

Toward Analog Neural Computation

Corey J. Maley¹

Received: 24 February 2017 / Accepted: 10 August 2017 / Published online: 19 August 2017
© Springer Science+Business Media B.V. 2017

Abstract Computationalism about the brain is the view that the brain literally performs computations. For the view to be interesting, we need an account of computation. The most well-developed account of computation is Turing Machine computation, the account provided by theoretical computer science which provides the basis for contemporary digital computers. Some have thought that, given the seemingly-close analogy between the all-or-nothing nature of neural spikes in brains and the binary nature of digital logic, neural computation could be a species of digital computation. A few recent authors have offered arguments against this idea; here, I review recent findings in neuroscience that further cement the implausibility of this view. However, I argue that we can retain the view that the brain is a computer if we expand what we mean by “computation” to include *analog* computation. I articulate an account of analog computation as the manipulation of analog representations based on previous work on the difference between analog and non-analog representations, extending a view originally articulated in Shagrir (Stud Hist Philos Sci 41(3):271–279, 2010). Given that analog computation constitutes a significant chapter in the history of computation, this revision of computationalism to include analog computation is not an ad hoc addition. Brains may well be computers, but of the analog kind, rather than the digital kind.

Keywords Computation · Analog and digital · Neural signaling

✉ Corey J. Maley
cmaley@ku.edu

¹ University of Kansas, 3090 Wescoe Hall, 1445 Jayhawk Blvd., Lawrence, KS, USA

1 Introduction

Is the brain a computer? Perhaps: it depends, of course, on what it means to be a computer. Many parties to the discussion have agreed—sometimes implicitly, sometimes explicitly—that what it means to be a computer is to be a kind of digital computer, or something that implements a Turing Machine. Although the architecture of the Turing Machine may not be relevant to the architecture of the mind or brain,¹ the account of computation provided by Turing and adopted by theoretical computer scientists is thought by many to be the correct account of computation appropriate for neural systems. For example, we would not expect to find a tape or read head in the brain; but for that matter, we do not find a tape or read head in modern digital computers, either. However, in both cases, the theoretical account of the nature of the computation being performed is grounded in Turing computability. Furthermore, given the nature of neural signaling, it has seemed to many that there is a quite natural fit between digital computation and neural systems; so close, in fact, that it has seemed plausible to say that neural systems just *are* computational systems.

For better or worse, this view cannot be correct. Recent findings in neuroscience show that it is simply false that neural systems operate like digital systems in any theoretically interesting manner. To show that this is the case, I will first outline the standard views about computation and neural signaling. Next, I will discuss some of the recent findings in neuroscience that demonstrate how different digital signals and neural signals really are, and argue that characterizing the brain as a digital computer is no longer a tenable position. However, the computationalist need not despair! I argue that we should expand computationalism to include *analog* computation, which then allows us to view neural signaling as computation once again.

2 The Standard Views

2.1 Standard Computationalism

Computationalism is the view that the brain is some kind of computer.² Few researchers in the cognitive sciences would deny that there is a close connection, or perhaps an analogy, between neural systems and computational systems. However, computationalism makes the stronger claim that the brain literally *is* a computer. To provide substance to that view, one needs to be clear about what it is for something to be a computer.

One might be tempted to look to computer science for the answer to what counts as a computer. After all, theoretical computer science tells us much about the

¹ That is, insofar as a Turing Machine can be said to have an architecture, given that it is an abstract mathematical object.

² More precisely, computationalism can be a view about the brain *or* about the mind. In this essay, I focus only on computationalism about the brain.

capabilities and limitations of various abstract models of computation. Unfortunately, computer science does not provide an answer to the question of when a concrete object is a realization of an abstract model of computation. Fortunately, philosophers have taken up the task, and some answers are available.

Different solutions to this implementation problem are available, and the development of the literature on this problem has brought up a number of important related issues. For example, is it necessary that an implementation of a computational system manipulates representations? According to some (and for a variety of reasons), the answer is yes (Fodor 1975; Egan 1995). According to others, the answer is no (Piccinini 2008; Egan 2010). Another issue is whether there are principled constraints on which automata a physical system implements. According to some, the answer is yes (Chalmers 1996; Piccinini 2007); according to others, no (Putnam 1988; Searle 1980).

Many more issues comprise the landscape of philosophical literature on computation and computationalism, but for the present essay we need not survey that landscape. What unites virtually all of the discussion within this landscape is the assumption that computation is discrete, or as some would say, digital. This is not an unfounded assumption: on the side of computer science, the canonical models of computation are all discrete automata, with the Turing Machine being the most well-known. The atomic units in discrete automata are discrete symbols, and automata manipulate patterns (or strings) of these discrete symbols. Virtually all modern computational systems are based on systems of digital logic, which use binary representations of numbers (i.e. numbers composed of “bits”) to perform all of their operations. Computation, on the received view, is discrete through and through.

2.2 Standard Neural Firing

The assumption of discreteness is also supported on the neuroscience side. It has long been established that the action potential (or neural spike, or neural firing) is the basic building block of neural signaling. Close study of the action potential over the last several decades has shown that it is taken to be an all-or-nothing phenomenon: once a neuron has reached a particular voltage threshold, an action potential is generated, and the size and shape of the action potential is constant. In that regard, the action potential is like a standard smoke detector. Smoke detectors sound an alarm once a certain threshold of smoke has been detected. But the alarm in a smoke detector is not louder or higher-pitched if there is more smoke: like an action potential, it is an all-or-nothing phenomenon. This fact about the nature of the action potential has led many to compare it to a binary signal, essentially identical to the kind of binary signal that serves as the basic building block (the single bit) of digital computation.

The neural spike is not all there is to neural signaling, however; a bit of background is in order for later discussion. Much of the interesting activity in neural systems happens at the synapse, which is a very small physical gap between two neurons. If a single neuron gets enough “input” (meaning that its voltage reaches a particular threshold), it will fire (meaning that an action potential—a wave of voltage change—will propagate down its axon). When an action potential reaches the end of an axon, it releases neurotransmitters at synapses where the axon is very

close to another neuron. These neurotransmitters then have an effect on other neurons, basically serving as input to these next “downstream” neurons. Depending on a variety of factors, this input might raise (or lower) the voltage of the next neuron, making it more (or less) likely to fire. When discussing what happens at a synapse, the neuron that generates the action potential is known as the presynaptic neuron, and the neuron that is affected by the neurotransmitters released by the presynaptic neuron is known as the postsynaptic neuron.

Not all inter-neural communication is based on action potentials. Some rely on direct electrical signaling across synapses via what are called gap junctions. This signaling is not all-or-nothing, but continuous. Unlike neurons connected via chemical synapses of the kind mentioned above, where a signal can only go in one direction (meaning that there is a well-defined presynaptic and postsynaptic neuron), gap junctions are bidirectional. Furthermore, because the current flowing from one neuron to another in a gap junction is not gated or attenuated, the signal is not significantly transformed. In essence, the gap junction connects one neuron to another such that the electrical signal flows to the second neuron as if it were merely an extension of the first. Because of these factors, neuroscientists have thought that gap junctions do not play a major role in neural signaling (although some have argued that the functions of gap junctions have been underestimated (Hormuzdi et al. 2004; Söhl et al. 2005)).

2.3 Neural Systems as Digital Systems

One of the core assumptions underwriting the idea that the brain could be a digital computer of some kind is the similarity between the all-or-nothing character of action potentials and the binary nature of digital circuitry. If we were to look closely at the voltage levels involved when an element of a modern digital computer goes from a “0” to a “1” we would see that the voltage changes continuously. However, because of the way that the circuits function, at the level of digital logic we can treat the voltage as if there were only two states: a low state and a high state, or “0” and “1” (see top of Fig. 1). Similarly, we can see that the precise voltage levels change continuously when an action potential is generated. Again, because of the way that the mechanism functions, we can treat the action potential as a simple on-or-off pulse (see bottom of Fig. 1). Different sources in theoretical neuroscience make this point. For example:

Although action potentials can vary somewhat in duration, amplitude, and shape, they are typically treated as identical stereotyped events (Dayan and Abbott 2005, p. 8).

The action potential is all-or-none: Stimuli below the threshold do not produce a signal, but stimuli above the threshold all produce signals of the same amplitude. However much the stimuli vary in intensity or durations, the amplitude and duration of each action potential are pretty much the same (Kandel et al. 2012, p. 33).

Since isolated spikes of a given neuron look alike, the form of the action potential does not carry any information (Gerstner et al. 2014, p. 5).

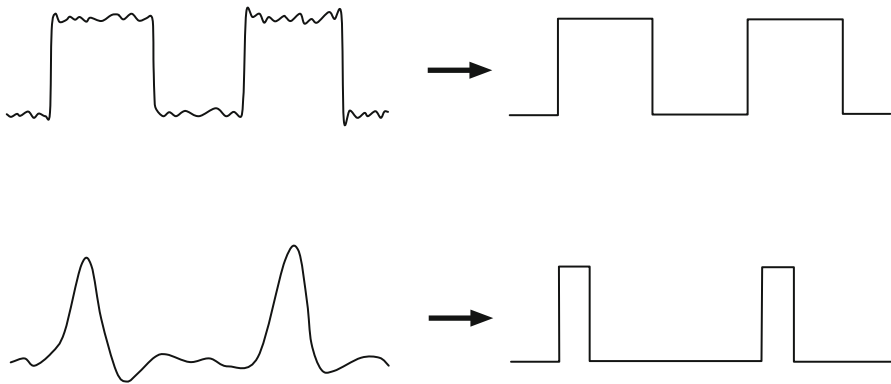


Fig. 1 *Top* actual voltage in binary computer element; treated as 0 or 1. *Bottom* actual voltage in action potential; treated as fire or not

We have, it seems, a match made in heaven: computational systems and neural systems both traffic in discrete patterns of discrete elementary units. Thus, it is natural to view neural systems as a kind of computational system. There are, of course, many philosophical issues that need to be addressed, some of which are mentioned above. But those interested in thinking of neural systems as computational systems have almost exclusively taken “computation” to refer to discrete, digital computation that has the Turing Machine as its theoretical basis. A very good articulation of this idea is seen here, at the end of a chapter explicitly discussing Turing Machines:

We have reviewed and explicated key concepts underlying our current understanding of machines that compute—in the belief that the brain is one such machine.... If one believes that the brain is an organ of computation—and we take that to be the core belief of cognitive scientists—then to understand the brain one must understand computation and how it may be physically implemented (Gallistel and King 2009, pp. 124–125).

2.4 Recent Alternatives

Before continuing, some important exceptions to the picture portrayed above must be noted. Both Miłkowski (2013) and Piccinini and Bahar (2013) have argued that computation, at least as it applies to, and is used in, cognitive science, is better understood in terms of a particular kind of mechanism (in the sense of Machamer et al. 2000) that has the function of computing. While I am sympathetic to the spirit of these authors’ approaches, both have drawbacks that I think warrant the alternative approach outlined below. A fully developed refutation of each will have to wait for another time, but a bit can be said here.

According to Piccinini and Bahar (2013), neural computation is a *sui generis* kind of computation. Neural computation cannot be digital, because digital computation requires the manipulation of strings of digits, and neural systems do not manipulate strings of digits. But neural computation cannot be analog, because

analog computation requires the manipulation of continuous variables, and neural systems do not do that, either. Instead, they do something that has elements of both. What makes neural computation worthy of the label “computation” is that they compute in a very general sense, which is “the processing of vehicles (defined as entities or variables that can change state) in accordance with rules that are sensitive to certain vehicle properties and, specifically, to differences between different portions (i.e., spatiotemporal parts) of the vehicles. A rule in the present sense is just a map from inputs to outputs; it need not be represented within the computing system” (Piccinini and Bahar 2013, p. 458).

According to Miłkowski (2013), computation is information processing performed by a mechanism. Information processing, according to Miłkowski, need not involve representation; indeed, the notion he seems to endorse is quite broad. Discussing the implementation of a computational system, Miłkowski writes, “For my purposes, the only thing that is important here is that the philosophical account of implementation should be compatible with any notion of mathematical computation used in computer science, mathematics, or logic” (Miłkowski 2013, p. 28). So, like Piccinini and Bahar’s account, computation is construed very generally.

Given that both accounts construe computation very generally, it is unproblematic for each to count mental and neural processes as computational. However, it seems that the general construal of computation is somewhat *ad hoc*: the only thing captured by the broadened conception of computation is neural and/or mental computation. Again, I cannot fully address these concerns here, but I think it is *prima facie* better to broaden our conception of computation in a way that captures other canonical examples of computers.

In contrast, Shagrir (2010) argues that brains are best understood as what he calls “analog-model” computers, a view which takes analog computation as its inspiration. This view is closest to the one I defend here. According to Shagrir (2010, p. 272), “to view the brain as a computer is to take cellular activity that takes place in the brain as reflecting or simulating external mathematical relations between the entities that are being represented”. Like Shagrir, I think there is much to be said for the view that analog computation can inform the notion of computation appropriate to neural systems. This essay is an attempt to further develop and motivate an account of analog neural computation.

3 Continuous Neural Phenomena

There are problems with the standard view of neural firing given above. The first is that, even assuming that neural spikes are all-or-nothing, the frequency or precise timing between them—both continuous quantities—play an important role in neural signaling. These are not new findings, and have been discussed to some degree in the literature on neural computation. I will review some of the main points here. The second problem *is* relatively new: there is strong evidence to suggest that neural spikes are not always all-or-nothing phenomena, and that continuous properties of the individual neural spike matter. Taken together, these facts about neural firing

suggest that there is virtually nothing about neural signaling that is discrete in any interesting sense.

3.1 Continuity Between Action Potentials

In a typical digital computer, the time between individual occurrences of 0s and 1s in a circuit is discrete and fixed. These machines are designed in such a way that the precise time between 0s and 1s does not matter: the system clock determines the discrete time window in which a voltage level counts as either a 0 or 1. However, this is not how things work when it comes to action potentials (APs). Two separate, well-known phenomena illustrate the importance of continuity, rather than discreteness, in neural signaling. The first, called temporal coding, occurs when the time *between* individual occurrences of APs has some functional significance. The second, called rate coding, occurs when the overall number of APs within a given duration—the frequency of APs—has some functional significance.

Temporal coding occurs when the precise timing of an AP matters. This is not something used in the design of digital circuit elements. An analogy with a certain kitschy piece of technology helps here: consider “The Clapper”. This is a device that detects the sound of hand claps, which then turns an electrical circuit on and off. So imagine that it turns a light on (or off) whenever someone in the vicinity claps once. If it has very high temporal resolution, then there will be no (perceptible) delay between a clap and when the light is switched. If it has low temporal resolution (say it only samples whether a clap has occurred every five seconds), there may be a large delay between a clap and when the light is switched. Furthermore, in the case of low temporal resolution, if someone claps numerous times within the five-second window, these additional claps will have no effect. This kind of neural coding occurs in many sensory systems. For example, the precise timing difference between neural signals coming from different parts of the auditory system can be used to compute the location of a sound’s source (Gerstner et al. 1996).

Rate coding occurs when the rate of AP firing matters. This is also not something used in the design of digital circuit elements. Consider, once more, The Clapper: imagine a light that is only on as long as someone is clapping, where the faster the claps, the brighter the light.³ This kind of neural coding occurs in many sensorimotor systems. For example, the rate that a neuron fires can correspond with strain on a muscle, or the angle of a particular joint (Brody et al. 2003).

Temporal and rate coding exemplify the important role that continuous variation plays in neural function, even under the assumption that individual neural spikes are discrete events. By itself, the existence of this continuous variation presents a problem for the analogy between neural signaling and digital computation. However, recent research shows that there is an additional fact about neural signaling that presents another problem.

³ In fact, this is how older light dimmers actually work: they send a rapid series of pulses to the light bulb, with a shorter gap between pulses for brighter light, and a longer gap for dimmer light.

3.2 Continuity Within Action Potentials

The phenomena just mentioned illustrate that discrete neural signals give rise to continuous phenomena. But the continuity resides in the time between neural spikes, or the rate at which spikes are being generated, and not the individual spikes themselves. In other words, the continuity is not an inherent property of neural spikes, but a relational property of multiple spikes assumed to be discrete in the way illustrated by the quotations above. Recent research, however, shows that continuous variation in individual neural spikes can have functional significance in neural systems. This research undermines the long-held belief that the precise waveform of the neural spike does not matter, which in turn further undermines the analogy that neural spikes can be thought of as similar to the individual bits of digital circuitry.

Neurons in several different parts of the vertebrate brain have been shown to emit APs whose precise shape has functional consequences. For example, this phenomenon has been demonstrated in the neurons of the cortex (Kole et al. 2007), the cerebellum (Christie et al. 2010), and the hippocampus (Alle and Geiger 2006). Current research is just beginning to uncover the mechanisms responsible for the variation in neural spikes, as well as providing more precise descriptions of the effects of those continuous variations in neural spike waveform. It is worth examining some of these results here.

The first three examples come from Bialowas et al. (2015). The received view has been that the voltage of a pre-synaptic neuron does not have an effect on a post-synaptic neuron until the threshold potential is reached, at which point an AP is generated, which itself then has a discrete, all-or-nothing effect. However, this is not completely true. For some neurons, the pre-synaptic voltage level before threshold is reached makes a difference: the closer the neuron has been to threshold, the more the effect on the post-synaptic neuron. In other words, the larger the neural spike is relative to its past voltage (the “spikier” it is), the smaller the effect on the post-synaptic neuron.

Another effect is that an increase in the pre-synaptic voltage prior to an AP results in an increase in the probability that the AP will have an effect. Not every AP results in the release of neurotransmitter from the pre-synaptic neuron: occasionally (and seemingly randomly), an AP is generated, but neurotransmitter is not released, meaning that there is no effect on the post-synaptic neuron. This phenomenon is known as synaptic failure. Bialowas and colleagues have shown that a higher pre-synaptic voltage prior to AP generation results in a lower rate of presynaptic failure. Synaptic failure rates, in turn, have effects on how efficiently connections between neurons are altered (Sullivan et al. 2003).

This increase in pre-synaptic voltage also has an effect on the pulsed-pair ratio between neural spikes. A variety of synapses display effects in which the continued presence of APs results in a change in how subsequent APs effect the post-synaptic neuron. For example, in long-term potentiation, the result of a large number of neural spikes in a short time results in subsequent spikes having a larger effect than the earlier ones: loosely speaking, the synapse has “learned” to respond more strongly to spikes. In long-term depression, the opposite happens: a larger number

of spikes results in a decreased response to subsequent spikes. The pulsed-pair ratio is a measure of how strong the effect of an AP is on a post-synaptic neuron from one spike to the next; we can call these spikes S_1 and S_2 . Bialowas and colleagues have shown that when the pre-synaptic voltage is low relative to the generation of a pair of APs, the voltage of S_2 is larger, on average, than that of S_1 . But when the pre-synaptic voltage is high relative to the pair of APs, the opposite occurs: the voltage of S_2 is smaller, on average, than that of S_1 . Again, when the spikes are “spikier,” potentiation occurs; when the spikes are shallower, depression occurs.

There are several other effects, but I will mention only one more here. Recent research has demonstrated that the voltage level of the body of the pre-synaptic neuron can have an effect on the *width* of the waveform of the AP (i.e. its temporal duration), which then has an effect on the post-synaptic neuron. Basically, the wider the neural spike, the larger the effect on the post-synaptic neuron (Debanne et al. 2013; Rama et al. 2015). Once again, this is in stark contrast to the view of APs as all-or-nothing, discrete phenomena.

3.3 Abandoning Discreteness

We have now seen that there are several ways in which the continuity of a phenomenon related to neural spiking plays an important role in neural signaling. First, unlike digital computers, there is no discrete “clock” that determines when a physical change in a circuit element should count as a change in signal. Second, unlike digital computers, the rate at which physical changes in a circuit element occur—which itself is a continuous quantity—can have effects on downstream elements in the system. And third, unlike digital computers, the precise shape of the voltage waveform of a circuit element going from “off” to “on” can have differential effects on other elements in the system.

Taken together, these empirical discoveries about neural function make it virtually impossible to maintain any interesting relation between discrete, Turing-style computation (including digital circuitry of the kind we find in contemporary computers) and the neural circuitry we find in nervous systems. Neural spikes simply are not analogous to discrete symbols (or bits) in any interesting sense. One choice is to declare computationalism to have been falsified. This is the option that many proponents of dynamical systems approaches to understanding the brain have taken. Another choice is to revise computationalism to include these phenomena. In the next section, I offer a sketch of a new way of how we might go about doing the latter.

4 Revising Computationalism

Rather than abandoning computationalism about the brain, I propose that we expand what counts as computational to include so-called “analog” computation. In this way, a computational system could be one that is digital (as in classical computationalism), one that is analog, or one that is an analog/digital hybrid. As noted above, both Miłkowski (2013) and Piccinini and Bahar (2013) have also offered revised conceptions of computation. What I propose here is an alternative,

and one that is better motivated than either of these. Instead of creating a new, *sui generis* category of generic computation that incorporates neural and cognitive computation, I believe we can simply appeal to analog computation, properly construed. This expansion is not ad hoc: although it has fallen out of favor, analog computation was once the dominant computing paradigm until digital computation replaced it (one excellent history of analog computers is given in Mindell 2002). What follows is, in the interest of space, only a sketch, but a sketch which can be fruitfully developed in future work.

4.1 Analog Representation and Computation

First, let me make clear what I take “analog” to mean. I will follow the taxonomy I argue for in Maley (2011). Briefly, I take an analog representation to be one in which some quantity varies with the quantity being represented in a strictly monotonic manner, where this variation can be either discrete or continuous. For example, consider an analog thermometer: it represents the temperature by the height of a column of mercury. Another example is an hourglass: it represents the amount of time elapsed from a starting point by the amount of sand that has passed from one compartment to another. On my account, both are analog representations because a quantity (mercury, sand) represents another quantity (temperature, time) such that an increase (decrease) in the representational quantity represents an increase (decrease) in the quantity being represented. Furthermore, this is true *regardless* of whether the quantity doing the representing is discrete or continuous. One might think that mercury, a fluid, is continuous, while sand comes in discrete particles; yet again, mercury is also composed of discrete molecules; yet again, molecules are composed of particles that might be continuous fields; yet again.... This can all be set aside on my account, because, unlike the received view of the distinction between the analog and the digital, “analog” and “continuous” are not synonymous: analog representations can be continuous *or* discrete.

With this conception of analog representation, we can count all of the continuously-varying neural phenomena above as analog representations. Again, note that it is not that they are continuous that makes them analog on my account. Rather, it is that variation in the variable under consideration (firing rate, specific time, or waveform) makes a difference in what is being represented. In most contexts it is simplest to take these variables to be continuous, but we need not worry about whether they are “really” continuous or not.

Now, to be clear, I am simply assuming that the neural phenomena in question are representations. Of course, not all neural phenomena are themselves representations: some neurons randomly fire, and random firing does not represent anything at all. While I do not presume to have a theory of representation on offer, I take computation to rely essentially on representation, which itself is an assumption that virtually all authors agree upon (see Piccinini 2008 for an important exception). For those neural phenomena that are not representations to begin with, this discussion is irrelevant, because there can be (to paraphrase Jerry Fodor) no computation without representation.

Next, we must consider how to characterize analog *computation* in a principled manner. I propose we take the simplest available route: look at analog computers. At first glance, one might think that the important—even defining—feature of analog computers is their use of continuous representations. This is the usual reading of the distinction between the analog and the digital: “analog” just is synonymous with “continuous,” while “digital” is synonymous with “discrete.” While it is largely true that analog computers operate on continuous representations, and it is completely true that digital computers operate on discrete representations, continuity is *not* the essential feature of analog representation. Rather, the essential feature of analog computers is that they operate on analog representations, where I mean “analog” in the sense listed above. This view is, at times, articulated in literature on analog computation. For example:

Analog computers are sometimes described as devices which handle continuous variables.... This is only part of the concept, however. While true, it tends to obscure the perhaps *more important* idea that the word, analog, implies some kind of a representative correspondence between two otherwise dissimilar entities.... Although not absolutely essential, analog computation characteristically involves 1:1 relationships between the components of the real system and those of the analog. (Robertson 1964, p. 553, italics added).

I would, of course, disagree with Robertson’s contention that analog computers *must* handle continuous variables; more important is his point about the “more important idea.”

Another more recent example is found here (where, by “digital”, Ulmann means what I would call “discrete”):

First of all it should be noted that the common misconception that the difference between digital computers on one side and analog computers on the other is the fact that former use discrete values for computations while the latter work in the regime of continuous values is wrong! In fact there were and still are analog computers that are based on purely digital elements. (Ulmann 2013, p. 2)

Thus, analog computations rely on particular relationships between components of the system and what is being represented. More succinctly, analog computations operate on analog representations in the sense of Maley (2011).

A particular example from the literature on analog computation further illustrates this last point. In order to create or work with an analog computer, one has to be familiar with a potentially wide variety of analog components. One such component is a kind of differential mechanism for adding two quantities. A monograph on the history of analog computers illustrates two kinds of differential mechanisms (Bromley 1990, p. 162). One is made of a pulley suspended by two wires that can be independently manipulated. The other is made of a gear between two toothed racks, where the racks can be independently manipulated. In both devices, moving one of the two components a given distance causes the center component to move half that distance (see Fig. 2). We can label the distance from a fixed point to a marked point on the left component a , and the corresponding distance on the right

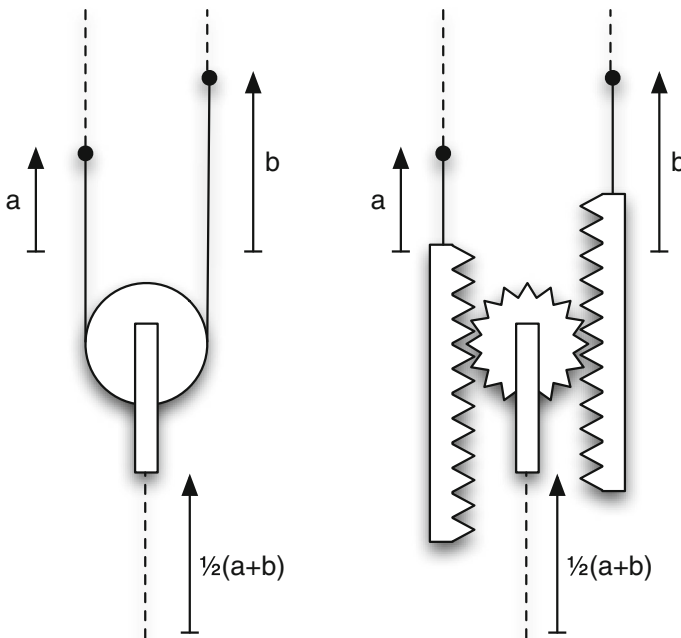


Fig. 2 *Left* differential adding mechanism using a pulley and wires. *Right* differential adding mechanism using a gear and toothed racks. Adapted from Bromley (1990, p. 162)

component b . Then, the distance between another fixed point and the center component is $\frac{1}{2}(a+b)$. Importantly, this type of mechanism will continue to work properly even if the geared version moves only in discrete “clicks.” What is important is not that they are continuous, or approximately continuous. Rather, what is important for them *qua* analog components is that they represent and manipulate variables in an analog manner (in my sense).

Generally, then, analog computers compute the solutions to various mathematical problems by using representations in a fundamentally different way than digital computers. In digital computation, variables are represented as we normally represent numbers in our everyday dealings with them: as a string of numerals. If we want to represent 345, we use the string “345”.⁴ In analog computation, variables are represented as quantities. If we want to represent 345, we use 345 V (or units of wire, or teeth on a gear, etc.). Analog computation can thus be defined as the manipulation of analog representations.

A full treatment of analog computation will have to wait for another time, but for the present essay it is enough to note that this kind of computation matches well with the analog neural phenomena mentioned earlier. In each of those cases, we have a one-to-one relationship between two variables in the manner which I claim counts as analog representation. Furthermore, the manipulation of those variables

⁴ Of course, virtually all modern digital computers use binary digits, so the string would actually be “101011001”.

corresponds with the way variables are manipulated in analog computation. Coupled with the account of analog representation I have previously defended, it does not matter whether neural signals or variables are discrete or continuous; as long as the right relationships exist, neural computation can be analog through and through, and discreteness versus continuity is an orthogonal consideration. If that is correct, then we can revise computationalism to include analog computation, and conclude that neural systems perform computations after all.

4.2 Objections

There are many possible objections to the view I propose; here I will address two. First, one might maintain that all of these findings about the brain are actually irrelevant to computationalism about the mind: after all, the “hardware” of the brain may well have a certain kind of architecture while the “software” of the mind has a completely different nature. A similar response was directed at connectionist research in past decades: whether or not the *implementation* of the mind is connectionist makes no difference to the nature of the mind itself. There is much to say on this matter, but I will limit my discussion to the claim that the *brain* is a computational device, a claim ubiquitous in current neuroscience, including cognitive neuroscience, computational neuroscience, and systems neuroscience. The entrenched computationalist who believes that claims about the computational nature of the brain are uninteresting or irrelevant to the doctrine of computationalism does so at her peril. Or so I claim, without argument.

Another worry might be that expanding the reference of “computation” to include analog computation lets in too much. Computationalism would certainly fail to be an interesting claim if *everything* is computational. However, that worry is unfounded. Just as not everything is an implementation of a digital computer, not everything is an implementation of an analog computer. As discussed above, analog computation is computation defined on analog representations; given that only certain things count as analog representations, only certain processes will count as analog computation. This, of course, depends on the view that computation is defined in terms of representation. Space prevents a complete defense of this view, which is not without its critics (again, see Piccinini 2008). However, the majority view is that computation *is* defined in terms of representation, and for now I will defer to that majority.

We have seen that all of the interesting analogies between digital circuits and neural circuits fail when scrutinized: neural signals are simply not like the bits of computers in any interesting way. However, rather than packing up our computationalism and going home, I have proposed here that we fix our computationalism to fit these new facts. We must be careful, though: like all good philosophical views, the theory needs to be able to capture the phenomena we want to include, and keep out the phenomena we want to exclude. The view I have sketched above fits this bill: standard digital systems of the kind with which we are all familiar get counted as computers, analog computers of previous decades get counted as computers, and so do neural systems of the kind that neuroscientists call computational.

5 Conclusion

Computationalism about the brain is the view that the brain is a computer. The original enthusiasm about the view came, in part, from discoveries about the nature of neural spikes and their similarity to bits in digital computation. Neuroscientific evidence has put strain on the degree to which one can consider neural systems to be analogous to digital computational systems, in turn putting strain on the truth of computationalism.

Here, I have shown how we might be able to avoid throwing out our computational baby with the digital bath water. Computation can be analog as well as digital, so computationalism about the brain should include the possibility that neural systems are analog computational systems. There are independent motivations for including analog computation under the umbrella of computation *simpliciter*; after all, analog computation was the name of the game before digital computation was established. However, close examination of recent developments in neuroscience give us excellent reason to revise computationalism in the way I have argued in this essay. Our brains may well be computers after all.

References

- Alle, H., & Geiger, J. R. P. (2006). Combined analog and action potential coding in hippocampal mossy fibers. *Science*, *311*, 1290–1293.
- Bialowas, A., Rama, S., Zbili, M., Marra, V., Fronzaroli Molinieres, L., Ankri, N., et al. (2015). Analog modulation of spike-evoked transmission in CA3 circuits is determined by axonal Kv1.1 channels in a time-dependent manner. *European Journal of Neuroscience*, *41*(3), 293–304.
- Brody, C., Romo, R., & Kepecs, A. (2003). Basic mechanisms for graded persistent activity: discrete attractors, continuous attractors, and dynamic representations. *Current Opinion in Neurobiology*, *13*, 204–211.
- Bromley, A. G. (1990). Analog computing devices. In W. Aspray (Ed.), *Computing before computers*. Ames, IA: Iowa State University Press.
- Chalmers, D. J. (1996). Does a rock implement every finite-state automaton? *Synthese*, *108*(3), 309–333.
- Christie, J. M., Chiu, D. N., & Jahr, C. E. (2010). Ca²⁺-dependent enhancement of release by subthreshold somatic depolarization. *Nature Neuroscience*, *14*(1), 62–68.
- Dayan, P., & Abbott, L. F. (2005). *Theoretical neuroscience*. Cambridge, MA: MIT Press.
- Debanne, D., Bialowas, A., & Rama, S. (2013). What are the mechanisms for analogue and digital signalling in the brain? *Nature Reviews Neuroscience*, *14*(1), 63–69.
- Egan, F. (1995). Computation and content. *Philosophical Review*, *104*(2), 181–203.
- Egan, F. (2010). Computational models: A modest role for content. *Studies in History and Philosophy of Science Part A*, *41*(3), 253–259.
- Fodor, J. A. (1975). *The language of thought*. Cambridge, MA: Harvard University Press.
- Gallistel, C. R., & King, A. P. (2009). *Memory and the computational brain*. Malden, MA: Wiley-Blackwell.
- Gerstner, W., Kempter, R., van Hemmen, J. L., & Wagner, H. (1996). A neuronal learning rule for sub-millisecond temporal coding. *Nature*, *383*(6595), 76–81.
- Gerstner, W., Kistler, W. M., Naud, R., & Paninski, L. (2014). *Neuronal dynamics*. Cambridge: Cambridge University Press.
- Hormuzdi, S. G., Filippov, M. A., Mitropoulou, G., Monyer, H., & Bruzzone, R. (2004). Electrical synapses: A dynamic signaling system that shapes the activity of neuronal networks. *Biochimica et Biophysica Acta (BBA)-Biomembranes*, *1662*(1–2), 113–137.

- Kandel, E. R., Schwartz, J. H., Jessell, T. M., Siegelbaum, S. A., & Hudspeth, A. J. (2012). *Principles of neural science* (5th ed.). New York, NY: McGraw-Hill Education.
- Kole, M. H. P., Letzkus, J. J., & Stuart, G. J. (2007). Axon initial segment Kv1 channels control axonal action potential waveform and synaptic efficacy. *Neuron*, 55(4), 633–647.
- Machamer, P., Darden, L., & Craver, C. F. (2000). Thinking about mechanisms. *Philosophy of Science*, 67(1), 1–25.
- Maley, C. J. (2011). Analog and digital, continuous and discrete. *Philosophical Studies*, 155(1), 117–131.
- Milkowski, M. (2013). *Explaining the computational mind*. Cambridge, MA: MIT Press.
- Mindell, D. A. (2002). *Between human and machine*. Baltimore, MD: Johns Hopkins University Press.
- Piccinini, G. (2007). Computational modelling vs. computational explanation: Is everything a Turing machine, and does it matter to the philosophy of mind? *Australasian Journal of Philosophy*, 85(1), 93–115.
- Piccinini, G. (2008). Computation without representation. *Philosophical Studies*, 137(2), 205–241.
- Piccinini, G., & Bahar, S. (2013). Neural computation and the computational theory of cognition. *Cognitive Science*, 34, 453–488.
- Putnam, H. (1988). *Representation and reality*. Cambridge, MA: MIT Press.
- Rama, S., Zbili, M., & Debanne, D. (2015). Modulation of spike-evoked synaptic transmission: The role of presynaptic calcium and potassium channels. *Biochimica et Biophysica Acta (BBA)-Molecular Cell Research*, 1853(9), 1933–1939.
- Robertson, J. S. (1964). Analog computation: Definition and characteristics. *Annals of the New York Academy of Sciences*, 115(1), 553–557.
- Searle, J. R. (1980). Minds, brains, and programs. *Behavioral and Brain Sciences*, 3(3), 417–424.
- Shagrir, O. (2010). Brains as analog-model computers. *Studies in History and Philosophy of Science*, 41(3), 271–279.
- Söhl, G., Maxeiner, S., & Willecke, K. (2005). Expression and functions of neuronal gap junctions. *Nature Reviews Neuroscience*, 6(3), 191–200.
- Sullivan, D. W., & Levy, W. B. (2003). Quantal synaptic failures improve performance in a sequence learning model of hippocampal CA3. *Neurocomputing*, 52–54, 397–401.
- Ullmann, B. (2013). *Analog computing*. Berlin: De Gruyter.