Communicated by Richard Andersen

Integration and Differentiation in Dynamic Recurrent Neural Networks

Edwin E. Munro
Larry E. Shupe
Eberhard E. Fetz
Department of Physiology and Biophysics and
Regional Primate Research Center,
University of Washington, Seattle, WA 98195 USA

Dynamic neural networks with recurrent connections were trained by backpropagation to generate the differential or the leaky integral of a nonrepeating frequency-modulated sinusoidal signal. The trained networks performed these operations on arbitrary input waveforms. Reducing the network size by deleting ineffective hidden units and combining redundant units, and then retraining the network produced a minimal network that computed the same function and revealed the underlying computational algorithm. Networks could also be trained to compute simultaneously the differential and integral of the input on two outputs; the two operations were performed in distributed overlapping fashion, and the activations of the hidden units were dominated by the integral. Incorporating units with time constants into model networks generally enhanced their performance as integrators and interfered with their ability to differentiate.

1 Introduction

Dynamic neural networks, which incorporate time-varying activity and recurrent connections, can be trained to generate a variety of transforms between spatio-temporal input and output patterns (Fetz 1993). Biologically motivated examples include networks that simulate the reflex responses of the leech (Lockery and Sejnowski 1992), oscillatory activity of pattern generators (Tsung *et al.* 1990; Rowat and Selverston 1991; Williams and Zipser 1989), performance of a manual step-tracking task in primates (Fetz *et al.* 1990; Fetz and Shupe 1990), the vestibulo-ocular reflex (Anastasio 1991; Arnold and Robinson 1991; Lisberger and Sejnowski 1992a,b), and short-term memory (Zipser 1991). Dynamic recurrent networks can be trained to compute specific transforms from examples by using a modified form of the backpropagation algorithm (Watrous and Shastri 1986; Williams and Zipser 1989). To explore their ability to compute analytical functions, we trained recurrent networks to generate the

differential or the leaky integral of a continuously changing nonrepeating input; the resulting networks appropriately transformed any subsequent input pattern.

These networks were not intended to model any particular biological systems, although functions analogous to differentiation and integration do appear in physiological networks. Many neurons in the visual, auditory, and somatosensory systems respond preferentially to changes in peripheral stimulation (Patton *et al.* 1989), with transient responses resembling the differential of the stimulus. In the motor system, neural integrators have been postulated to transform transient commands into sustained activity, and to mediate the vestibulo-ocular reflex [as modeled by Anastasio (1991), Arnold and Robinson (1991), Cannon and Robinson (1985), Fuchs (1981), Lisberger and Sejnowski, (1992a,b), and Robinson (1989)]. The purpose of our networks is not to model specific biological circuits but to investigate the mechanisms of neural computation of integration and differentiation in networks of sigmoidal units. Biological neurons have history-dependent ionic and membrane properties that may help to generate these functions, but for simplicity we did not incorporate such mechanisms. However, we did examine the capacities of networks whose hidden units had intrinsic time constants. A further purpose of this study was to demonstrate that the underlying neural algorithm can be elucidated by reducing these networks to their minimal size (Mozer and Smolensky 1989; LeCun *et al.* 1990).

2 Methods

Our models utilized a recurrent network trained by a dynamic backpropagation algorithm (Watrous and Shastri 1986; Williams and Zipser 1989). A single input unit provided the time-varying input signal to the hidden units, and a bias unit provided a source of constant input. The hidden units had either excitatory or inhibitory connections, to each other and to the output(s). Self-connections and recurrent connections among the inhibitory units were sometimes omitted for efficiency, when their inclusion did not alter the basic results. The output layer consisted of one or two units with a linear input–output characteristic, representing the function(s) to be computed.

The total input to a given unit was the weighted sum of activity of its input units and bias, weighted by the respective connection strengths. The output of "sigmoidal" hidden units (appearing at the next timestep) was derived by the sigmoid squashing function, shifted along the abscissa by a constant = 4 (equivalent to a nonshifted sigmoid with a constant bias of -4). As described below we also investigated units with intrinsic time constants.

Each network was trained with a nonrepeating quasisinusoidal input signal whose frequency was randomly varied (periods between 2 and 60

timesteps). The desired target output waveform was calculated as the differential or the leaky integral delayed by two timesteps, to accommodate the delays between layers. Each training cycle consisted of an epoch spanning a fixed number of timesteps (normally 50). The difference between the actual and desired network outputs was summed over the entire epoch to calculate the net error, and the weights were changed to reduce this error. Both the duration of the training cycle and the number of previous timesteps included in the backpropagation could be adjusted to optimize the training. Varying the frequency of the training signal ensured that the network performed accurately over a broad range of frequencies.

3 Differentiation

Figure 1a illustrates a representative network with 16 hidden units that differentiates an arbitrary time-varying input. The units are identified by number and shown with their activation pattern; from top to bottom along the left-hand column, these are the bias, the input (i1), excitatory hidden units (a1–a8), inhibitory hidden units (b1–b8), and the output unit (o1). Each square of the matrix represents the connection weight from the unit in the left-hand column to the unit in the upper row. The area of the square represents the magnitude of the weight, and the shading designates excitatory (black) or inhibitory (gray) connections. [In some cases weights which exceeded the scale (at the top) are given numerically.] The activity of each unit in response to the illustrated input signal is also displayed along the left-hand column and along the top. The illustrated input signal is representative of the training waveforms. The activity of the output unit is the derivative of the input, shifted forward by two timesteps to accommodate propagation delays.

For this network, the nature of the solution can be deduced in part from the dominant connections in the weight matrix. Most of the excitatory units (a2–a7) receive strong input from the input unit and relay it directly to the output, with a net delay of two time steps. Many inhibitory units produce a time-shifted contribution to the output by receiving a delayed signal from the excitatory units and relaying this to the output for a total delay of three time steps (e.g., units b1–b4). The function of other inhibitory units (b6–b8) is less obvious, since they also receive input directly from i1 and provide negative feedback to the excitatory layer. In general the activations of the hidden units appear to resemble the input signal.

To elucidate the essential operation of differentiating networks we reduced them to their minimal form. In incremental stages, we removed inactive or ineffective units (e.g., a1 and b5) and combined units with similar activity and connectivity (e.g., a5 and a6), each time retraining the

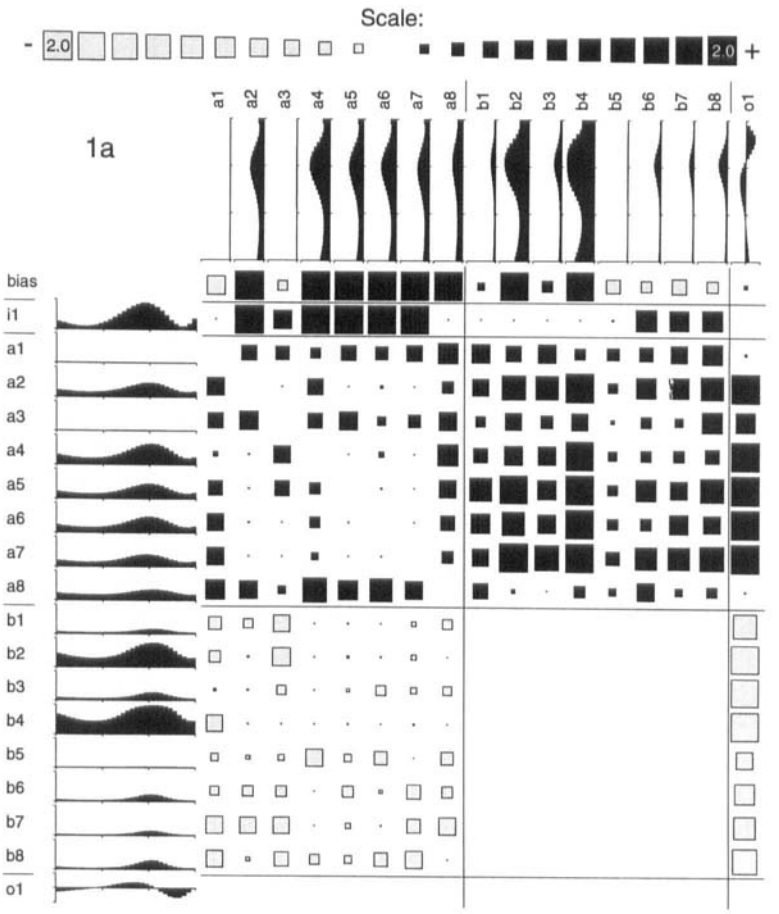

Figure 1: Differentiating networks. Each box represents the strength of the connection from the unit on the left to the unit at the top of the diagram. The scale boxes calibrate connection strengths for the networks below. The waveform displayed next to each unit's name shows that unit's activity over the time course of the input pattern. Unit i1 is the input unit, a1 through a8 are excitatory hidden units, b1 through b8 are inhibitory hidden units, and o1 is the output unit. (a) Network trained with 16 hidden units and a randomly varying sinusoidal input. *Continued facing page.*

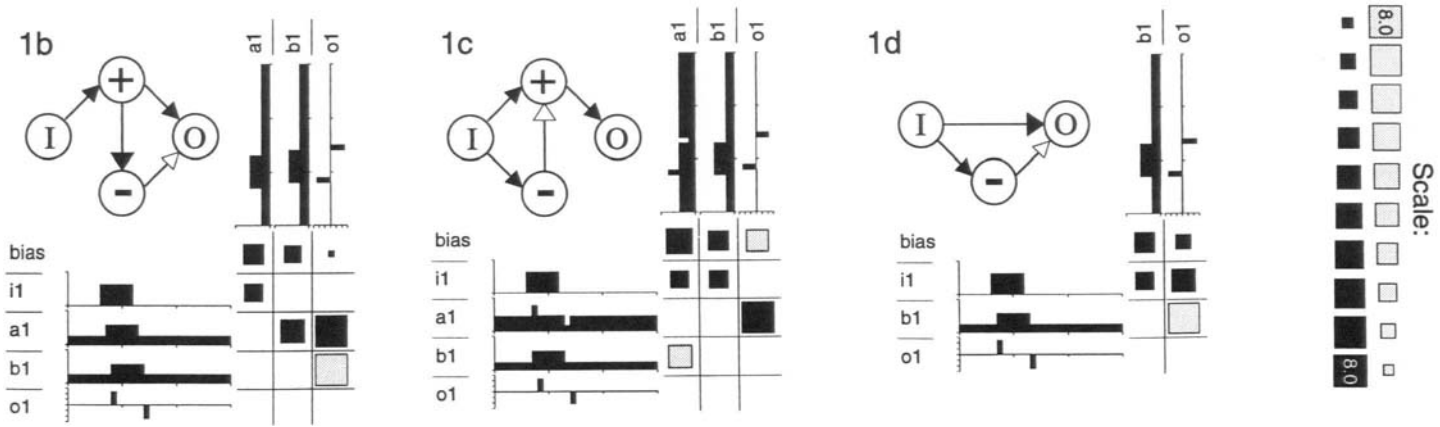

Figure 1: *continued.* (b,c) Reduced networks with minimal hidden units and no direct connection from the input to the output unit. The diagrams depict units and their connections. (d) Reduced network with a direct connection from the input to the output unit.

network. In some cases, this process was facilitated by implementing an automatic uniform weight decay during the training period. Differentiating networks could be reduced to one of two basic networks, each with a single excitatory and inhibitory hidden unit, as illustrated in Figure 1b and c, and shown with a pulse input. The circuit in Figure 1b relayed the input serially via the excitatory units to the inhibitory units, calculating the difference between them at the output. The circuit in Figure 1c fed the input simultaneously to the excitatory and inhibitory units, calculating the difference at the excitatory unit. The net result of both pathways is that the output represents the input signal minus the input delayed by one timestep, which produces the derivative: $o(t+2) = i(t)-i(t-1)$. These reduced networks are the smallest that produce an output mediated by the hidden layer and delayed by two time steps. An even smaller differentiating network can be constructed if the input is connected directly to the linear output; this network consists of the essential differentiating triad in Figure 1d. These networks have the familiar configurations of feedforward and collateral inhibitory circuits commonly found in biological sensory systems, which probably contribute to transient sensory responses.

The larger differentiating networks appear to implement the same time-shift algorithms, but in a distributed and intermingled form. Thus, network 1a implements primarily the first algorithm (1b), but also incorporates a component of the second. A large number of differentiating networks were obtained, using different starting weights and networks having more complete connectivity (including self-connections and recurrent inhibitory connections); all solutions implemented essentially the same time-shift subtraction algorithm, with variable combinations of these two basic circuits. The algorithm in Figure 1b was usually more prominent and more often survived weight reduction.

4 Integration

Recurrent networks of sigmoidal units could also be trained to simulate leaky integrators with various output decay constants. For the networks with longer decays it was necessary to scale down the output to prevent saturation with low frequency inputs.

Figure 2 illustrates a network trained to simulate a leaky integrator with an output decay constant of $\tau = 20$ timesteps, and exemplifies the mode of computation for integrating networks. The inhibitory units remained essentially unused, since they did not develop effective connections or activations. The excitatory units were strongly interconnected, and the activity of each resembled the output—viz., the leaky integral of the input. A pulse input generated rising and decaying activations whose decay constant τ (estimated by fitting the falling phase to an exponential) were found to be similar for the hidden units and were com-

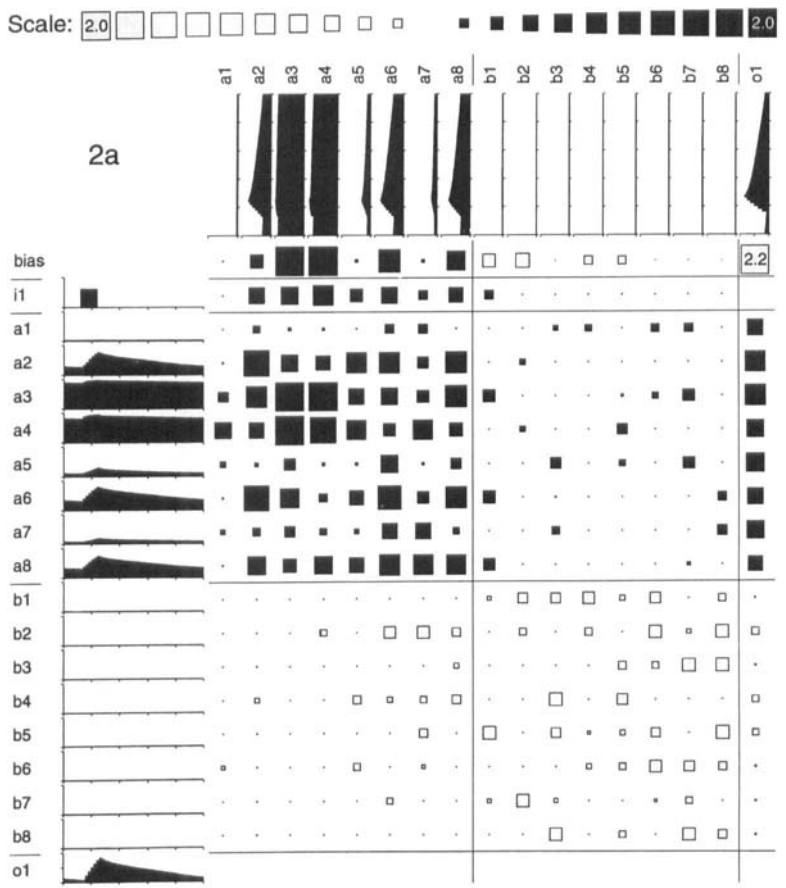

Figure 2: Integrating networks. (a) Full network with 16 hidden units and a pulse input. Decay constant $\tau = 20$ time steps. *Continued next page.*

parable to that of the output. Deleting individual hidden units resulted in a uniform reduction of these decay constants across the hidden and output units. These observations indicate that the recurrent excitatory connections within the hidden layer perform the integration. Again, solutions obtained from different starting weights differed in their detailed connections, but all involved recurrent excitatory connections.

The integrator networks could also be reduced to a smaller "essential" network containing two hidden units with reciprocal excitatory connec-

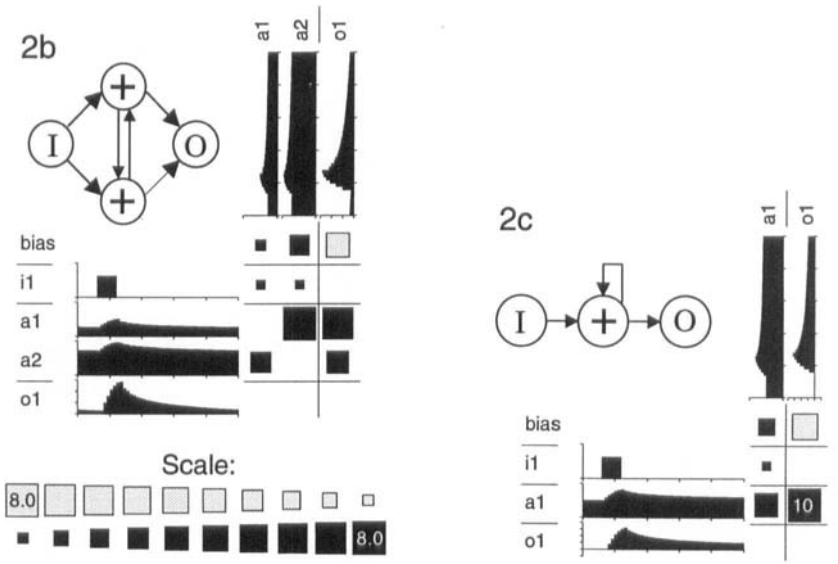

Figure 2: (b) Minimal network which excludes self-connected units ($\tau = 20$). (c) Minimal network with one self-connected hidden unit ($\tau = 100$).

tions if self-connections are excluded, as shown in Figure 2b. When self-connections are allowed, integration can be performed with a single excitatory hidden unit (Fig. 2c). Indeed, if the linear output unit is provided with self-excitation, no hidden unit is necessary and a pure integrator can be constructed by making the product of the self-connection weight and the slope of the linear activation function equal to one.

5 Simultaneous Differentiation and Integration

We also trained networks with sigmoidal units to produce both the differential and leaky integral as two simultaneous outputs for a single arbitrary input (Fig. 3). While networks quickly learned to compute either the derivative or the leaky integral, generally reaching an error level of $< 5\%$ within 1000–2000 training cycles, it proved more difficult to train networks to compute both functions simultaneously. Such networks generally learned to integrate with ease but failed to develop the inhibitory connections necessary for the differentiation, probably because the smaller differential signal contributed less to the error used for back-propagation. We overcame this problem by increasing the scale of the differential output (and therefore the differential error signal) relative to

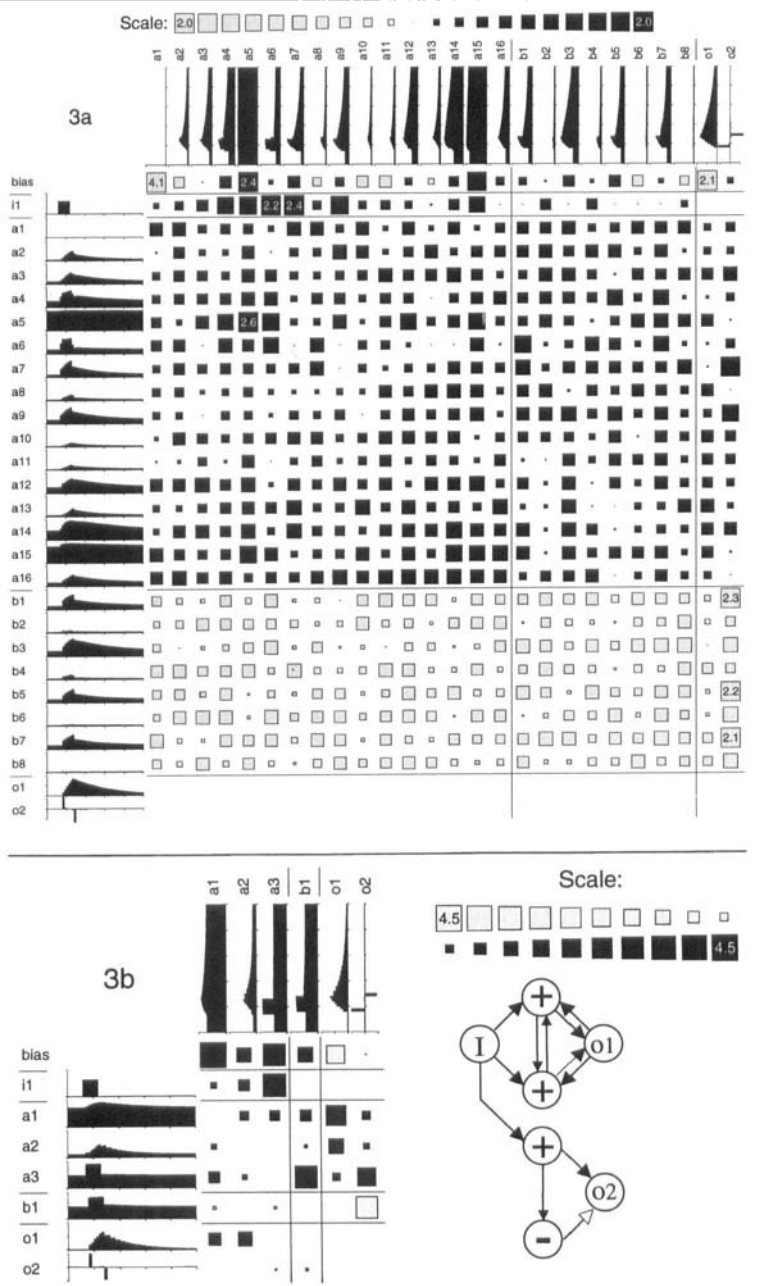

Figure 3: Networks that simultaneously differentiate and integrate the input. (a) Full network. (b) Reduced network.

the integral output until an acceptable solution was obtained. The number of hidden units and the overall number of training cycles required to reduce the error below the 5% level were larger for the networks performing both functions than for the single-function networks.

Figure 3a illustrates a full network, which integrates and differentiates the input. The dual-function networks perform this simultaneous computation in a distributed manner. In general, a larger number of hidden units were actively involved in the dual computation and many of these contributed significantly to both outputs, as seen by comparing the vertical columns of input weights to o1 and o2. The recurrent connections of the excitatory units are similar to those obtained for the simple integrator network. The activity patterns of most hidden units (both excitatory and inhibitory) carry a component that resembles the integral output and deletion of individual units produced a reduction of activation decay constants across the network, suggesting an integrating mechanism similar to the pure integrator. The computation of the differential within the dual-function network bears some resemblence to the pure differentiator as well. The activation patterns of many hidden units carry a component resembling the input. Within this class, excitatory units receive a strong direct input whereas inhibitory units receive little (cf. Fig. 1b). However, the extensive cross-connections among the hidden units and their dominant integral signal precludes a clean implementation of the same time-shift algorithm used by the pure differentiator. The algorithms found in the pure networks appear to be used in a distributed and overlapping fashion in the dual-function network.

Reduced networks could also be derived from dual-function networks. In general, the hidden units involved in computing the integral and differential tended to decouple as the networks became smaller. The overlapping nature of the computation could be preserved in these networks only at the expense of some residual error. The smallest possible reduced network contained four hidden units, three excitatory and one inhibitory (Fig. 3b). Units contributing largely to the integral (a1 and a2) are clearly distinguished by their activation patterns and connectivity from those contributing to the differential (a3 and b1).

6 Network Size

Since the small "reduced" networks can compute analytic functions as well as the large, it is interesting to consider the rationale for having networks with a larger number of units. Larger networks are essential for training, since smaller networks rarely converged on acceptable solutions. During training small networks usually became stuck in local error minima that are avoided by the larger degrees of freedom provided by additional connections. Second, for units with sigmoidal input–output functions that saturate at the upper and lower levels, additional units

are required to extend the linear range of the network operation. For example, in the multiunit integrator (Fig. 2a) the activations of the involved hidden units have a variety of baseline offsets. The sum of these activations provides a linear output over a wider range than could be achieved by fewer units. Third, the operation of the larger networks would obviously be less affected by the loss of any particular units, since many would be redundant. Fourth, it seems significant that larger networks can use the same units in several different elemental algorithms simultaneously. For example, in Figure 1a the inhibitory hidden units b6–b8 are involved in both of the differentiating algorithms in Figure 1b and c, and the hidden units in Figure 3a are involved in both differentiation and integration. Thus the larger networks can have significantly increased computational power, including the ability to compute different functions simultaneously.

7 Unit Time Constants

Prolonged responses of biological neurons are often modeled by incorporating intrinsic time constants into the hidden units. To see how our networks would perform when the hidden units retained some of their activity, we redefined the output value of unit i at a given time step $X_i(t)$ to be a weighted sum of its value at the previous timestep $X_i(t-1)$, and the output of the sigmoidal transfer function (F) defined in section two. Thus:

$$X_i(t) = \alpha X_i(t-1) + (1-\alpha)F[\text{weighted sum of inputs to } X_i]$$

where α varies between 0 and 1. With this definition, the intrinsic time constant of unit i in time steps is given by

$$\tau_i = 1/1 - \alpha$$

We chose values of α from 0 to 0.8, giving time constants in the range of 1–5 timesteps.

7.1 Integrator. As the time constants of hidden units was increased, the networks generally integrated with a lower level of error for a given number of training iterations. Moreover, networks incorporating larger time constants accomplished the same integration using smaller net recurrent weights (i.e., smaller sum of self- and feedback connections). The minimal leaky integrator with one self-recurrent unit required a lower recurrent weight when the time constant was larger, as expected from the equivalent effects of these two parameters. Similarly, in larger integrating networks the sum of recurrent weights was inversely related to intrinsic time constants.

To further test the frequency responses of networks we documented their output response to input sinusoids of different frequencies with the

Bode amplitude plot [the log of the output–input amplitude ratio vs. log of frequency] and Bode phase plot [the phase of the output relative to input vs. log of frequency]. The performance of leaky integrator networks was equally good in the training range, whether the hidden units had time constants or not; however, outside the training range the networks with intrinsic time constants had Bode plots that approximated the ideal leaky integrator slightly better.

7.2 Differentiator. Incorporating units with time constants into networks substantially impaired their ability to differentiate. Networks with the same architecture as those in Section 3, but with modest time constants (e.g., $\tau_i = 2$), could no longer be trained to differentiate from random initial conditions. In general, as the time constants of individual units increased, the networks took longer to learn to differentiate and performed at a greater absolute error level. This error was disproportionately due to the higher frequency component of the input signal. For networks with pure sigmoidal units without decay ($\alpha = 0$) the Bode amplitude plots approximated the ideal relation fairly closely, but fell off at a high frequency limit determined by the time step duration. For differentiating networks with units having time constants $\tau_i > 1$, the Bode amplitude plot reached a maximum at a lower frequency and then began to decrease. This inflection frequency was inversely related to the time constant of the hidden units.

We explored several methods to improve the training and performance of differentiator networks with time constants, including (1) increasing τ_i by small increments, and retraining between increases; (2) relaxing the sign constraint on hidden unit outputs; and (3) increasing the delay between the network's input and output signals from 2 time steps to 4. These methods produced additive improvements of network performance for a given time constant, as reflected by decreases in absolute error levels and increases in the high-frequency limit of Bode plots. Like differentiators composed of sigmoidal units without time constants, these networks appeared to generate a distributed version of the time shift algorithm described above, although they tended to use only a fraction of the units available. Titrating τ_i by small increments may improve learning performance by allowing the learning network to "track" a local error minimum as it varies parametrically with the time constant. Relaxing the sign constraint on hidden unit outputs allows more degrees of freedom for gradient descent, and gives networks a greater range for solutions. [Of course a network with unsigned units can be transformed to one with twice as many signed units.] The greater delay between input and output signals may provide two computational advantages: The larger delay could increase the number of paths through which a time shift and subtraction could be implemented [e.g., by varying the size of the time shift, the length of the path from input to output, and the time step at which the subtraction occurs]. It also makes it possible to produce

multiple estimates of the derivative in less than 4 time steps, allowing additional time to combine these estimates. We have not explored these possibilities systematically.

7.3 Combined Differentiator–Integrator. Networks that were previously trained to simultaneously differentiate and integrate an input signal as described above could be further trained (by successively incrementing τ_i and retraining) to both differentiate and integrate at reasonably low error levels with τ_i as high as 4.0. These dual-function networks differentiated more accurately than the pure differentiator networks obtained as described above, particularly at higher frequencies. For the dual-function networks without time constants the Bode amplitude plots for the differential output approximated the ideal as well as plots for pure differentiators without time constants. Increasing τ_i up to 4.0 for dual-function networks reduced the linearity and the high-frequency cutoff of the Bode plot only slightly. For $\tau > 4.0$ the high-frequency cutoff dropped markedly and the ability to differentiate high frequency signals deteriorated. The greater ease with which dual-function networks could accommodate increases in time constants compared to pure differentiators may reflect the fact that hidden units already had effective time constants imposed by their simultaneous participation in the integration. The combined differentiator/integrator networks tend to use most or all of their units in the computation whereas pure differentiators with time constants used as few as 10–20% of their units actively. Training networks to both integrate and differentiate may force them to distribute their computation of the derivative, which may allow the dual-function networks to deal better with the introduction of time constants.

8 Conclusions

These simulations have provided several insights into the capacity of recurrent dynamic neural networks to compute the analytic functions of differentiation and integration.

1. Using the backpropagation algorithm with nonrepeating input patterns produced neural networks that differentiated and/or integrated any subsequent test signal. By eliminating unnecessary units and combining redundant units it was possible to derive reduced networks that performed the same function. The reduced networks often revealed the computational algorithm more clearly. However, networks this small could not be trained from random starting weights.
2. The differentiating networks implemented a strategy of delayed subtraction of the input, and the hidden units often carried signals resembling the input. The integrator networks used recurrent

connections between the excitatory hidden units, whose activation all resembled the output (i.e., the integral of the input).

3. Networks could also be trained to produce the integral and differential simultaneously on two separate outputs. These networks combined the strategies used by the pure differentiator and integrator in a distributed and overlapping manner. Reduced networks performed these two operations best by dissociating into separate subnetworks for each function.

4. Integrating networks incorporating units with intrinsic time constants τ_i learned faster and performed better as τ_i were increased. By contrast, performance of differentiating networks deteriorated as τ_i were increased, particularly at high frequencies. Combined differentiator–integrator networks that had been trained without time constants could be trained to maintain their performance with incremental increases of τ_i over a limited range.

5. These circuits have some analogues in biological systems. Many neurons in sensory systems exhibit transient responses to stimuli (resembling differentiation) that may be sharpened by ubiquitous recurrent and feedforward inhibitory connections. The intrinsic membrane time constants of neurons (ca. 5 msec) would not compromise differentiation of signals changing at lower rates. In addition to network mechanisms, intrinsic receptor mechanisms can also generate transient sensory responses. Cells in motor systems, notably the oculomotor system, exhibit sustained responses to transient input, resembling integration; the network time constants are many times longer than intrinsic neural time constants, probably due to recurrent connections. In contrast to the recurrent excitation of our minimal integrator networks, the oculomotor system also uses inhibition in order to integrate push–pull signals without integrating baseline activity (Cannon and Robinson 1985; Anastasio 1991; Arnold and Robinson 1991).

Acknowledgments

This work was supported by the Office of Naval Research (contract number N00018-89-J-1240), and NIH Grants RR00166 and NS12542. E.E.M. was supported in part by the Graduate Program in Neurobiology at the University of Washington.

References

Anastasio, T. J. 1991. Neural network models of velocity storage in the horizontal vestibulo-ocular reflex. *Biol. Cybern.* **64**, 187–196.

Arnold, D. B., and Robinson, D. A. 1991. A learning network of the neural integrator of the oculomotor system. *Biol. Cybern.* **64**, 447–454.

Cannon, S. C., and Robinson, D. A. 1985. An improved neural-network model for the neural integrator of the oculomotor system: more realistic neuron behavior. *Biol. Cybern.* **53**, 93–108.

Fetz, E. E. 1993. Dynamic recurrent neural network models of sensorimotor behavior. In *The Neurobiology of Neural Networks*, D. Gardner, ed., pp. 165–190. MIT Press, Cambridge, MA.

Fetz, E. E., and Shupe, L. E. 1990. Neural network models of the primate motor system. In *Advanced Neural Computers*, R. Eckmiller, ed., pp. 43–50. Elsevier North Holland, Amsterdam.

Fetz, E. E., Shupe, L. E., and Murthy, V. 1990. Neural networks controlling wrist movements. *Proc. IJCNN-90* **II**, 675–679.

Fuchs, A. F. 1981. Eye-head coordination. In *Handbook of Behavioral Neurobiology, Vol. 5: Motor Coordination*, A. L. Towe and E. S. Luschei, eds., pp. 303–366. Plenum, NY.

LeCun, Y., Denker, J., Solla, S., Howard, R. E., and Jackel, L. D. 1990. Optimal brain damage. In *Advances in Neural Information Processing Systems 2*, D. S. Touretzky, ed., pp. 598–605. Morgan Kaufmann, San Mateo, CA.

Lisberger, S. G., and and Sejnowski, T. J. 1992a. Computational analysis suggests a new hypothesis for motor learning in the vestibulo-ocular reflex. Tech. Rep. INC-9201, Institute for Neural Computation, U. C. San Diego.

Lisberger, S. G., and and Sejnowski, T. J. 1992b. A novel mechanism of motor learning in a recurrent network model based on the vestibulo-ocular reflex. *Nature (London)* **360**, 159–161.

Lockery, S. R., and Sejnowski, T. J. 1992. Distributed processing of sensory information in the leech: A dynamical neural network model of the local bending reflex. *J. Neurosci.* **12**, 3877–3895.

Mozer, M. C., and Smolensky, P. 1989. Using relevance to reduce network size automatically. *Connection Sci.* **1**, 3–16.

Patton, H. D., Fuchs, A. F., Hille, B., Scher, A. M., and Steiner, R. (eds.) 1989. *Textbook of Physiology*. W. B. Saunders, Philadelphia, PA.

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

Rowat, P. F., and Selverston, A. I. 1991. Learning algorithms for oscillatory networks with gap junctions and membrane currents. *Network* **2**, 17–41.

Tsung, F.-S., Cottrell, G. W., and Selverston, A. I. 1990. Experiments on learning stable network oscillations. *Proc. IJCNN-90* **I**, 169–174.

Watrous, R. L., and Shastri, L. 1986. Learning phonetic features using connectionist networks: An experiment in speech recognition. Tech. Rep. MS-CIS-86-78. Linc Lab 44, University of Pennsylvania.

Williams, R. J., and Zipser, D. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Comp.* **1**, 270–280.

Zipser, D. 1991. Recurrent network model of the neural mechanism of short-term active memory. *Neural Comp.* **3**, 179–193.

Received June 18, 1992; accepted July 15, 1993.