Solving the Problem of Negative Synaptic Weights in Cortical Models

Christopher Parisien
chris@cs.toronto.edu
Department of Computer Science, University of Toronto, Toronto, ON M5S 3G4, Canada

Charles H. Anderson
cha@wustl.edu
Department of Anatomy and Neurobiology, Washington University School of Medicine, St. Louis, MO 63110, U.S.A.

Chris Eliasmith
celiasmith@uwaterloo.ca
Centre for Theoretical Neuroscience, Departments of Philosophy and Systems Design Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada

In cortical neural networks, connections from a given neuron are either inhibitory or excitatory but not both. This constraint is often ignored by theoreticians who build models of these systems. There is currently no general solution to the problem of converting such unrealistic network models into biologically plausible models that respect this constraint. We demonstrate a constructive transformation of models that solves this problem for both feedforward and dynamic recurrent networks. The resulting models give a close approximation to the original network functions and temporal dynamics of the system, and they are biologically plausible. More precisely, we identify a general form for the solution to this problem. As a result, we also describe how the precise solution for a given cortical network can be determined empirically.

1 Introduction

It is often suggested that constructing detailed theoretical models of neural systems is indispensible to advancing our understanding of those systems (Aamodt, 2000). The degree of biological plausibility of these models corresponds to our certainty regarding whether actual biological systems operate in the same way as the model. Currently, biologically realistic models are designed largely using a bottom-up methodology: observable physiological constraints are respected by hand-wiring small populations of neurons together (see, e.g., Selverston & Miller, 1980; Ekeberg, Lansner,

& Grillner, 1995; Menschik & Finkel, 1999; McAlpine & Grothe, 2003). However, many important behaviors of cellular networks are not yet amenable to this method. The reasons vary from lack of physiological information to uncertainty regarding how to connect the elements to realize a given network-level function. As a result, a more common approach to modeling is to adopt a top-down method. In this case, models begin with more simplified neurons and then either learn the desired function (see, e.g., Zipser & Andersen, 1988; Deneve & Pouget, 2003) or analytically solve for the necessary weights to realize some function (see, e.g., Zhang, 1996; Pouget, Zhang, Deneve, & Latham, 1998; Seung, Lee, Reis, & Tank, 2000; Xie, Hahnloser, & Seung, 1996; Conklin & Eliasmith, 2005; Kuo & Eliasmith, 2005). While such networks solve the problems of bottom-up models, they have the converse problem of not being biologically plausible because of the initial simplifying assumptions.

One central, and unrealistic, feature of networks resulting from top-down approaches, whether learned or analytic, is that the resulting networks usually contain individual neurons with both negative (inhibitory) and positive (excitatory) projections to other neurons in the network. In contrast, there is a clear division between excitatory and inhibitory neurons in the brain. This fact has become referred to as "Dale's principle": that all projections from excitatory neurons are excitatory and all projections from inhibitory neurons are inhibitory, except in rare cases (Strata & Harvey, 1999; Burnstock, 2004; Marty & Llano, 2005). Thus, there is seldom a mix of excitation and inhibition resulting from a single neuron. This is in stark contrast to neurons in top-down models. To "fix" these models, we need to find some transformation of their connections such that all resulting neurons have either inhibitory or excitatory connections, and all weights from a single neuron must be positive (so that inhibitory neurons inhibit and excitatory neurons excite). We refer to this problem of eliminating negative weights from such model networks as the *negative weights problem*. Currently there exist no demonstrations of a general, biologically plausible solution to this problem.

One simple and intuitive solution to the problem is to introduce an inhibitory interneuron wherever a negative connection is needed, converting an excitatory signal into an inhibitory signal of equal magnitude. This approach was used by Churchland (1995) in a model of stereoptic vision. However, this solution remains biologically unrealistic for two reasons. First it assumes that there are as many inhibitory cells as there are inhibitorily connected cells in the original model. Clearly, this will vary from model to model, but can be as high as 50% of the neurons in the model (Pouget et al., 1998) despite the fact that only about 20% of neurons in cortex are inhibitory (Hendry & Jones, 1981; Gabbott & Somogyi, 1986). Second, this solution assumes that inhibitory neurons either receive input from or send output to a single excitatory cell, which is clearly not the typical case (Freund & Buzsáki, 1996). A cross-inhibitory circuit (Kamps & Velde, 2001) provides a solution for artificial neural networks, but it is both inefficient (introducing, at a minimum, a 600% increase in the number of neurons) and violates known biological connection constraints (Somogyi, Tamás, Lujan, & Buhl, 1998).

Building on work presented in Eliasmith and Anderson (2003), we demonstrate that our general solution solves the negative weights problem for arbitrarily connected feedforward or recurrent networks under linear decoding schemes. The result of our proposed transformation consists of a functionally equivalent network with no negative weights, whose proportion of inhibitory to excitatory neurons matches known constraints, and that results in only a 20% increase (i.e., the added inhibitory neurons) in the number of cells in the original network.

2 Correcting Negative Weights

The solution is a constructive transformation for a network with a set of mixed positive and negative synaptic weights between two neural ensembles. The method makes all of the original weights excitatory and introduces a small population of inhibitory interneurons. The end result provides two parallel pathways, direct (excitatory) and indirect (inhibitory), which together are functionally equivalent to the original set of synapses.

To realize the solution, we take a population of neurons to encode the value of a higher-level variable (e.g., stimulus parameter, behavioral parameter, internal state) in an ensemble of spike trains. A particular neuron in such a population has its encoding determined by its traditionally measured tuning curve (e.g., activity over a receptive field sensitive to a stimulus parameter such as orientation). Mathematically we can express any such encoding as

$$a_i(\mathbf{x}) = G_i\left[\alpha_i \left\langle \mathbf{x} \cdot \tilde{\boldsymbol{\phi}}_i \right\rangle_n + J_i^{bg}\right], \tag{2.1}$$

where the activity a_i (spike trains) of neuron i encodes aspects of the higher-level variable $\mathbf{x}$ as determined by its preferred direction vector $\tilde{\boldsymbol{\phi}}$ into a somatic current that includes a real-valued bias current, J_i^{bg}, to account for background activity, and a real-valued gain, α_i, that scales and converts units from the higher-level variable. The biophysical properties of the neuron, captured by the nonlinear function G_i, map this somatic current onto the neural activity as usual, using a standard neuron model such as Hodgkin-Huxley, Rose-Hindmarsh, or leaky integrate-and-fire (Eliasmith & Anderson, 2003). The angle brackets denote an inner product between the two vectors of dimension n. While we have expressed the neural activity $a_i(\mathbf{x})$ as a real-valued function (i.e., a rate code) to simplify the analysis, all of the simulations here are performed using spike trains. The methods used here are equivalent for both rate and spiking neuron models (Eliasmith & Anderson, 2003). This model can be used to match the experimentally observed behavior of populations of neurons. Typically the preferred direction vectors over the population are chosen to result in a statistically similar population of bell-shaped (cosine or gaussian) or monotonic tuning curves. However, a much wider variety of tuning curves can be captured by this model.

Once the encoding is defined, it is possible to find an optimal linear decoder for estimating the higher-level variable given the activities across the population of neurons. To determine the contribution of a particular neuron to the representation, we use a least-squares optimal linear regression method to find decoders $\boldsymbol{\phi}$ that minimize the error between the original variable and the estimate (Salinas & Abbott, 1994; Eliasmith & Anderson, 2003). This results in an estimate of the form

$$\hat{\mathbf{x}} = \sum_{i=1}^{N} a_i(\mathbf{x})\boldsymbol{\phi}_i, \tag{2.2}$$

where N is the number of neurons in the population, $a_i(\mathbf{x})$ is the activity (i.e., spike train) from neuron i and the $\boldsymbol{\phi}_i$, vectors of reals, are the optimal linear decoders (Eliasmith & Anderson, 2003). In combination, equations 2.2 and 2.1 define the neural representation of the variable $\mathbf{x}$.

Given this encoding and decoding, we can determine the connection weights between two populations, A and B, which have different tuning curves but represent the same variable. We assume that population A receives input of the current value of $\mathbf{x}$ and transmits this information to B. The set of connection weights should give a linear transformation from the output activities of A to the input currents of B. The connection weights for this function (i.e., the identity function $b(\mathbf{x}) = \mathbf{x}$) are

$$\mathbf{w}_{ji} = \alpha_j \left\langle \tilde{\boldsymbol{\phi}}_j \boldsymbol{\phi}_i \right\rangle_n, \tag{2.3}$$

where $\boldsymbol{\phi}_i$ is the decoding from a presynaptic neuron a_i, $\tilde{\boldsymbol{\phi}}_j$ is the preferred direction tuning for a postsynaptic neuron b_j, and α_j is the gain-and-conversion factor as defined above. These weights can be found by simply substituting the decoding equation for population A (see equation 2.2) into the encoding equation for population B (see equation 2.1). As a result, we can express the activities of the b_j neurons as

$$b_j(\mathbf{x}) = G_j \left[\sum_{i=1}^{N} \mathbf{w}_{ji} a_i(\mathbf{x}) + J_j^{bg} \right]. \tag{2.4}$$

This set of connection weights will implicitly decode the higher-level variable from the spikes of one population of neurons and convert this to somatic currents for the next population. These particular weights (see

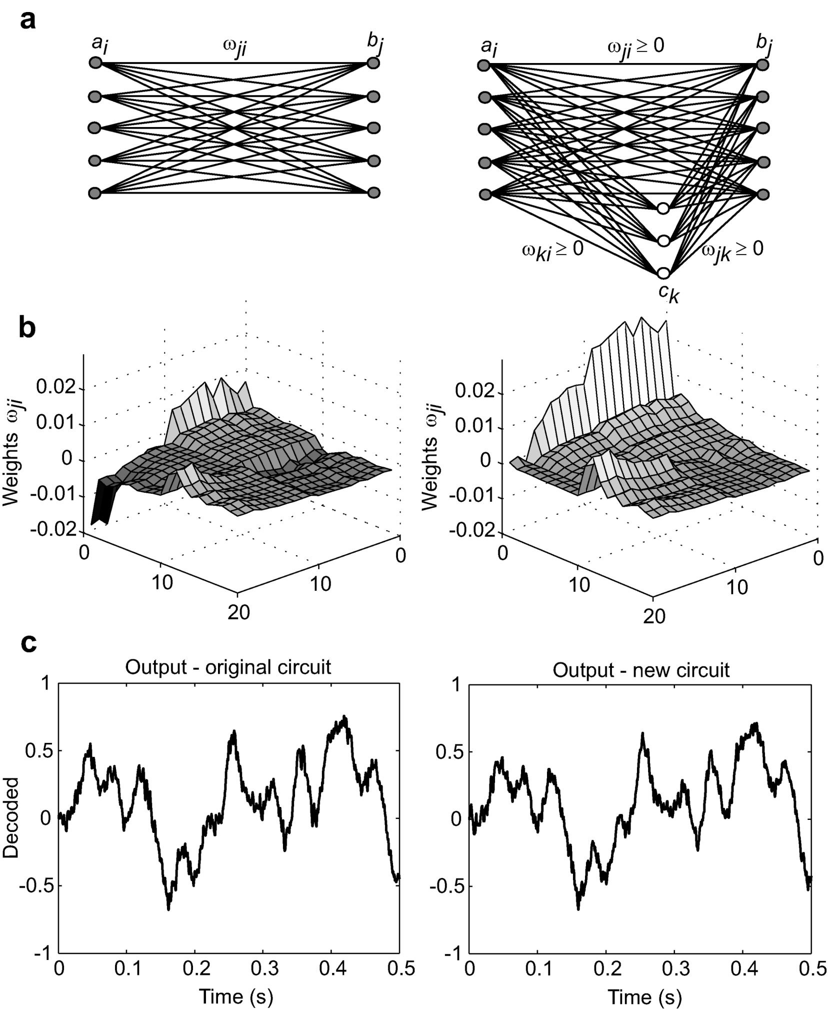

Figure 1: Feedforward networks computing the identity function ($b(\mathbf{x}) = \mathbf{x}$). (a) Structure of the networks before and after the elimination of negative weights. The neurons a_i, b_j, and c_k form the populations A, B, and C, respectively. (b) Synaptic weights for an example with 20-neuron excitatory populations. (c) Decoded output from the 20-neuron examples, stimulated with 30 Hz band-limited white noise over 0.5 s. This demonstrates that the information passed is largely unaffected by the change in network topologies.

Figure 1b) effectively transfer the value of the higher-level variable from the sending to receiving populations of neurons (see Figure 1c).

While this kind of communication channel is a useful example, it performs no interesting function. It is more important to be able to perform arbitrary transformations on the encoded variables. Conveniently, the same methods can be employed. Instead of finding decoders $\boldsymbol{\phi}$ to decode an estimate of $\mathbf{x}$ (i.e., computing the identity function), the same linear least-squares method can be used to provide decoders $\boldsymbol{\phi}^{g(\mathbf{x})}$ for some arbitrary function $g(\mathbf{x})$. These new decoders, placed into equation 2.3, then provide the synaptic weights needed to compute an estimate of $g(\mathbf{x})$, which is represented in the receiving population, B.

In either case, however, there are no constraints on the sign (positive or negative) of the weights governing the projection from A to B. To remove the negative weights from such a network, we can systematically manipulate the decoders and encoders. We present a two-step method for effecting this transformation. We begin by examining the feedforward network (see Figure 1a), although we eventually show that the method extends to recurrent connections as well (see Figure 2a).

Suppose that there is a connection from a neural ensemble A to an ensemble B that contains a mixture of positive and negative synaptic weights. Suppose that A encodes a value of the variable $\mathbf{x}$ and transmits some function of $\mathbf{x}$ to B. To guarantee that all weights connecting A and B are positive, we can add a bias to each of b_j 's connections equal to the magnitude of its largest negative weight (from A). Performing this step (step 1) systematically for each neuron in B will make all of the connection weights positive. However, doing this will also result in excess current flowing into neurons in B. That is, since we have increased only connection weights in the circuit, we are guaranteed that there will be more current flowing into at least some postsynaptic cells after this augmentation. However, our goal is to preserve the function of the connection, so we must find a way to balance this excess current by using an appropriate inhibitory input (step 2). To effect this balance, we first determine what new higher-level signal (i.e., function of $\mathbf{x}$) has been introduced by the weight augmentation. We then remove this added bias by appropriately decoding, in population B, a set of inhibitory neurons that also receives information about changes in $\mathbf{x}$ from population A. In short, determining the excitatory bias that results from the weight augmentation (step 1) allows us to correct it with an ensemble of inhibitory interneurons (step 2). What is unique is that we are able to do this abstractly with higher-level variables and so have great flexibility in relating the solution to specific neural realizations (e.g., we can vary the number of inhibitory neurons, the response properties of the various populations, or the functions being computed in the circuit).

Step 1: Given a communication channel as defined above, the currents flowing into the b_j neurons are as follows:

$$J_j(\mathbf{x}) = \sum_{i=1}^{N} \omega_{ji}^{orig} a_i(\mathbf{x}) + J_j^{bg}, \tag{2.5}$$

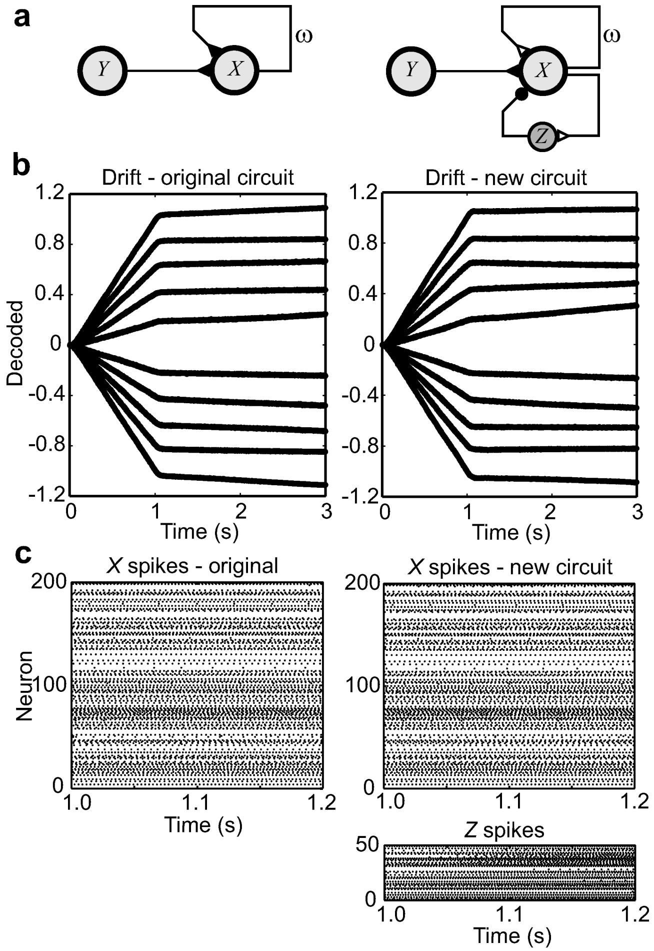

Figure 2: Recurrent networks acting as a neural integrator ($\dot{x} = 0$). (a) Structure of the networks before and after the elimination of negative weights. Neural ensembles are drawn as single units for simplicity. The recurrent synapses ω are targeted by the transformation. (b) Decoded output from a neural integrator before and after the elimination of negative weights. Inputs are 1 s square pulses at various levels, and the networks are run for 3 s. The individual curves are from each of 10 distinct simulations. This demonstrates that the network dynamics are largely unaffected by the change in network topologies. (c) A section of spike trains for the recurrent neurons and the interneurons, showing characteristic firing patterns.

where ω_{ji}^{orig} contain both positive and negative connection weights. We augment each of these weights by a positive amount, equal in magnitude to the most negative weight entering b_j:

$$\omega_{ji} = \omega_{ji}^{orig} + \Delta\omega_j \tag{2.6}$$

$$\Delta\omega_j = -\min_i\left(\omega_{ji}^{orig}\right). \tag{2.7}$$

Now the current entering b_j is given by

$$J_j(\mathbf{x}) = \sum_{i=1}^{N} \omega_{ji} a_i(\mathbf{x}) + J_j^{bg} \tag{2.8}$$

$$= \sum_{i=1}^{N} \omega_{ji}^{orig} a_i(\mathbf{x}) + \sum_{i=1}^{N} \Delta\omega_j a_i(\mathbf{x}) + J_j^{bg}, \tag{2.9}$$

where ω_{ji} are all nonnegative. The bias term $\sum_{i=1}^{N} \Delta\omega_j a_i(\mathbf{x})$ results in a positive excess current flowing into each b_j neuron. We will introduce a population of inhibitory interneurons to correct this bias.

Step 2: To define this inhibitory population, it is helpful to consider a higher-level signal represented by the bias current. Decomposing the bias term using the encoder-decoder relationship of equation 2.3,

$$\sum_{i=1}^{N} \Delta\omega_j a_i(\mathbf{x}) = \sum_{i=1}^{N} \alpha_j \tilde{\phi}_j^f \phi^f a_i(\mathbf{x}) \tag{2.10}$$

$$= \alpha_j \tilde{\phi}_j^f \sum_{i=1}^{N} \phi^f a_i(\mathbf{x}) \tag{2.11}$$

$$= \alpha_j \tilde{\phi}_j^f f(\mathbf{x}), \tag{2.12}$$

where $\tilde{\phi}_j^f$ and ϕ^f, respectively, are the encoders and decoders for a scalar signal $f(\mathbf{x})$. The ϕ^f can be indexed by each input neuron a_i, as in equation 2.3, but for simplicity, we define each decoder to be a small, uniform, positive constant. This choice of decoders is not important for our exposition of the method. However, because this choice defines $f(\mathbf{x})$, the bias function, and because the bias function determines weights to and from the inhibitory population (as we discuss shortly), our choice of decoders is important to the precise topology of the network. In addition, an assumed bias function is empirically testable, thus connecting our method to experimental evidence, as we describe in section 5. This means that certain choices for $f(\mathbf{x})$ will be determined to be appropriate and others will not be. Indeed, it is quite likely that $f(\mathbf{x})$ varies with different kinds of inhibitory neurons, different transmitters, different anatomical regions, and so on. Thus, our purpose in having characterized the method in terms of a bias function is to provide a degree of flexibility that allows its application across a wide variety of circuits in an empirically relevant manner.

In this simple case, we scale ϕ^f so that $f(\mathbf{x})$ remains within the range [0,1] to be consistent with the range of the representation of $\mathbf{x}$ in the circuit (see Figures 1 and 2). Having defined ϕ^f, we can solve for the function encoders:

$$\tilde{\phi}_j^f = \frac{\Delta\omega_{ji}}{\alpha_j \phi^f}. \tag{2.13}$$

As a brief aside, note that in cases where the A population always has some neuron active for a value of $\mathbf{x}$ (which we assume is the common case), the bias function will be nonzero for the entire domain of $\mathbf{x}$. Let us then break the bias function into a constant part (the minimum value) and a variable part, so that $f(\mathbf{x}) = f_1 + f_2(\mathbf{x})$, where $f_1 = \min(f(\mathbf{x}))$. We have found that it is advisable to slightly underestimate this minimum to compensate for decoding errors and noise in the circuit. Since only the variable component is dependent on $\mathbf{x}$, the constant can be absorbed into the background current of the B neurons. This reduces the amount of inhibitory current that must be provided by the interneurons, which can prove useful for reducing the firing rates of the inhibitory cells. Note, however, that splitting the bias function into two parts is not essential to the method. Presumably empirical evidence for a given circuit would determine whether this variation is relevant.

Adopting this decomposition of the bias function, we can now express the bias current as follows:

$$J_j^f(\mathbf{x}) = \alpha_j \tilde{\phi}_j^f f(\mathbf{x}) \tag{2.14}$$

$$= \alpha_j \tilde{\phi}_j^f (f_1 + f_2(\mathbf{x})) \tag{2.15}$$

$$= \alpha_j \tilde{\phi}_j^f f_1 + \alpha_j \tilde{\phi}_j^f f_2(\mathbf{x}). \tag{2.16}$$

The first component of equation 2.16 is constant, so it can be incorporated into the background current of the b_j neurons as suggested:

$$J_j^{bg_{new}} = J_j^{bg} - \alpha_j \tilde{\phi}_j^f f_1. \tag{2.17}$$

The second part of equation 2.16 is dependent on $\mathbf{x}$ and so must be corrected by an explicit input from the inhibitory population C.

The first step in creating this inhibitory population is to define how the interneurons c_k will be tuned. Since the bias function $f(\mathbf{x})$ is always positive, these neurons will receive only inputs greater than zero. Specifically,

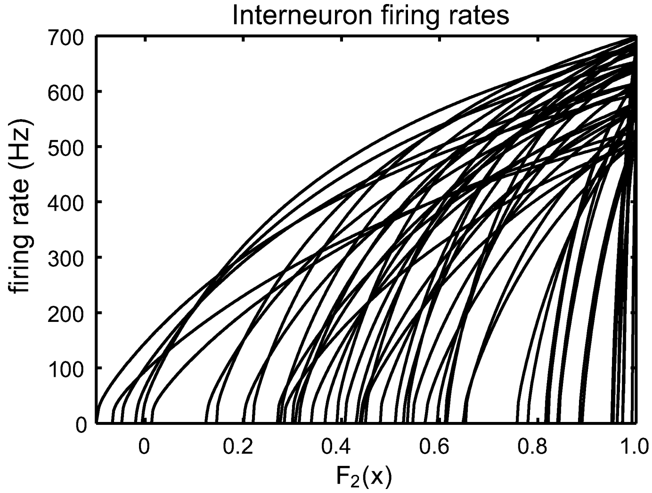

Figure 3: Sample tuning curves for an ensemble of 50 interneurons. The small, negative domain allows some neurons to show substantial minimal firing at low values of the positive-only input.

recall that we have scaled ϕ^f to be in the range $[0, 1]$. We set all of the encoders $\tilde{\phi}_k^f$ in C to 1, giving only positively sloped ("on") neurons. Since the decoders ϕ^f are already positive, all of the connection weights from A to C will now be positive. To determine appropriate thresholds for the inhibitory neurons, we can turn to experimental studies of the tuning sensitivities of interneurons in the striate cortex (Azouz, Gray, Nowak, & McCormick, 1997). In response to a range of direct injected current, neurons showed near-zero firing at low current. However, in response to visual stimuli, some neurons showed a substantial minimal response (i.e., nonzero firing to no stimulus, that is, at $\mathbf{x} = 0$). We therefore allow firing thresholds in C to reach -0.1 which results in neurons with similar minimal firing. Figure 3 shows a sample set of tuning curves for c_k neurons in our networks, illustrating this property. Functionally, minimal firing is important as it allows greater neural activity at low magnitudes of $\mathbf{x}$, substantially improving the representational accuracy of $f(\mathbf{x})$ in C.

We can now define the soma current delivered to the c_k neurons:

$$J_k(\mathbf{x}) = \alpha_k \tilde{\phi}_k f(\mathbf{x}) + J_k^{bg} \tag{2.18}$$

$$= \alpha_k \sum_i \phi^f a_i(\mathbf{x}) + J_k^{bg} \tag{2.19}$$

$$= \sum_i \omega_{ki} a_i(\mathbf{x}) + J_k^{bg}, \tag{2.20}$$

where $\omega_{ki} = \alpha_k \phi^f$ are the positive input weights. Since we have corrected for the constant part of $f(\mathbf{x})$ in the background current of the b_j neurons, we adjust the background current for each c_k:

$$J_k^{bg_{new}} = J_k^{bg} - \alpha_k f_1. \tag{2.21}$$

This way, C will output only the needed correction, the variable component of $f(\mathbf{x})$, $f_2(\mathbf{x})$, defined earlier.

We can now use C to estimate $f_2(\mathbf{x})$ by finding the appropriate decoders as in equation 2.2. Specifically, we find a set of positive decoders ϕ_k to decode the variable component:

$$\widehat{f_2}(\mathbf{x}) = \sum_k \phi_k c_k(f(\mathbf{x})). \tag{2.22}$$

Importantly, our earlier choice of the bias function plays a critical role in determining the value of these decoders. This is because these decoders are being found in order to estimate (part of) that original bias function. If the bias function changes, so will the ϕ_k, and hence so will the connection weights from C to B.

Note that even though some of the c_k neurons may be sensitive to negative values, these values never need to be represented since the bias function is always defined to be positive. We can thus ignore negative values when solving for the ϕ_k, guaranteeing only positive decoders. Note also that while larger populations of neurons improve the accuracy of the representation of this function in C (Eliasmith & Anderson, 2003), a highly accurate representation is provided by using 20% of the total number of cells in A and B (see section 4), meeting the biological constraint that originally motivated this method. In general, the proportion of inhibitory cells is determined by our knowledge of the specific biological circuit being modeled.

We can now complete our circuit by incorporating the additional current from the inhibitory neurons into B. The total current into the b_j neurons is now given by the total from A and C:

$$J_j(\mathbf{x}) = \sum_i \omega_{ji} a_i(\mathbf{x}) + J_j^{bg_{new}} - \sum_k \omega_{jk} c_k(f(\mathbf{x})), \tag{2.23}$$

where $\omega_{jk} = \alpha_j \tilde{\phi}_j^f \phi_k$. The resulting network preserves the original transformation using only positive synaptic weights as desired.

In the case of a recurrent circuit, we simply substitute b_i neurons for the a_i neurons in the above derivation, so that we consider synapses from neurons b_i to b_j. Otherwise the method is identical. Note also that we made no limiting assumptions regarding the function of the original weights or the nature of the representation across the populations. As a result, the method is generally applicable.

To summarize, the end result of the application of this method is the transformation of an original mixed weight circuit to a new positive weight circuit with parallel pathways that performs the same neural computation as the original circuit. A direct excitatory pathway computes the intended function with an extra bias component resulting from a weight augmentation that guarantees only positive connections. An inhibitory pathway comprising interneurons corrects for that bias also using only positive weights. The target neurons thus keep the same soma current and the same representation that they had in the original network. This transformation is clearly repeatable for multiple-layer or recurrent networks. The resulting circuits bear a strong resemblance to biological networks since feedforward and feedback inhibition are common cortical features, the proportion of inhibitory and excitatory connections can be made to reflect that observed in cortical networks, and the structure as shown in Figure 1a is consistent with highly connected interneurons (Buzsáki, 1984; Freund & Buzsáki, 1996; Somogyi et al., 1998).

3 Network Simulations

To demonstrate the solution, we apply the transformation to models of two typical network structures, feedforward and recurrent. Both the original and transformed networks are run in the Neural Engineering Simulator (NESim), which provides a Matlab environment for developing and testing computational models under the framework described here and by Eliasmith and Anderson (2003).[1] All simulations use leaky integrate-and-fire (LIF) neurons (Koch, 1999) with 10% background noise and realistic postsynaptic currents (PSCs; see below).

3.1 Feedforward Networks. The feedforward networks are an implementation of the general structure defined in equations 2.1 to 2.4 and corrected using the method defined above. While the simulations use LIF neurons, the construction method itself is independent of the LIF model, and requires no additional considerations to account for its temporal aspects. We aim to demonstrate that network performance is robust to the temporal changes introduced by the inhibitory pathway.

The feedforward networks consist of two populations, A and B, of 200 LIF neurons each, or 300 each in the case of vector representation. A encodes the input signal, which it passes on to B. To determine the relevant biophysical parameters, we simulate hippocampal principal neurons with

[1] NESim may be found at http://compneuro.uwaterloo.ca/. The method to correct negative weights has been added to NESim as an automated process.

AMPA-mediated PSCs with decay constants of $\tau = 5$ ms (Jonas, Major, & Sakmann, 1993). The transformation introduces a population of either 50 or 75 fast-spiking (FS) inhibitory interneurons, to match the 20% proportion found in cortex (Hendry & Jones, 1981; Gabbott & Somogyi, 1986). Hippocampal AMPA-mediated synapses on inhibitory interneurons are fast (Geiger, Lübke, Roth, Frotscher, & Jonas, 1997; Carter & Regehr, 2002; Walker, Lawrence, & McBain, 2002), being well modeled by PSCs with $\tau = 1$ ms for these synapses. Slower GABA-mediated inhibitory synapses with $\tau = 4$ ms project onto the B neurons (Bartos, Vida, Frotscher, Geiger, & Jonas, 2001; Bartos et al., 2002). Thus, signals from both pathways arrive with similar amounts of delay at target cells. The parameters for the LIF neuron models are as follows: membrane time constant $\tau_{RC} = 10$ ms; refractory period $\tau_{ref} = 1$ ms; peak firing rates over the represented range are chosen from a uniform distribution over 200 to 400 Hz. Inhibitory interneurons are similarly modeled, except that their firing saturates at a higher rate: between 500 and 700 Hz (Azouz et al., 1997).

For the scalar communication channel, neurons respond to values within the range $[-1, 1]$. In the vector example, neurons are tuned to magnitudes less than or equal to 2. In the polynomial transformation, A neurons respond over $[-1$ to $1]$, and B neurons respond over $[-1.5$ to $1.5]$. The inhibitory neurons in C always respond over the range $[-.1, 1]$ as discussed earlier. In all cases, the thresholds at which neurons begin to fire are chosen from a uniform distribution over the relevant range.

Two of the feedforward networks represent scalars in both ensembles, and one network represents a three-dimensional vector. Connection weights between these ensembles are found by least-squares optimization (Eliasmith & Anderson, 2003), depending on the transformation desired. For the simple communication channel, we let $g(\mathbf{x}) = \mathbf{x}$, as discussed in the previous section, and for the polynomial transformation, we let $g(x) = 0.5x^2 - x$.

In the subsequent simulations, we generate 10 networks to determine the systematic effects of the network transformation. For each of the 10 networks, we independently transform the network to remove negative weights and then simulate it over a 1 second period. We use 30 Hz band-limited white noise signals to test the scalar networks. For the vector networks, the three-dimensional vector input follows a helical path, $(\sin(40t), \cos(40t), \sin(10t))$.

3.2 Postsynaptic Currents and Temporal Decoding. Spikes in the network simulation produce PSCs in postsynaptic cells with exponential decay. A model of a simple exponential PSC is given by

$$h'(t) = \frac{1}{\tau} e^{-t/\tau}, \tag{3.1}$$

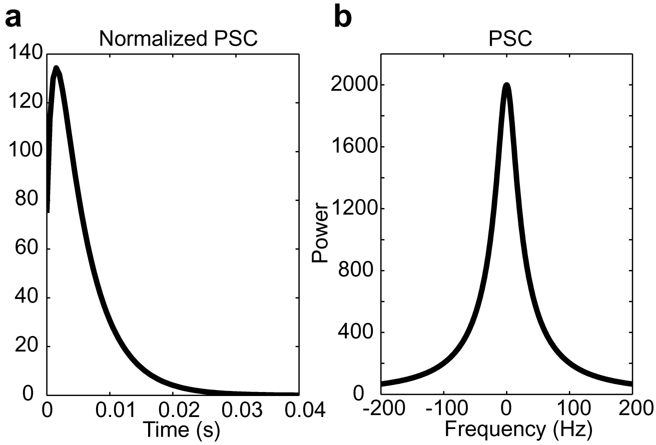

Figure 4: Postsynaptic current. (a) A 5 ms PSC in the time domain. The double filter gives the PSC a noninstantaneous rise time. (b) Frequency response of the PSC.

where τ is the synaptic time constant. However, this model results in instantaneous rise times in the PSC. For the simulations in this work, we use a more realistic PSC, defined as the application of two such exponentials. Specifically, this PSC model is the convolution of two of the above exponential filters, $h'_1(t) * h'_2(t)$, where $h'_1(t)$ and $h'_2(t)$ have different time constants. We set the constant for the first filter to be the primary decay constant, and we set the second at 20% of this. As shown in Figure 4, the PSC now has a noninstantaneous rise time, more typical of PSCs observed in vivo (Jonas et al., 1993). The additional filtering of the PSC does not substantially change the dynamics described for the recurrent networks.

Given the expression for the PSC, $h(t)$, we can decode an estimate for a temporal signal $\mathbf{x}(t)$ from a population of spiking neurons using the following:

$$\hat{\mathbf{x}}(t) = \sum_{i,n} \boldsymbol{\phi}_i h(t - t_{in}), \tag{3.2}$$

where t_{in} give the spike time occurrences from neuron i. Temporal decoding, then, is the summation of individual PSCs over time using the same decoders $\boldsymbol{\phi}_i$ found earlier. We introduce noise into the simulations using gaussian spike time jitter with zero mean. This equation is analogous to equation 2.2, and does not change the rest of the derivations. Input to the first layer of neurons (here, layer A) is given by multiplying the signal $\mathbf{x}(t)$ by the encoders $\tilde{\boldsymbol{\phi}}_i$ and directly injecting the current into the neuron somas as in equation 2.1.

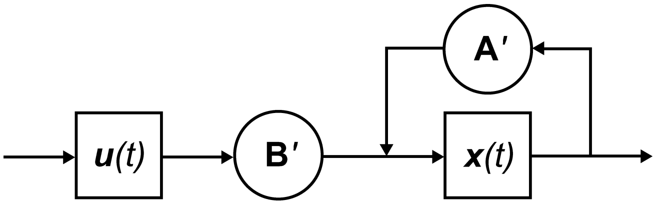

Figure 5: Higher-level block diagram for the neural integrator. The boxes denote the input and state variables, and the circles denote the input and dynamics matrices. For clarity, we use the state variable, $x(t)$, in place of the transfer function, $h'(t)$, to designate the population.

3.3 Recurrent Networks. To implement a network with recurrent connections within an ensemble, we use an adaptation of modern control theory to neural dynamics as described in Eliasmith and Anderson (2003). There it is shown that the state equation for a linear dynamical system implemented in a biologically plausible network can be written as

$$\mathbf{x}(t) = h'(t) * [\mathbf{A}'\mathbf{x}(t) + \mathbf{B}'\mathbf{u}(t)], \tag{3.3}$$

for the state variables $\mathbf{x}(t)$, input $\mathbf{u}(t)$, postsynaptic current response $h'(t)$, neural dynamics matrix $\mathbf{A}' = \tau_{synapse}\mathbf{A} + \mathbf{I}$, and neural input matrix $\mathbf{B}' = \tau_{synapse}\mathbf{B}$, where $\mathbf{A}$ and $\mathbf{B}$ are the dynamics and input matrices for any linear dynamical system in standard form (the dimensions of the matrices reflect the dimension of the represented signal). Applying this characterization to a scalar integrator (i.e., where $\mathbf{A} = [0]$ and $\mathbf{B} = [1]$) results in $\mathbf{A}' = [1]$ and $\mathbf{B}' = [\tau_{synapse}]$ defining the neural circuit (Eliasmith & Anderson, 2003). This circuit is shown in Figure 5.

To explicitly implement this circuit, we define two neural ensembles, H and G, to represent the two state variables $\mathbf{u}(t)$ and $\mathbf{x}(t)$, respectively. For this simulation, H has 100 LIF neurons, and G has 200 neurons. We find the connection weights by substituting equation 3.3 into the encoding equation equation 2.1 for G, giving:

$$\begin{aligned} g_i(x(t)) &= G_i\left[\alpha_i \langle x(t) \cdot \tilde{\phi}_i\rangle + J_i^{bg}\right] \\ &= G_i\left[\alpha_i \langle h'(t) * [A'x(t) + B'u(t)]\tilde{\phi}_i\rangle + J_i^{bg}\right] \\ &= G_i\left[\sum_i \omega_{ji} g_i(x(t)) + \sum_i \omega_{il} h_l(u(t)) + J_i^{bg}\right], \end{aligned}$$

where the recurrent connection weights $\omega_{ji} = \alpha_i \tilde{\phi}_i \phi_j A'$ and the weights from H to G are $\omega_{il} = \alpha_i \tilde{\phi}_i \phi_l B'$. The resulting network contains fully recurrent connections among the neurons of G.

We run this simulation using recurrent NMDA-mediated synapses with $\tau = 150$ ms as is common (Kinney, Peterson, & Slater, 1994; Seung, 1996). In keeping with the previous method, we introduce a population of 40 inhibitory interneurons I into the recurrent connections. We do not modify the input connections, so all changes in network function are attributable to the transformed recurrent weights. Synapses from G to I are NMDA mediated, with $\tau = 150$ ms. Synapses from I to G are GABA mediated, with $\tau = 4$ ms. The LIF neuron parameters for the integrator network are otherwise the same as for the neurons of the feedforward networks.

For each of 10 networks generated of this type, we independently transform the network to correct negative weights and test each version with square-pulse input signals. We test 10 pulses of amplitudes from −1 to 1 at intervals of 0.2 (omitting 0). We test each pulse for 1 s and observe the drift of the signals for 2 s after the pulse.

4 Results

We determine the effectiveness of the method by generating a variety of network types. We investigate linear, nonlinear, and multidimensional transformations in feedforward networks, as well as recurrent networks simulating a neural integrator. The results shown here examine changes in the accuracy and dynamics of the networks as a result of correcting negative synaptic weights. These results demonstrate that the method is effective in preserving the intended functionality of the networks.

We first consider a feedforward network that acts as a simple communication channel, which sends a scalar time-varying signal between two 200-neuron populations A and B. The transformation introduces a population of inhibitory interneurons, as demonstrated in Figure 1a. We simulate with parameters based on hippocampal networks by using fast AMPA-mediated PSCs (Jonas et al., 1993). Figure 1b demonstrates that all negative connection weights are eliminated with this method. Figure 1c shows the results of using a scalar input defined by 30 Hz band-limited white noise with root mean square (RMS) = 0.5 on both the original and transformed networks. Table 1 summarizes the effects of the transformation on the error in various representations and computations averaged over 10 different networks (i.e., networks with independently and randomly chosen neuron response functions) of each type. As demonstrated there, the transformation is both effective, with differences in RMS error below 0.25%, and general, being robust to the dimension of the represented signal or the linearity of the computation performed by the network. While the behavior of the transformed network does not perfectly match that of the original, the differences are reasonable and maintain the intended behavior of the network.

Recurrent networks present unique challenges for introducing inhibition. If the inhibitory pathway is too slow, unbalanced excitatory feedback could cause instability. As well, errors introduced by the additional pathway

Table 1: Network Simulation Results.

Network Type	Original RMS Error (% of Magnitude)	Corrected RMS Error (% of Magnitude)
Scalar	2.68	2.68
Vector	5.61	5.85
Polynomial transformation	3.49	3.46
Learned weights	4.35	4.23
	Original drift τ (s)	Corrected drift τ (s)
Integrator	34.6	27.3

Notes: The RMS error is in the signals represented by 10 networks before and after the weight correction. The error is given as the difference in magnitude of the represented value from the ideal result. The integrator drift is the average time constant for an exponential drift in the signal.

could substantially change the dynamics of the system. To test the response of such a network, we examine a model of the scalar neural integrator described above, which has been implicated in eye control in the oculomotor system (Robinson, 1989; Fukushima, Kaneko, & Fuchs, 1992; Moschovakis, 1997; Eliasmith & Anderson, 2003). A side-by side comparison between the integration of input signals before and after this transformation is shown in Figure 2b. Table 1 summarizes the average drift speed time constant difference between 10 sets of original and transformed networks. The transformed networks largely preserve the dynamics of the original networks. We do see some adverse effects, as the drift rate increases by 21.1% on average and the standard deviation of $1/\tau$ increases significantly from 0.0416 to 0.0557. However, the networks still provide a reasonable approximation to the intended behavior.

4.1 Arbitrary Weight Structures. While general, the example networks we have considered so far are generated by the analytic methods described by Eliasmith and Anderson (2003). In particular, we have expressed the synaptic weights as the product of decoding and encoding components and the relevant dynamics and input matrices. Since this is a new method, the majority of models do not use it. As a result, for the solution to be usable in general, it must be possible to take an arbitrary mixed-sign connection matrix and transform it into a biologically plausible circuit. Conveniently, the method described earlier extends to models with experimentally determined, prespecified, or learned weights. In these cases, the encoding-decoding abstraction defined here has not been used; it is not possible to decompose the weights into separate encoding and decoding components. Notice, however, that the original weights $\mathbf{w}_{ji} = \alpha_j \langle \tilde{\boldsymbol{\phi}}^{\mathbf{y}}_j \boldsymbol{\phi}^{\mathbf{x}}_i \rangle_n$ only need to be decomposed for the encoder-decoder relationship of the bias function in equation 2.13. Thus, it is sufficient to know only the gain-and-conversion factor α_j to decompose the weights as needed. To find α_j, we can simply fit the neural nonlinearity $G_j[\cdot]$ to the neural responses assumed in the original model (many models will in fact already have defined an equivalent parameter). Once α_j is determined, we can correct negative weights for an arbitrary circuit.

To demonstrate this method, we use a Hebbian rule to learn a communication channel like that shown in Figure 1a (Eliasmith & Anderson, 2003). As in the earlier example, we connect two populations of 200 LIF neurons each, except that the initial connection weights ω_{ji} are chosen from a uniform distribution over the same range as found in one of the existing networks. We update the connection weights using a standard delta rule, seeking to minimize the variance in the neural responses:

$$\Delta\omega_{ji} = -\kappa(b_j(x) > 0)a_i(x)(b_j(x) - \bar{b}_j(x)), \tag{4.1}$$

where κ is the learning rate and $\bar{b}_j(x)$ is a running mean of the activity. Once the weight updates stabilize, we perform the transformation on the network.

Table 1 shows the RMS error averaged over 10 learned networks before and after the transformation tested on 30 Hz band-limited white noise. As with the analytic circuits, the RMS difference between the mixed-weight and positive weight circuits is small and remains below 0.15%.

5 Discussion

Both feedforward and recurrent networks, and both analytic and learned circuits, can be effectively transformed with this method. In each of the feedforward cases, the circuits demonstrate only minor changes in performance as measured by RMS error, indicating that the method is robust and largely insensitive to the computation performed. It is notable that the dynamics of the neural integrators changes, as shown by a slightly faster mean drift rate. This is expected, since the transformed network is subject to a small delay and extra noise in the inhibitory path. Practically speaking, this suggests that models that allow the presence of negative weights (i.e., ignore interneurons) may systematically overestimate network performance for a given number of neurons. However, the intended dynamics remain highly similar, preserving the overall function of the network.

Importantly, the synaptic organization required by this method is strongly similar to that found in cortical networks. Depending on the type of network transformed, we introduce either feedforward or feedback inhibition (Buzsáki, 1984; Somogyi et al., 1998). Since the inhibitory neurons are connected as an ensemble, each neuron receives a large number of inputs and projects a large number of outputs (on the order of the connected populations). Again, this matches cortical observations (Freund & Buzsáki, 1996), eliminating a central shortcoming of past solutions, as outlined in section 1. The resulting inhibitory circuits are also robust; in virtue of our high-level solution to the problem (i.e., having adopted a population coding approach), the loss of a small number of neurons or connections will not dramatically affect the overall accuracy of the network (Eliasmith & Anderson, 2003). For similar reasons, there are no constraints on how many inhibitory neurons must be used. Although more neurons result in a better correction of the bias, this is a parameter that can be experimentally informed. In short, unlike previous techniques, this method can be applied to a variety of networks while maintaining biological plausibility.

5.1 Predictions. As noted during the description of the method, there are some steps in the transformation that are subject to experimental tests. Recall that in the network transformation, we add a bias to each of the synaptic weights in order to make them positive (i.e., all excitatory). For each postsynaptic neuron b_j, we add a bias to all of its incoming weights equal to the magnitude of its largest negative weight. These augmented weights increase the level of current flowing into the postsynaptic neurons, which is then corrected by the new inhibitory interneurons. As explained earlier, the weight bias computes some additional function of the higher-level values represented by the neurons. We let this function, which we call the bias function, take on an arbitrary form by defining a set of linear decoding weights for the function to be uniform positive constants. As an example, given monotonic response functions, this bias function will take on an approximately parabolic shape, where the decoded value increases with the magnitude of the input. This is an interesting prediction about the response properties of the neurons involved: while the excitatory neurons are tuned to the time-dependent signal ($\mathbf{x}(t)$), the inhibitory neurons are tuned to the sum of their inputs.

While the bias function is an abstract computation, the resulting current must be explicitly processed by the inhibitory interneurons and as such will determine their firing patterns. It is here that this solution can be informed by experimental results. We have let the bias function take a form that is effective, but it is not based on the observed effects of neural inhibition in a circuit. As a result, we can consider the effects of removing inhibition from this model circuit (see Figure 6). Specifically, we can predict that if this is the correct biasing current, a GABA$_A$-receptor antagonist bicuculline experiment that blocks inhibition should result in deviations like those shown in Figure 6. Such an experiment, even if it contradicts this specific prediction would provide valuable information as to the precise form of the bias function.

It is an open problem as to how the inhibitory weights in these networks may be learned. While in this description the inhibitory weights are dependent on the most negative weight in the initial network, in reality there would be no initial network. It may be possible to pose an optimization problem related to Figure 6, where the inhibition linearizes the response at

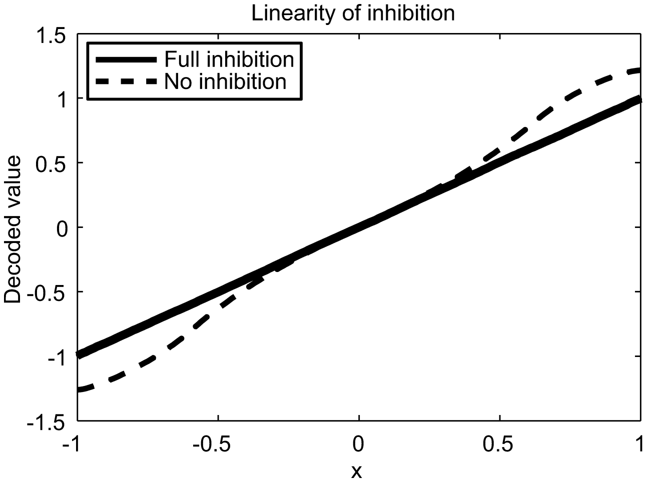

Figure 6: The effects of removing inhibition in a feedforward network. The x-axis is the input to the network, and the y-axis is the decoded output for that input. As shown, removing inhibition from the modified network retains good representation at low magnitudes, suggesting that the inhibition primarily linearizes the response at higher magnitudes.

high magnitudes. Such an optimization must be solved by a plausible local learning rule. We leave exploration of such rules as an important future challenge.

6 Conclusion

In general, this method can be taken to show that biologically implausible models that assume mixed weights do not in principle perform functions that cannot be performed by the brain. However, it is clear that direct comparisons between mixed weight models and the mechanisms found in biological systems may be misleading. In particular, the magnitude and distribution of weights and the kinds of currents observed within cells will be different in the models before and after employing this transformation. As a result, we expect the method to be particularly relevant for more explicit comparisons of models to experimental results.

Acknowledgments

We are grateful to Bryan Tripp, Ray Singh, and the anonymous reviewers for their helpful discussions and comments. This work is supported by the National Science and Engineering Research Council of Canada (261453-05), the Canadian Foundation for Innovation (3358401), and the Ontario Innovation Trust (3358501).

References

Aamodt, S. (Ed). (2000). Computational approaches to brain function [Special issue]. *Nature Neurosci.*, *3*(11s).

Azouz, R., Gray, C. M., Nowak, L. G., & McCormick, D. A. (1997). Physiological properties of inhibitory interneurons in cat striate cortex. *Cereb. Cortex*, *7*, 534–545.

Bartos, M., Vida, I., Frotscher, M., Geiger, J. R. P., & Jonas, P. (2001). Rapid signaling at inhibitory synapses in a dentate gyrus interneuron network. *J. Neurosci.*, *21*(8), 2687–2698.

Bartos, M., Vida, I., Frotscher, M., Meyer, A., Monyer, H., Geiger, J. R. P., et al. (2002). Fast synaptic inhibition promotes synchronized gamma oscillations in hippocampal interneuron networks. *Proc. Natl. Acad. Sci. U.S.A.*, *99*(20), 13222–13227.

Burnstock, G. (2004). Cotransmission. *Curr. Opin. Pharmacol.*, *4*, 47–52.

Buzsáki, G. (1984). Feed-forward inhibition in the hippocampal formation. *Prog. Neurobiol.*, *22*, 131–153.

Carter, A. G., & Regehr, W. G. (2002). Quantal events shape cerebellar interneuron firing. *Nature Neurosci.*, *5*, 1309–1318.

Churchland, P. M. (1995). *The engine of reason, the seat of the soul: A philosophical journey into the brain*. Cambridge, MA: MIT Press.

Conklin, J., & Eliasmith, C. (2005). A controlled attractor network model of path integration in the rat. *J. Comput. Neurosci.*, *18*(2), 183–203.

Deneve, S., & Pouget, A. (2003). Basis functions for object-centered representations. *Neuron*, *37*, 347–359.

Ekeberg, O., Lansner, A., & Grillner, S. (1995). The neural control of fish swimming studied through numerical simulations. *Adapt. Beh.*, *3*(4), 363–384.

Eliasmith, C., & Anderson, C. H. (2003). *Neural engineering: Computation, representation and dynamics in neurobiological systems*. Cambridge, MA: MIT Press.

Freund, T., & Buzsáki, G. (1996). Interneurons of the hippocampus. *Hippocampus*, *6*, 637–470.

Fukushima, K., Kaneko, C. R. S., & Fuchs, A. F. (1992). The neuronal substrate of integration in the oculomotor system. *Prog. Neurobiol.*, *39*, 609–639.

Gabbott, P. L., & Somogyi, P. (1986). Quantitative distribution of GABA-immunoreactive neurons in the visual cortex (area 17) of the cat. *Exp. Brain Res.*, *61*(2), 323–331.

Geiger, J. R. P., Lübke, J., Roth, A., Frotscher, M., & Jonas, P. (1997). Submillisecond AMPA receptor-mediated signaling at a principal neuron-interneuron synapse. *Neuron*, *18*, 1009–1023.

Hendry, S. H., & Jones, E. G. (1981). Sizes and distributions of intrinsic neurons incorporating tritiated GABA in monkey sensory-motor cortex. *J. Neurosci.*, *1*(4), 390–408.

Jonas, P., Major, G., & Sakmann, B. (1993). Quantal components of unitary EPSCs at the mossy fibre synapse on CA3 pyramidal cells of rat hippocampus. *J. Physiol. (London)*, *472*, 615–663.

Kamps, M. de, & Velde, F. van der. (2001). From artificial neural networks to spiking neuron populations and back again. *Neural Networks*, *14*, 941–953.

Kinney, G. A., Peterson, B. W., & Slater, N. T. (1994). The synaptic activation of IV-methyl-D-aspartate receptors in the rat medial vestibular nucleus. *J. Neurophysiol.*, *72*(4), 1588–1595.

Koch, C. (1999). *Biophysics of computation: Information processing in single neurons*. New York: Oxford University Press.

Kuo, P. D., & Eliasmith, C. (2005). Integrating behavioral and neural data in a model of zebrafish network interaction. *Biol. Cybern.*, *93*(3), 178–187.

Marty, A., & Llano, I. (2005). Excitatory effects of GABA in established brain networks. *Trends Neurosci.*, *28*(6), 284–289.

McAlpine, D., & Grothe, B. (2003). Sound localization and delay lines—Do mammals fit the model? *Trends Neurosci.*, *26*(7), 347–350.

Menschik, E. D., & Finkel, L. H. (1999). Cholinergic neuromodulation and Alzheimer's disease: From single cells to network simulations. *Prog. Brain Res.*, *121*, 19–45.

Moschovakis, A. K. (1997). The neural integrators of the mammalian saccadic system. *Front. Biosci.*, *2*, d552–577.

Pouget, A., Zhang, K., Deneve, S., & Latham, P. E. (1998). Statistically efficient estimation using population coding. *Neural Comput.*, *10*, 373–401.

Robinson, D. A. (1989). Integrating with neurons. *Annu. Rev. Neurosci.*, *12*, 33–45.

Salinas, E., & Abbott, L. F. (1994). Vector reconstruction from firing rates. *J. Comput. Neurosci.*, *1*, 89–107.

Selverston, A. I., & Miller, J. P. (1980). Mechanisms underlying pattern generation in lobster stomatogastric ganglion as determined by selective inactivation of identified neurons. I. Pyloric system. *J. Neurophysiol.*, *44*(6), 1102–1121.

Seung, H. S. (1996). How the brain keeps the eyes still. *Proc. Natl. Acad. Sci. U.S.A.*, *93*, 13339–13344.

Seung, H. S., Lee, D. D., Reis, B. Y., & Tank, D. W. (2000). Stability of the memory of eye position in a recurrent network of conductance-based model neurons. *Neuron*, *26*, 259–271.

Somogyi, P., Tamás, G., Lujan, R., & Buhl, E. H. (1998). Salient features of synaptic organisation in the cerebral cortex. *Brain Res. Rev.*, *26*, 113–135.

Strata, P., & Harvey, R. (1999). Dale's principle. *Brain Res. Bull.*, *50*(5/6), 349–350.

Walker, H. C., Lawrence, J. J., & McBain, C. J. (2002). Activation of kinetically distinct synaptic conductances on inhibitory interneurons by electrotonically overlapping afferents. *Neuron*, *35*, 161–171.

Xie, X., Hahnloser, R. H. R., & Seung, H. S. (1996). Double-ring network model of the head-direction system. *Phys. Rev. E*, *66*, 041902.

Zhang, K. (1996). Representation of spatial orientation by the intrinsic dynamics of the head-direction cell ensemble: A theory. *J. Neurosci.*, *16*(6), 2112–2126.

Zipser, D., & Andersen, R. A. (1988). A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons. *Nature*, *331*, 679–684.

Received July 30, 2006; accepted August 2, 2007.