir of tools and tricks to implement efficient computational circuits. If the latter is true, then the simple question of which computational principle do we discard is not sufficient. Instead, we need to address efficiency in terms of performance and the use of resources, robustness to noise and structural changes, and generalizability of the computational principles for different tasks. The ultimate question, however, remains essentially open: How does the cortex, or the brain, compute information?