PHILOSOPHICAL TRANSACTIONS B

rstb.royalsocietypublishing.org

Opinion piece

Cite this article: Gallistel CR. 2017 Finding numbers in the brain. *Phil. Trans. R. Soc. B* **373**: 20170119.
http://dx.doi.org/10.1098/rstb.2017.0119

Accepted: 21 June 2017

One contribution of 19 to a discussion meeting issue 'The origins of numerical abilities'.

Subject Areas:
behaviour, biochemistry, cognition, computational biology, molecular biology, neuroscience

Keywords:
fixed-point two's complement, polynucleotides, number codes

Author for correspondence:
C. R. Gallistel
e-mail: galliste@ruccs.rutgers.edu

Finding numbers in the brain

C. R. Gallistel

Rutgers Center for Cognitive Science, 152 Frelinghuysen Road, Piscataway, NJ 08854-8020, USA

CRG, 0000-0002-4860-5637

After listing functional constraints on what numbers in the brain must do, I sketch the two's complement fixed-point representation of numbers because it has stood the test of time and because it illustrates the non-obvious ways in which an effective coding scheme may operate. I briefly consider its neurobiological implementation. It is easier to imagine its implementation at the cell-intrinsic molecular level, with thermodynamically stable, volumetrically minimal polynucleotides encoding the remembered numbers, than at the circuit level, with plastic synapses encoding them.

This article is part of a discussion meeting issue 'The origins of numerical abilities'.

1. Introduction

Behavioural and electrophysiological data now convincingly establish the existence of numbers in the brain—in animals from insects to humans. Other papers in this issue cover much of the evidence for this once controversial idea. Gelman & Gallistel [1] coined the term 'numeron' for a number in the brain. In this paper I pose the question, What are we looking for when we look for numerons? Are there constraints that hypothesized physical realizations of numerons must satisfy in order to be regarded as such? How will we know when we have found them?

Much of the work on numbers in the brain focuses only on the referential role of numerons—more particularly, on their reference to numerosities. Measurement processes establish numerical reference by mapping from magnitudes in the world (numerosities, lengths, durations and so on) to the numbers with which we represent those magnitudes. Counting is the process by which numerosities (discrete magnitudes) are measured. While counting is in principle precise, in practice, there are errors [2–4]. More generally, there is always uncertainty in measurements. Gelman & Gallistel [1] called the measurement processes that map from quantities to numerons 'estimators', stressing thereby that there was always some uncertainty attendant on referential mappings. As we will see, measurement error gives rise to a constraint on numerons: they must convey to future computational operations not only an estimate of a previously experienced magnitude but also the uncertainty about the true value of that magnitude. The uncertainty arose from the process of perceiving that magnitude in the first place.

Reference is but one aspect of the brain's representation of quantities. The other aspect is the computational operations the brain performs on those representations [5,6]. Gelman and Gallistel called these 'operators'. Operators are constructed from the basic operations of arithmetic—ordering, adding and multiplying. Numerons—neurobiologically realized symbols in memory—have a dual role: they represent (stand for) previously experienced quantities and they enter into computations. We must keep both roles in mind when we look for them.

When looking for a brain mechanism, we entertain various candidate mechanisms. These are hypotheses of the form 'x is the thing for which we are looking'. To decide among the candidates, we must form as clear a picture as we can of the properties that must be exhibited by a neurobiological entity for it to be the basis for a plausible hypothesis. I elaborate on the referential and operational constraints on numerons first. I then consider what these constraints suggest about the neurobiological realization of numerons.

My focus is on numerons as entities in long-term memory, that is, on symbols that carry quantitative experiential facts forward in time for use in computations

THE ROYAL SOCIETY PUBLISHING

that may be performed in the indefinite future [6]. My focus on the symbols in memory stands in contrast to the focus of contemporary neurobiological work, which is understandably on the signalling of currently experienced quantities by the firing of neurons [7–12]. I say 'understandably', because we know the physical realization of a neurobiological signal—the action potential—but we do not know the physical realization of an engram [6,13], a symbol in memory that carries information forward in time. As always, we look where the neurobiological concepts and tools currently available to us enable us to look. We look where the brain's representations of quantities becomes manifest in its signalling. My focus is on the engrams that underlie much of that signalling.

2. Constraints on numerons

(a) Referential

Basic computational operations take two numbers as input and produce a third as output. In many behaviourally relevant computations, one input refers to a numerosity while the other, or their product, or both refer to a continuous magnitude. Thus, numerons must be able to represent both discrete and continuous magnitudes. A rate is a continuous magnitude, but it is estimated by dividing numerosity (discrete magnitude) by duration (continuous magnitude). A probability is a continuous magnitude that is the proportion between two discrete magnitudes (two numerosities). (See also [14]; [10,15–25]). Thus, symbols that refer only to numerosities are not what we should be looking for. What we are looking for must be like the numbers we work with daily, which refer equally well to discrete and to continuous magnitudes.

Numerons must also refer to quantities that range over many orders of magnitude. Bees navigate over distances that measure in centimetres inside the hive and in kilometres outside it [26]. Artic terns navigate from the Arctic to the Antarctic and back annually, covering up to 80 000 km on a round trip [27].

Probabilities are always ≤ 1; thus, there must be numerons between the additive identity (the numeron for 0) and the multiplicative identity (the numeron for 1).

The elements of distance vectors, like the elements of direction vectors, must be signed—in order to distinguish between oppositely directed distances.

In sum, the numerons in the brain's number system must refer equally readily to numerosities (discrete magnitudes) and to signed continuous magnitudes, and they must refer equally readily to magnitudes much less than 1 and to magnitudes many powers of 10 greater than 1.

(b) Operational

On the operational side, an arithmetic operation performed on two numerons by the brain's implementation of arithmetic should always yield another numeron, that is another symbol for a magnitude. In formal developments of number (number theory), this property is called closure. The numbers in number theory form a system that is said to be 'closed under the basic operations of arithmetic'. The need to develop such a system drove much of the historical evolution of what are today are regarded as 'the numbers' by those who work with numbers professionally (mathematicians, engineers and computer scientists). Failure to satisfy closure constraints would cause the brain's computations to break down whenever its computational machinery was fed two numerons for which that machinery could not generate an output numeron. In the design of computing machines and computing software, much thought and effort goes into obtaining as much closure as possible, even though it is well understood that complete closure is impossible. The machine will sometimes be asked to compute 0/0, and there is no generally satisfactory answer as to what number that operation should generate. Thus, the computer engineering community has agreed that whenever a 'compliant' (well behaved) computing machine is asked to divide 0 by 0, it will generate the symbol NaN, which stands for 'Not a Number'.

I work daily with numbers on computing machines, and I can testify that NaNs are a nuisance. They are a nuisance because computations are the composition of functions. That is, the results of one computation are routinely fed to further computations. A computer program—which is a recipe for performing some complex computation—is just a huge composition of functions. It specifies a chain of operations in which the results of each operation are fed to the next operation. Unless special care is taken, when one of the operations in the chain generates a NaN, so do all subsequent operations. Suppose, for example, that the code calls for a mean to be computed, but one of the symbols fed in is a NaN rather than the symbol for a number. Who's to say what the mean of 10, 22.5 and NaN could possibly be? The mean is an operation that only applies to numbers. Any machine that works with numbers—and the brain is such a machine—cannot avoid dealing with this problem. That is why closure is a major consideration when we look for numerons.

(c) Joint constraints

Other constraints are jointly determined by referential and operational considerations. For example, as noted earlier, the extent to which a numeron can be relied on depends on the precision of the measurement process or mechanism that maps from the quantity measured to the numeron that stands for that quantity in the brain's computations. The precision of the numerons that a chain of computational operations starts with puts an upper limit on the precision of the result. Thus, a numeron must not only stand for a quantity, it must also somehow indicate how precisely that quantity is assumed to be known. It should do this in a way that propagates readily through a sequence of computational operations so that the computed result also conveys both a magnitude and an indication of how precisely it is known. This is a subtle but strong constraint.

Numerons should not carry noise into computations. Gallistel & Gelman [28], following Gibbon [29], suggested that Weber's law should be explained on the assumption that numerons had scalar noise. On further reflection, this suggestion was ill considered: in a chain of computations, it would lead to an ever decreasing signal to noise ratio. Dead reckoning, for example, requires the repeated adding of small displacements to the sum of previous displacements. If the numeron representing the sum has scalar noise, then the ever-increasing noise in the sum will cause it to drift randomly far from its true value (figure 1). Thus, Weber's law needs a different explanation.

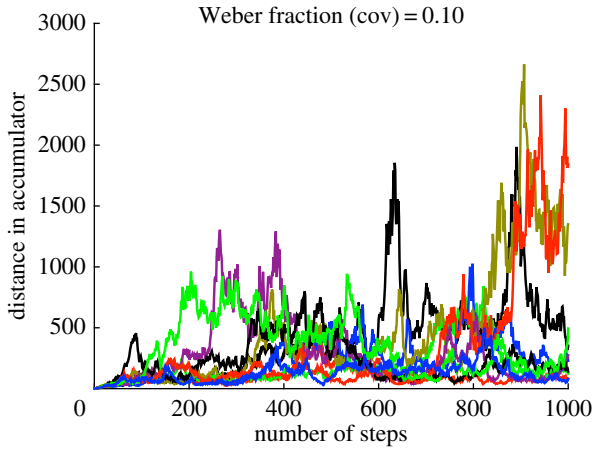

Figure 1. Repeated runs of simulated dead reckoning with noise in the accumulator of ±10%. At each step, the reading from the accumulator is drawn from a normal distribution in which the mean is the current sum, S, and the standard deviation is 0.1 × S. That value is added to the small displacement from the current step, and the result is returned to the accumulator. Random drift soon renders the sum in the accumulator useless. (Online version in colour.)

3. Shannon's principle

In thinking about numerons, it is fundamentally important to distinguish between a symbol as a physical entity that conveys information, on the one hand, and, on the other hand, as something that stands for something else and must therefore be interpreted and dealt with in a manner congruent with what it stands for. We have already considered an example of the latter—the role of reference in constraining interpretation: asking a computer to compute the mean of '10', '22.5' and 'NaN' is a bad move because the first two symbols refer to numbers, while the third is explicitly defined not to refer to a number. In thinking about this example, it is of further importance to bear in mind that in the machine itself 'NaN' is a bit pattern. Qua bit pattern, it is completely indistinguishable from a number. If you encrypt your data, then the 'NaNs'—which are also used to stand for missing data—will be treated as numbers by the number-theory-based algorithms that encrypt data of all kinds. Thus, the physically realized NaN symbol can be and sometimes is treated as if it referred to a number, even though functionally it is defined not to be a number.

Neither the semantics nor the syntax applicable to a symbol is relevant when our only concern is the conveyance of information. Shannon articulated this deeply important point in an opening paragraph of the paper that created information theory [30, p. 379]: 'Frequently the messages [to be conveyed] have meaning; that is they refer to or are correlated according to some system with certain physical or conceptual entities. These *semantic aspects of communication are irrelevant to the engineering problem*. The significant aspect is that the actual message is one selected from a set of possible messages. *The system must be designed to operate for each possible selection*, not just the one which will actually be chosen since this is unknown at the time of design'. [italics mine]

Memory is commonly discussed in domain-specific terms—'working memory', 'long-term memory', 'spatial memory', 'emotional memory', 'declarative memory', 'procedural memory', 'motor memory' and so on. These discussions fail to distinguish, on the one hand, between what an entity in memory refers to and how it enters into the control of behaviour and, on the other hand, the underlying mechanism that conveys information forward in time so that it may be read and processed at some future time [6]. Shannon's principle is that the first of these considerations—what the information is about (semantics) and how it may be processed (syntax)—is irrelevant to the second—how to convey the information, that is, to transmit it and store it.

There is little reason to think that Shannon's principle fails to apply to the brain. DNA, for example, conveys every kind of information in every kind of organism. Because it is a highly efficient, extremely stable, general purpose memory medium, it is being actively investigated for use as the memory medium in a conventional computer [31]. Like a thumb drive, DNA is oblivious to the semantics and the syntax of the information it carries. Thus, Shannon's insight applies to the only currently known biological medium for the conveyance of information.

4. Numeron desiderata

The constraints just sketched suggest properties we should look for when looking for numerons. They should

— be *generable*, that is, producible as needed. In the above quote, Shannon calls attention to the practically infinite size of the set of messages that might have to be conveyed—to the infinity of the possible [6]. Therefore, the brain must be able to generate new numerons as they are needed
— have *indefinite range*: approximate any possible magnitude
— be *noiseless* in computations (figure 1)
— *be signed*
— *include the identities* (0 and 1)
— *represent precision as well as magnitude*
— be *accessible to computation*
— be *closed under computation*
— be *writable*: there must be a mechanism that translates quantities conveyed by incoming spike trains into enduring change in the engram, the memory medium
— be *readable*: there must be a mechanism that translates numerons back into the spike train code or codes for quantities
— be *durable*: they must last a long time with little or no energy input required
— be *compact*: they must occupy as little volume as possible

5. Where to look for inspiration

The neuroscientific literature contains little in the way of suggestions as to how brains might covey quantitative information into the indefinite future in computationally accessible form. When it comes to memory, neuroscientists have adopted the perspective of associationist philosophers and the behaviourist psychologists: they conceptualize memory as the process of forming associative bonds of varying strengths [32–34]. These are called connection weights, Hebbian synapses, plastic synapses or altered synaptic conductances. They are thought to determine how a neural network maps input vectors to output vectors, but they are not generally thought to encode quantitative facts about the experienced world [35].

Only in molecular biology and computer science do we find treatments of information storage. Molecular biology deals with the conveyance of heritable information from generation to generation for use in directing the building of

the organism and controlling much of its physiology and behaviour. Computer science deals with the conveyance of acquired information forward in time for use in future computations. In both cases, the memory code is foundational. Everything rests on it.

The correspondence between the principles governing memory in these otherwise disparate examples is remarkable [36]. Both use a digital code with a minimal number of code elements. In both, the memory registers have two portions, a coding portion and an address portion. In both, hierarchical data structures are created by indirect addressing, which naturally mediates the binding of variables to their values [6,37]. When computer engineers established the architecture of a random access memory, they reinvented the wheel. Evolution had hit on the principle a billion years earlier. This suggests that the architecture of an effective memory is strongly constrained. This in turn suggests that what computer engineers have found to be the most effective representation of numbers in the memory of a computing machine may have lessons to teach us when we look for numerons.

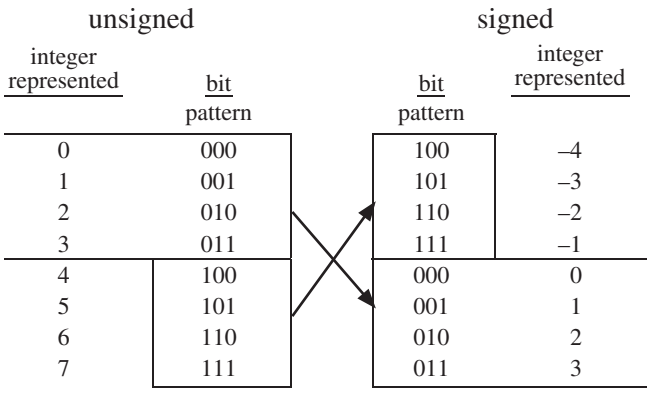

Figure 2. Three-bit significands partition the range of quantities specified by an exponent into 8 approximate values at even intervals that tile that range. The most obvious use of these 8 patterns is to encode the integers from 0 to 7 (left half of figure). However, we can use them to represent signed (directed) magnitudes by using the four-bit patterns in which the left-most bit is 1 to represent oppositely directed quantities, that is, negative numbers. This is the two's complement scheme for encoding signed numbers.

6. The two's complement fixed-point scheme

At the dawn of the computer age, von Neuman suggested using the two's complement fixed-point code for numbers. It remains the standard when speed, minimal energy use and maximally simple mechanisms for implementing the arithmetic operations are the desiderata [38]. Clearly, therefore, it is not easy to imagine a better code. Insofar as evolution tends to find good solutions to basic problems, it may be worthwhile for those interested in finding numerons to become familiar with this scheme.

It is a binary code, a code with only two elements, the additive identity (0) and the multiplicative identify (1). Each symbol has two basic parts, a significand and an exponent. The exponent specifies the binary order of magnitude, which is to say, the scale of the representation, the ballpark, so to speak. The significand specifies an interval within that range. The number of bits (binary digits) in the significand indicates how finely that range is subdivided.

The same two-part principle underlies scientific number notation, that is, notation of the form 1.3e3 = 1300. In scientific notation the power to which 10 is raised (3 in the example) specifies the scale at which we are representing something. The number of digits in the significand (digits to the left of e) specifies the scalar precision, which is to say, the number of distinguishable quantities at the given scale. Two digits in the significand implies a precision of roughly $\pm 1\%$. That is, there are 100 distinguishable quantities at whatever scale is specified by the exponent.

As remarked earlier, we need a better explanation of Weber's law (the scalar uncertainty in the brain's representation of quantities). Assuming scalar noise in the numerons themselves has intolerable computational consequences. Thus, we should take note of the fact that a scheme that divides a number into an exponent and a significand is a scheme that obeys Weber's law. The exponent establishes the range of magnitudes, while the number of digits in the significand indicates how finely quantities within that range are specified. The imprecision with which a quantity is specified is then proportionate to the scale. Thus, the fixed-point number code provides an example of physically realized symbols for numbers that specify an approximate quantity and also indicate the precision of the approximation. The representation of quantity by this code obeys Weber's law even though the symbols are noise free. It is their noise-freeness that makes digital symbols for numbers so much better than analog symbols for numbers. (An analog symbol represents one physical quantity, with a physical quantity of a different kind, e.g. a length represented by a voltage.) With analog symbols, one sees ever increasing noise as one traces the progress of a computation through multiple operations (figure 1). With digital symbols, this does not happen.

In adults, the Weber fractions for both numerosities and interval durations fall mostly in the range from 0.125 to 0.25 [39]. This implies that these quantities are represented with between three- and four-bit precision, that is, using 3 or 4 binary digits. In what follows, I assume 3 binary digits, because that restricts the number of bit patterns to be considered to the eight 3-bit patterns shown in figure 2.

The 8 patterns that may be formed from 3 binary digits are naturally thought of as encoding the integers from 0 to 7, as suggested on the left half of figure 2. However, an essential point—indeed, the single most important point in this paper—is that in the two's-complement fixed point scheme, those 8 patterns are used to approximate any 8 numbers whatsoever—positive, negative, integer or rational, and to represent with 3-bit precision quantities arbitrarily small and arbitrarily large.

The fixed point coding of number represents many numbers only approximately, that is, with limited precision. The same is true in any physically realizable number-coding scheme. Ever since the Greeks discovered irrational numbers, we have understood that no scheme for the physical encoding of numbers allows for the exact representation of every real number [40]. There is, for example, no scheme that can exactly represent both the length of the side of a unit square and the length of its diagonal.

(a) Sign

To incorporate sign, the two's complement code reinterprets half of the bit patterns; it uses them to represent negatively directed quantities (figure 2). When a distance travelled to

Table 1. Two's-complementation of the first 3 positive integers.

decimal	binary	flipped	flipped +1	decimal
1	001	110	111	−1
2	010	101	110	−2
3	011	100	101	−3

the north has positive sign, then a distance travelled to the south has negative sign. When a direction to egocentric right has positive sign, then the opposing direction has negative sign. When positive numbers represent what is owed to us, then negative numbers represent what we owe to others.

In the two's complement encoding, the left-most bit (aka the most significant bit) is 0 when the quantity referred to is given positive sign and 1 when it is given negative sign. Notice that this is true in the right half of figure 2: all the negative bit patterns begin with 1; all the positive ones begin with 0. For this reason, the leftmost bit in two's complementation is often called the sign bit.

There are other ways to introduce sign, but the two's-complement scheme has the advantage when it comes to ease of implementation. If we already have machinery that implements bit flipping—converting '0's to '1's and vice versa—and machinery that implements addition and multiplication with positive bit patterns, then we do not need to create further machinery to implement these operations when we introduce signed bit patterns. For example, to subtract one number from another, we take the two's complement of the subtrahend and then add it to the minuend. In other words, changing the interpretation of the bit patterns—their semantics, what they stand for—enables the use the same processing machinery to different effect.

The two's complement of a binary integer is produced by flipping the bits and then adding 1 (table 1). Flipping the bits in a string of bits is called taking the one's complement. Addition, the second step in taking the two's complement, is the second most basic operation. Implementing it is more complex than flipping a bit, but there is no way to avoid this complexity (table 2; figure 3).

Addition is implemented with logic gates. A logic gate takes two bits as input and generates a bit as output (figure 3). For addition to be implemented, the addend and the augend (the two patterns being added) must both have the same number of bits in their significands and the same exponent. That is why the code for +1 is 001 in the 3-bit scheme, even though it could be rendered as 01.

(b) Circular addition

The additions in table 2 discard those bits in the output that fall to the left of the bits that appear in the inputs. When the addition of n-bit numbers is implemented in this constrained way, it maps the set of n-bit numbers into the n-bit numbers. In doing so, it creates a system that is circular under addition. This will seem odd—nay disastrous—when we think of numbers as referring only to numerosities and we think of addition as modelling the effect on numerosity of uniting sets. It is just at that point in our thinking that we must remind ourselves that numbers may refer to quantities other than numerosities. They may refer to directions, for example, and to the magnitudes of turns, and to the phases of a cycle, such as the

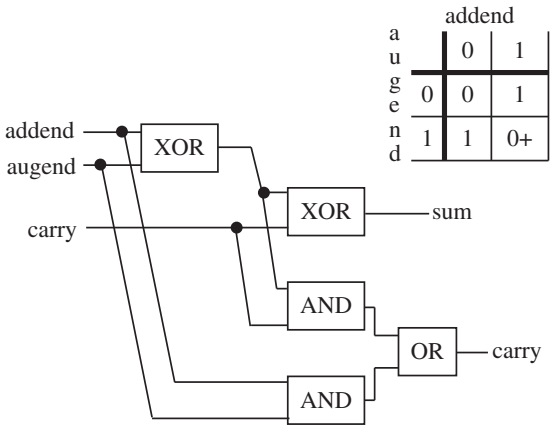

Figure 3. The addition table for bits and its implementation with logic gates. Note that the sum of 1 and 1 (lower right cell in table) is 0 + a carry bit. The carry bit must be added to the sum of the next most significant pair of bits. Thus, summing must proceed sequentially from right to left (from the least significant pair of bits to the most significant pair). If both the addend and the augend are 1 and there is no carry bit in the input, then when this pair are summed, the logic gates generate a '0' for the sum bit and a '1' for the carry bit, the bit that must be fed into the logic gates that operate on the next most significant pair of bits. The generation of the carry bit proceeds via the lowest branch on the logic gate diagram. The generation of the '0' for the sum bit proceeds via the uppermost branch. The topmost XOR gate produces a '1' only if either the addend or the augend is '1', but not both. Thus, when they both are '1' or they both are '0', its output is '0'. If the input carry bit is also '0', then so is the sum bit, which is generated by the second or lower XOR gate. If one of the addend and augend bits is '1' but not both, then the output of the top XOR gate is '1'. If the input carry bit is '0', then the second (lower) XOR gate generates a '0' for the sum bit. If, however, the input carry bit is '1', then it generates a '0' for the sum bit. Thus, this arrangement of logic gates generates the binary addition table shown at upper right. Conceptually, addition seems about as simple as it gets, but building a machine that implements it noiselessly is not so simple.

Table 2. Examples of circular two's complement addition. Note: The overflow bit (the carry bit generated by the leftmost addition) is dropped in the bottom two examples in this implementation of two's complement, fixed point addition. This implementation—this way of processing (interpreting) the input bit patterns—keeps only as many bits in the output as were in the inputs.

1 + 2 = 3				−3 + 2 = −1			
carry				carry			
augend	0	0	1	augend	1	0	1
addend	0	1	0	addend	0	1	0
sum	0	1	1	sum	1	1	1
−1 + −1 = −2				**−2 + −2 = −4**			
carry	1	1		carry	1		
augend	1	1	1	augend	1	1	0
addend	1	1	1	addend	1	1	0
sum	1	1	0	sum	1	0	0

day–night cycle that entrains the endogenous circadian clock. Animals learn directions [41,42]. They do angular dead reckoning, that is, they add up all the turns they have made to obtain the net change in their heading [5]. They remember

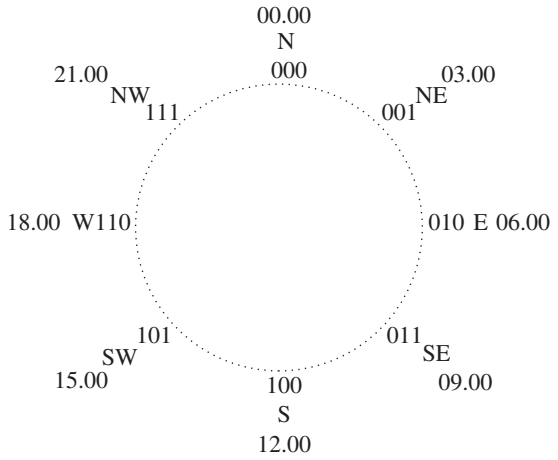

Figure 4. The 3-bit number circle may be used to represent the magnitudes of turns to an accuracy of 1/8 of a turn (see the bit patterns themselves: '000', '001', etc.), or the 8 principal points of the compass ('N', 'NE', etc.), or the phases of the circadian cycle (00.00, 03.00, etc.), because these variables have a circular structure under addition.

the circadian phases (times of day) at which events of interest happen [5]. They add up positive and negative temporal intervals—phase differences between events—in order to construct the temporal maps of past events that enable them to anticipate future events [43–46]. For all of these functions, circularity under addition is just what is needed (figure 4).

Consider first angular (that is, directional) dead reckoning. Two 135° turns to the right (two turns of binary magnitude 011) are equivalent to a 90° turn to the left (a turn of binary magnitude 110). Indeed, when we add 011 to 011, we get 110. Angular dead reckoning is the computation of the net effect on one's heading of a sequence of turns. The addition of the two's complement numbers with preservation of bit-length by left truncation correctly models the manner in which turns combine.

Consider next compass directions: in 3-bit two's complement binary code, the bit pattern 100 represents an about turn. An about turn from any point on the compass reverses the sign of one's heading. Adding the bit pattern '100' to the bit pattern that represents any point on the compass generates the bit pattern that lies directly across the circle from the initial bit pattern (figure 4).

Consider finally temporal intervals, that is, phase differences. The 6 h difference in the time of day from 21.00 to 03.00 is the same as the advance from 15.00 to 21.00, and, indeed, in binary 3-bit two's complement arithmetic (001−111) = (111−101) = 010. (Remember that to perform the subtraction, we take the two's complement of the subtrahend, then add.)

This representation of angles is the unique representation that has complete and unique closure: any addition of two angles coded in this way generates a bit pattern that is itself the representative of an angle and that is, moreover, the unique representative of the sum of those two angles. In no other physically realized representation of the angles is this true. Another constraint on numerons is that they facilitate the angular calculations that are central to temporal and spatial navigation. The two's complement fixed-point scheme does this uniquely well. This would seem to be another argument in favour of the hypothesis that the brain uses this code (the first argument being that this code obeys Weber's law).

(c) Linear addition

In the full implementation of two's complement fixed-point arithmetic, the overflows are not discarded; instead, all the outputs are augmented by one bit, as shown in tables 3 and 4. The full implementation maps the n-bit numbers into the $(n + 1)$-bit numbers. Under that mapping, the system is not circular; rather, it expands linearly in both directions. The outputs span double the range of the inputs (from −8 to 7 in the outputs versus from −4 to 3 in the inputs, in the case of 3-bit inputs). The computational machinery is the same in both restricted (circular) and the full (linear) implementation. The difference is in how the results are used: in the circular case, only the n least significant bits of the outputs are used; whereas in the linear case, all $n + 1$ bits are used—in order to accommodate the relatively few outputs that require an additional bit in their representation. Again, we see the importance of distinguishing the semantics from the syntax, but we also see that the appropriate syntax is determined by the semantics. How a bit pattern representing a magnitude should be processed depends on the kind of magnitude to which it refers. And, finally, we see that neither the syntax nor the semantics is relevant to the memory mechanism. The memory mechanism's only role is to preserve the bit patterns indefinitely. In doing so, it conveys the information from past experience into the indefinite future.

In tables 3 and 4, one may note the left padding of many of the positive numbers with additional and uninformative '0's and, similarly, the left padding of many of the negative numbers with additional and uninformative '1's. In the two's complement scheme, when a string begins with consecutive 0s, all but the rightmost are redundant. Similarly, when a string begins with consecutive '1's. Deleting all but the rightmost bit in these beginning strings does not alter the number. Table 5 shows the trimmed two's complement 5-bit binary numbers. Note that the entry for +16 is the only entry that requires 6 bits. As is apparent in table 5, the numbers of bits required to represent an integer grows with the size of the integer. However, it grows only logarithmically. Thus, very large numbers may be represented with bit strings of modest length. The circumference of the earth in millimetres is represented by a binary string only 26 characters long.

(d) Control and representation of precision

To represent the circumference of the earth to the millimetre is to adopt a foolish precision. For most purposes, the nearest 100 km (40,100) would do, or even the nearest 1,000 (40,000). In scientific notation, we would write 4.01e4 in the first case and 4.0e4 in the second case, where the e separates the significand from the exponent (the power of 10). Now consider the foraging ant *Cataglyphus* dead reckoning a run of 100.001 m by counting its millimetre steps [47] using a two's complement accumulator. At the end of its run, the count would be 100,001 decimal = 01,1000,0110,1010,0001 binary. This would represent the actual distance run only if every step were exactly 1 mm, there were no counting errors and the way were perfectly flat, none of which will be true under natural circumstances. Under natural circumstances, the ant odometer appears to be accurate to only about ±10%, which is similar to the human Weber fraction for a dead reckoned distance [48]. This implies an odometric precision of 3 or 4 bits. Thus, a much better representation of the distance run would be 0110e01110 or 01100e01101, depending on whether we

Table 3. The full 3-bit two's complement addition table. Note: The negative numbers are italicized; they are all and only those whose leftmost bit is '1'.

	000	001	010	011	*100*	*101*	*110*	*111*
000	0000	0001	0010	0011	*1100*	*1101*	*1110*	*1111*
001		0010	0011	0100	*1101*	*1110*	*1111*	0000
010			0100	0101	*1110*	*1111*	0000	0001
011				0110	*1111*	0000	0001	0010
100					*1000*	*1001*	*1010*	*1011*
101						*1010*	*1011*	*1100*
110							*1100*	*1101*
111								*1110*

Table 4. The decimal equivalent. As in table 3, negative values are italicized.

	0	1	2	3	*−4*	*−3*	*−2*	*−1*
0	0	1	2	3	*−4*	*−3*	*−2*	*−1*
1		2	3	4	*−3*	*−2*	*−1*	0
2			4	5	*−2*	*−1*	0	1
3				6	*−1*	0	1	2
−4					*−8*	*−7*	*−6*	*−5*
−3						*−6*	*−5*	*−4*
−2							*−4*	*−3*
−1								*−2*

assume 3- or 4-bit precision. The bits to the left of the 'e' are the significand; those to the right are the exponent. The significand is the first four or five bits of the 18-bit full count string, while the exponent is the binary representation of the number of bits dropped from the right-hand side of the string (14 or 13). This significand + exponent representation is better not only because it is more compact but also because it does not deceive as to the amount of information that the ant actually has about the distance it has run. By choosing the number of bits in the significand to be appropriate to the uncertainty inherent in the odometer, the system makes its representation of the distance run do double duty. It represents both the approximate distance run and the uncertainty about that distance (the precision of the approximation).

(e) Non-integer scaling

Having now brought the exponent into the picture, we begin to see how the fixed point scheme for representing quantity is able to represent non-integer values like probabilities. Consider the string 0110e101. The decimal equivalent of the significand ('0110') is 6. The decimal equivalent of the exponent ('101' = −3 decimal) is $2^{-3} = 1/8$. Therefore, the number represented is 6/8, a perfectly proper probability. In short, negative exponents yield non-integer values for fixed point numbers. We begin to see that, with a few bits and a well chosen coding scheme, one can represent any quantity a brain might need to represent. The coding scheme deploys in each case the same basic symbols—bit patterns—realized in whatever way bit patterns are realized on a given machine, and operated on with the same basic processing machinery.

Table 5. The trimmed 5-bit two's complement integers.

+ dec	+ two's com	− dec	− two's com
1	01	−1	1
2	010	−2	10
3	011	−3	101
4	0100	−4	100
5	0101	−5	1011
6	0110	−6	1010
7	0111	−7	1001
8	01000	−8	1000
9	01001	−9	10111
10	01010	−10	10110
11	01011	−11	10101
12	01100	−12	10100
13	01101	−13	10011
14	01110	−14	10010
15	01111	−15	10001
16	010000	−16	10000

(f) Offset (putting bits where they are most needed)

How can this scheme efficiently represent a small range of quantities a long way from 0? To do that, the system uses an offset, sometimes called a bias. Suppose one wants to represent the integers from 61 to 68. One could do that with the following 8 strings: 0111101, 0111110, 0111111, 01000000, 01000001, 01000010, 01000011 and 01000100.

Using a bias of 01000001 (=65 decimal) makes for a much more efficient representation. The bias, being common to every value, is stored only once, followed by the values to which it is to be added, which are the same 3-bit two's complement values that we first encountered: Thus, the above string can be reduced to 01000001 + 100, 101, 10, 1, 0, 01, 010, 011.

Using an additive bias, the bit patterns in figure 2 may be made to refer to the integers 61 : 68 rather than to the integers (−4 : 3). The more instances of integers within this narrow range we have to store, the greater the savings realized by the use of a common bias. The bias puts the stored bits into the range where they do the most work, thereby increasing the amount of information stored per element. The bias is a

separate 2-part symbol generally stored only once for a whole string of related quantity representations. Thus, it is not a basic part of the representation of a quantity. The use of bias is another example of a clever use of the same basic resources so as to minimize the resources required to represent a given amount of information. This is appears to be a general principle in the design of neural machinery [49–51].

7. Neurobiological hardware

In the abstract, the two's complement fixed point encoding scheme satisfies our list of desiderata, but is it neurobiologically plausible? The answer depends on what level of neurobiological structure one considers. If one thinks the brain's basic computations are carried out by structures realized at the level of neural circuits (the systems level), in which the information being manipulated is somehow stored in plastic synapses, then I am unable to imagine how this scheme could be implemented. The first question I would need to discern a possible answer to is: how would one store numbers in plastic synapses in computationally accessible form? Here already, my mental powers fail me [36].

The metabolic costs of carrying out computations using rate codes passed around in circuits of thousands or even millions of neurons are, however, enormous, because each spike requires the hydrolization of 10^8 ATPs [52,53]. The costs of carrying out the same computations using intracellular neurochemistry are many orders of magnitude lower. Metabolic cost has been suggested as a unifying principle in neuronal biophysics [54]. These metabolic cost considerations have recently led two influential neuroscientists to promulgate the principle: '. . .compute with chemistry. It is cheaper'. [55, p. 124].

It has recently been discovered that the CS–US interval in eyeblink conditioning is encoded by a cell-intrinsic mechanism in the cerebellar Purkinje cell [56]. In the vast literature on the neurobiology of memory, this is the only case in which the engram for a specified quantitative experiential fact (the duration of the CS–US interval) has been localized—and it has been localized to a sub-cellular level. In this case at least, the numeron encoding the duration of an interval resides in a cell-intrinsic medium, not in a plastic synapse.

These considerations—the metabolic and volumetric costs and the fact that the CS–US interval is stored by a cell-intrinsic mechanism—lead me to seek answers at the cell-intrinsic level, which is to say at the molecular level. At that level, it is easier to find structures of the kind one would look for to implement an efficient, versatile number-encoding scheme.

As mentioned in §1, it is easy to see how to encode numbers into polynucleotide strings (DNA and RNA). These string-like molecules are found in abundance inside every cell. Machinery for constructing these strings is also present in every cell. Thus, these strings are constructible as needed. Moreover, the complementizing properties of these strings—the properties that enable genes to make copies of themselves—are ready-made for the implementation of two of the most basic computational operations: copying and bit-flipping.

To suggest that numerons are polynucleotides is a speculation so extreme as to place it beyond the pale of ordinary computational neuroscience discourse. I nonetheless pursue the suggestion a little way here because it illustrates so strikingly what happens when one shifts the level of structure entertained in one's hypotheses from the circuit level to the molecular level. Systems level computational models are themselves highly speculative and they often make assumptions like back propagation and context units for which there is no neurobiological evidence, so it is arguably time that we consider whether the systems level is really the right level for an understanding of the basic mechanisms of computation in neural tissue.

(a) Encoding numbers into polynucleotides

There are many ways to do it. Here's one: the base pairs, **A**denine, **C**ytosine, **T**hymine and **G**uanine, which are the elements (nucleotides) out of which polynucleotide strings are built, form complementary pairs, G–C and A–T (the Watson–Crick DNA base pairs; uracil replaces thymine in RNA). Assume that one of the base pairs is used to encode bits, while the other is used to encode string function. Let G encode 0 and C encode 1. Let a string consisting only of G and C be functionally identified as the significand in a three-part representation (significand, exponent, bias). Let a string that starts with A be functionally identified as the exponent and a string that starts with T be functionally identified as the bias.

By Watson–Crick base pairing, a '011' string, embodied by GCC, will complementize with a CGG string. If they then separate, the CGG string will complementize with a GCC string, thereby creating a copy of the original encoding string. Moreover, the GCC string is the bit flip of the CGG string. Therefore, if we have machinery that can perform addition on these strings, and we use it to add 1 to the bit-flipped string, we will get the two's complement of the CGG string, which is the negative of the number encoded by the CGG string. I consider elements of the addition machinery after briefly explaining why copying is itself a fundamental computational operation.

(b) Why copying is fundamental

Consider the ant or bee that arrives at a food source after a wandering foraging expedition. It does two things: (i) it records the location of the source in memory; (ii) it inverts its dead-reckoning vector, which specifies its current location relative to the nest, to produce the home vector, which specifies the location of its nest or hive relative to its current location. The first operation is a copying operation: it copies the symbols in the accumulator that give the coordinates of its current location into memory registers that record that location for future use. Putting this copy in memory enables the ant to return directly to the food source on its next outing. The second operation—inverting the current-location vector—enables it to take the straight course home (dashed line in figure 5), even though it arrived at the location of the newly discovered food source by a tortuous food-seeking exploratory course (solid line in figure 5). Thus copying numerons in memory is a fundamental computational operation, one of the primitive operations in any modern computer.

(c) Biochemical logic gates

As schematized in figure 3, the addition of binary strings is efficiently implemented by sequentially operating logic gates. Biochemically realized logic gates are a fundamental aspect of the mechanisms by which the genetic code is read. Every

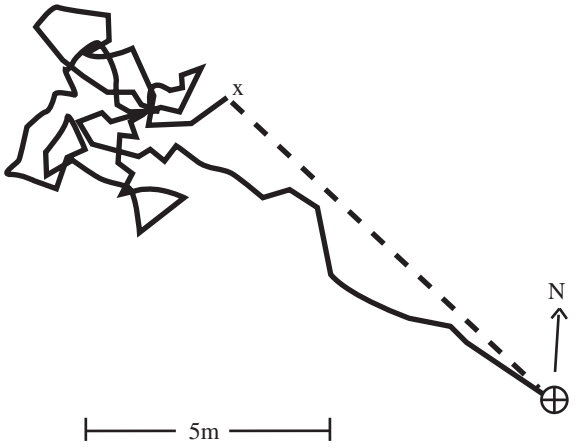

Figure 5. The homing path (dashed line) to its nest (N) taken by a foraging *Cataglyphus* ant after it discovered a large food source at x. Redrawn from Harkness and Maroudas [57] by permission of the author and publisher.

gene has two basic parts, a coding part and a promoter part. The binding of the transcription factor to the promoter of the gene triggers the transcription (reading) of the coding portion of the gene.

Transcription factors are proteins that selectively bind to the promoters of particular genes. Transcription factors often form dimers. A dimer is a transient binding of two proteins to produce a complex with properties distinct from its constituents. If a dimer binds to a promoter but neither of its constituents does, that biochemically implements an AND gate. If both constituents bind to the promoter but the dimer does not, that implements an XOR gate. If two transcription factors bind to the same promoter but do not dimerize, that implements an OR gate. These are the three gates out of which an adding mechanism is built (figure 3).

I am not sufficiently master of the intricacies of polynucleotide editing to spell out a complete story about how to construct a biochemical adding machine operating on strings composed of Watson–Crick base pairs. Nor would it be appropriate at this point to pursue these highly speculative observations further. Their purpose is only to make clear that one can readily imagine the exaptation of genetic machinery for the cell-intrinsic storage of acquired quantitative information and the performance of intracellular computations therewith. There is as much neurobiological plausibility in this scheme as in most systems-level computational schemes.

8. Conclusion

The time is ripe for a serious search for numbers in the brain. Before we set out, we should have as clear a conception as possible of what we are looking for. What constraints must hypothesized neurobiological realizations satisfy? How would we know that we had found them? I give 12 such constraints—11 in my list of Desiderata (§4) and a 12th in the section on circular computations (§6b). It is a daunting list; it is not easy to imagine in the abstract a scheme that satisfies these constraints, much less to find a neurobiological realization of that scheme. The fixed-point two's complement scheme has stood the test of time; no better scheme has been found after more than a century of intensive work on the physical realization of computing machines. It is an abstract scheme, with many possible physical embodiments. It is not easy to envision an embodiment in which numbers are encoded into plastic synapses. It is easier to envision an embodiment in which numbers are encoded into molecular strings with properties like those of the polynucleotides. Indeed, this is already being done in engineering laboratories investigating the use of bacterial DNA as the memory medium in conventional computers. The same cannot be said for plastic synapses; no one is investigating their use as a conventional memory mechanism.

Data accessibility. This article has no additional data.
Competing interests. I declare I have no competing interests.
Funding. I received no funding for this study.

References

1. Gelman R, Gallistel CR. 1978 *The child's understanding of number*. Cambridge, MA: Harvard University Press.
2. Cantlon JF, Merrit DJ, Brannon EM. 2016 Monkeys display classic signatures of human symbolic arithmetic. *Anim. Cogn.* **19**, 405–415. (doi:10.1007/s10071-015-0942-5)
3. Cordes S, Gelman R, Gallistel C, Whalen J. 2001 Variability signatures distinguish verbal from nonverbal counting for both large and small numbers. *Psychon. Bull. Rev.* **8**, 698–707. (doi:10.3758/BF03196206)
4. Cordes S, Gallistel CR, Gelman R, Latham P. 2007 Nonverbal arithmetic in humans: light from noise. *Percept. Psychophys.* **69**, 1185–1203. (doi:10.3758/BF03193955)
5. Gallistel CR. 1990 *The organization of learning*. Cambridge, MA: Bradford Books/MIT Press.
6. Gallistel CR, King AP. 2010 *Memory and the computational brain: why cognitive science will transform neuroscience*. New York, NY: Wiley/Blackwell.
7. Dehaene S, Piazza M, Pinel P, Cohen L. 2003 Three parietal circuits for number processing. *Cogn. Neuropsychol.* **20**, 487–506. (doi:10.1080/02643290244000239)
8. Jacob SN, Vallentin D, Nieder A. 2012 Relating magnitudes: the brain's code for proportions. *Trends Cogn. Sci.* **16**, 157–166. (doi:10.1016/j.tics.2012.02.002)
9. Nieder A, Diester I, Tudusciuc O. 2006 Temporal and spatial enumeration processes in the primate parietal cortex. *Science* **313**, 1431–1435. (doi:10.1126/science.1130308)
10. Tudusciuc O, Nieder A. 2007 Neuronal population coding of continuous and discrete quantity in the primate posterior parietal cortex. *Proc. Natl Acad. Sci. USA* **104**, 14 513–14 518. (doi:10.1073/pnas.0705495104)
11. Dehaene S, Meyniel F, Wacongne C, Wang L, Pallier C. 2015 The neural representation of sequences: from transition probabilities to algebraic patterns and linguistic trees. *Neuron* **88**, 2–19. (doi:10.1016/j.neuron.2015.09.019)
12. Ditz HM, Nieder A. 2016 Sensory and working memory representations of small and large numerosities in the crow endbrain. *J. Neurosci.* **36**, 12 044–12 052. (doi:10.1523/JNEUROSCI.1521-16.2016)
13. Queenan BN, Ryan TJ, Gazzaniga MS, Gallistel CR. 2017 On the research of time past: the hunt for the substrate of memory. *Ann. NY Acad. Sci.* **1396**, 108–125. (doi:10.1111/nyas.13348)
14. Davison M, McCarthy D. 1988 *The matching law: a research review*. Hillsdale, NJ: Erlbaum.
15. Castelli F, Glaser DE, Butterworth B. 2006 Discrete and analogue quantity processing in the parietal lobe: a functional MRI study. *Proc. Natl Acad. Sci. USA* **103**, 4693–4699. (doi:10.1073/pnas.0600444103)
16. Cordes S, Williams CL, Meck WH. 2007 Common representations of abstract quantities. *Curr. Dir. Psychol. Sci.* **16**, 156–161.

17. Gallistel CR. 2011 Mental magnitudes. In *Space, time and number in the brain: searching for the foundations of mathematical thought* (eds S Dehaene, L Brannon), pp. 3–12. New York, NY: Elsevier.
18. Walsh V. 2003 A theory of magnitude: common cortical metrics of time, space and quantity. *Trends Cogn. Sci.* **7**, 483–488. (doi:10.1016/j.tics.2003.09.002)
19. Livingstone MS. 2014 Symbol addition by monkeys provides evidence for normalized quantity coding. *Proc. Natl Acad. Sci. USA* **111**, 6822–6827. (doi:10.1073/pnas.1404208111)
20. Berkay D, Cavdaroglu B, Balcı F. 2016 Probabilistic numerical discrimination in mice. *Anim. Cogn.* **19**, 251–265. (doi:10.1007/s10071-015-0930-9)
21. Kheifets A, Gallistel CR. 2012 Mice take *calculated* risks. *Proc. Natl Acad. Sci. USA* **109**, 8776–8779. (doi:10.1073/pnas.1205131109)
22. Tosun T, Gür E, Balcı F. 2016 Mice plan decision strategies based on previously learned time intervals, locations, and probabilities. *Proc. Natl Acad. Sci. USA* **113**, 787–792. (doi:10.1073/pnas.1518316113)
23. Kheifets A, Freestone D, Gallistel CR. 2017 Theoretical implications of quantitative properties of interval timing and probability estimation in mouse and rat. *J. Exp. Anal. Behav.* **108**, 39–72. (doi:10.1002/jeab.261)
24. Gallistel CR, Krishan M, Liu Y, Miller RR, Latham PE. 2014 The perception of probability. *Psychol. Rev.* **121**, 96–123. (doi:10.1037/a0035232)
25. Ricci M, Gallistel CR. 2017 Accurate step-hold tracking of smoothly varying periodic and aperiodic probability. *Atten. Percept. Psychophys.* **79**, 1480–1494. (doi:10.3758/s13414-017-1310-0)
26. Menzel R, Eckoldt M. 2016 *Die intelligenz der bienen*. Munich, Germany: Knaus.
27. Egevang C, Stenhouse IJ, Phillips RA, Petersen A, Fox JW, Silk JRD. 2010 Tracking of Arctic terns *Sterna paradisaea* reveals longest animal migration. *Proc. Natl Acad. Sci. USA* **107**, 2078–2081. (doi:10.1073/pnas.0909493107)
28. Gallistel C, Gelman R. 1992 Preverbal and verbal counting and computation. *Cognition* **44**, 43–74. (doi:10.1016/0010-0277(92)90050-R)
29. Gibbon JG. 1977 Scalar expectancy theory and Weber's law in animal timing. *Psychol. Rev.* **84**, 279–325. (doi:10.1037/0033-295X.84.3.279)
30. Shannon CE. 1948 A mathematical theory of communication. *Bell Syst. Tech. J.* **27**, 379–423, 623–656. (doi:10.1002/j.1538-7305.1948.tb00917.x)
31. Goldman N, Bertone P, Chen S, Dessimoz C, Le Proust EM, Sipos B, Birney E. 2013 Towards practical, high-capacity, low-maintenance information storage in synthesized DNA. *Nature* **494**, 77–80. (doi:10.1038/nature11875)
32. Gerstner W, Kistler WM, Naud R, Paninski L. 2014 *Neuronal dynamics: from single neurons to networks and models of cognition*. New York: NY: Cambridge University Press.
33. Poo M *et al.* 2016 What is memory? The present state of the engram. *BMC Biol.* **14**, 40. (doi:10.1186/s12915-016-0261-6)
34. Gee CE, Oertner TG. 2016 Pull out the stops for plasticity. *Nature* **529**, 529. (doi:10.1038/529164a)
35. Graves A *et al.* 2016 Hybrid computing using a neural network with dynamic external memory. *Nature* **538**, 471–476. (doi:10.1038/nature20101)
36. Gallistel CR. 2017 The coding question. *Trends Cogn. Sci.* **21**, 498–508. (doi:10.1016/j.tics.2017.04.012)
37. Gallistel CR. 2017 The neurobiological bases for the computational theory of mind. In *Minds on language and thought: the status of cognitive science and its prospects* (eds RGD Almeida, L Gleitman), pp. 275–296. New York: NY: Oxford University Press.
38. Yates R. 2013 *Fixed-point arithmetic: an introduction.* (https://web.archive.org/web/20150912013429/http://www.digitalsignallabs.com/fp.pdf)
39. Halberda J, Feigenson L. 2008 Developmental change in the acuity of the 'number sense': the approximate number system in 3-, 4-, 5-, and 6-year-olds and adults. *Dev. Psychol.* **44**, 1457–1465. (doi:10.1037/a0012682)
40. Turing AM. 1936 On computable numbers, with an application to the *Entscheidungsproblem*. *Proceedings of the London Mathematical Society 2nd ser* **42**, 230–265.
41. Menzel R *et al.* 2011 A common frame of reference for learned and communicated vectors in honeybee navigation. *Curr. Biol.* **21**, 645–650. (doi:10.1016/j.cub.2011.02.039)
42. Menzel R, Lehmann K, Manz G, Fuchs J, Kobolofsky M, Greggers U. 2012 Vector integration and novel shortcutting in honeybee navigation. *Apidologie* **43**, 229–243. (doi:10.1007/s13592-012-0127-z)
43. Arcediano F, Escobar M, Miller RR. 2003 Temporal integration and temporal backward associations in humans and nonhuman subjects. *Learn. Behav.* **31**, 242–256. (doi:10.3758/BF03195986)
44. Arcediano F, Miller RR. 2002 Some constraints for models of timing: a temporal coding hypothesis perspective. *Learn. Motiv.* **33**, 105–123. (doi:10.1006/lmot.2001.1102)
45. Balsam PD, Drew MR, Gallistel CR. 2010 Time and associative learning. *Comp. Cogn. Behav. Rev.* **5**, 1–22. (doi:10.3819/ccbr.2010.50001)
46. Barnet RC, Cole RP, Miller RR. 1997 Temporal integration in second-order conditioning and sensory preconditioning. *Anim. Learn. Behav.* **25**, 221–233. (doi:10.3758/BF03199061)
47. Wittlinger M, Wehner R, Wolf H. 2006 The ant odometer: stepping on stilts and stumps. *Science* **312**, 1965–1967. (doi:10.1126/science.1126912).
48. Durgin FH, Akagi M, Gallistel CR, Haiken W. 2009 The precision of locomotor odometry in humans. *Exp. Brain Res.* **193**, 429–436. (doi:10.1007/s00221-008-1640-1)
49. Ratliff CP, Borghuis BG, Kao Y-H, Sterling P, Balasubramanian V. 2010 Retina is structured to process an excess of darkness in natural scenes. *Proc. Natl Acad. Sci. USA* **107**, 17 368–17 373. (doi:10.1073/pnas.1005846107)
50. Balasubramanian V, Sterling P. 2009 Receptive fields and functional architecture in the retina. *J. Physiol.* **587**, 2753–2767. (doi:10.1113/jphysiol.2009.170704)
51. Perge JA, Niven JE, Mugnaini E, Balasubramanian V, Sterling P. 2012 Why do axons differ in caliber? *J. Neurosci.* **32**, 626–638. (doi:10.1523/JNEUROSCI.4254-11.2012)
52. Laughlin SB. 2004 The implications of metabolic energy requirements for the representation of information in neurons. In *The cognitive neurosciences III* (ed. MS Gazzaniga), pp. 187–196. Cambridge, MA: MIT Press.
53. Laughlin SB. 2001 Energy as a constraint on the coding and processing of sensory information. *Curr. Opin. Neurobiol.* **11**, 475–480. (doi:10.1016/S0959-4388(00)00237-3)
54. Hasenstaub A, Otte S, Callaway E, Sejnowski TJ. 2010 Metabolic cost as a unifying principle governing neuronal biophysics. *Proc. Natl Acad. Sci. USA* **107**, 12 329–12 334. (doi:10.1073/pnas.0914886107)
55. Sterling P, Laughlin SB. 2015 *Principles of neural design*. Cambridge, MA: MIT Press.
56. Johansson F, Jirenhed D-A, Rasmussen A, Zucc R, Hesslow G. 2014 Memory trace and timing mechanism localized to cerebellar Purkinje cells. *Proc. Natl Acad. Sci. USA* **111**, 14 930–14 934. (doi:10.1073/pnas.1415371111)
57. Harkness RD, Maroudas NG. 1985 Central place foraging by an ant (*Cataglyphis bicolor* Fab.): a model of searching. *Anim. Behav.* **33**, 916–928. (doi:10.1016/S0003-3472(85)80026-9)