From: Eliasmith and Anderson, Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems. MIT Press, 2004.

Chapter 4

Temporal representation in spiking neurons

Few neurobiologists will heed any characterization of neural representation if it depends on the assumption that neurons output real-valued firing rates. This is because it might come as no surprise that real-valued output from neurons can be used to represent signals changing constantly in real time. But neurons in real nervous systems traffic in neural spikes. As a result, what we need to know is how these highly discontinuous, nonlinear outputs can be used to successfully represent continuous temporal signals. In other words, neural spikes are fundamental features of neurobiological systems that present an unavoidable challenge for understanding neural representation in its 'full-fledged' form; i.e., as the encoding and decoding of time-varying signals in populations of spiking neurons.

In this chapter, our goal is precisely to understand the representation of time-varying signals by spiking neurons. In order to reach this goal, we begin by introducing a spiking version of the leaky integrate-and-fire (LIF) model. We use this model extensively in our subsequent simulations and thus begin by discussing its strengths and weaknesses. Eventually, we demonstrate that our framework does not, in any way, depend on this particular neural model. Neverthelesss, it is a useful place to begin our consideration of temporal encoding in neural systems.

Again, because we are interested in characterizing representation, we are confronted with the challenge of developing a method for decoding the results of this kind of encoding as well. In many ways, the approach we develop is analogous to that in the previous chapters; we again rely on the mean square error to find optimal linear decoders. There are some important differences as well, but the similarities hint at the fact that the first principle outlined in chapter 1 applies to both population and temporal coding.

4.1 The leaky integrate-and-fire (LIF) neuron

4.1.1 Introduction

The properties of the leaky integrate-and-fire (LIF) model neuron have been investigated since 1907, even before much was known about actual spike generation in neural systems (Arbib 1995; Koch 1999, p. 335). Since that time, the LIF model has become a standard approximation to the complex behavior exhibited by real, spiking neurons. It is widely used for a number of reasons. First, it is a very simple model that affords a good approximation to the behavior of many kinds of real neurons under a wide range of conditions. Second, it has been shown to be a limiting case of more complex conductance models such as the well-known Hodgkin-Huxley model (Partridge 1966). Third, though simple, the LIF neuron is far more realistic than rate neurons because it introduces the most prominent nonlinearity in neural systems—the neural action potential, or spike.

It is this third reason that is most important for those concerned with understanding neural representation. As we show, being able to understand the neural code as modeled by the spiking behavior of the LIF neuron goes a long way to understanding representation in far more detailed neural models (see section 4.5). Thus, we could have based our discussion of representation on one of the more complex models found in section 4.5, but, as that section shows, little theoretical insight would be gained. The LIF neuron, then, is a convenient and fruitful mixture of realism and simplicity (Koch 1999, see especially chp. 14).

In its standard form, the LIF neuron has two behavioral regimes: sub-threshold and super-threshold. As shown in figure 4.1, the sub-threshold behavior during a constant soma input consists of an ever-slowing approach towards the input voltage. Once the threshold is reached, super-threshold behavior begins and an infinitely thin and infinitely tall delta-function voltage spike ($\delta(t-t_n)$) with unit area (i.e., $\int_{-\infty}^{\infty} \delta(t-t_n)\, dt = 1$) is produced. The system is then reset to zero for some time period, τ^{ref}, before it is allowed to again begin its approach to threshold.

There are a number of physiologically motivated features of this model. First, the spike which is produced is both narrow and stereotypical as is seen in many real neurons. Of course, representing the spike as a delta function is a mathematical convenience. However, as long as the width of the neural spike in the neuron being modeled is small compared to the typical interspike interval, using a delta function approximation is warranted. Typically, neural spikes are about 1–2 ms in width compared to interspike intervals between 50 and 200 ms (i.e., 50–200 Hz), so the delta-function approximation is a good one. Second, the absolute refractory period, τ^{ref}, that forces the LIF model voltage to zero for a short duration after a spike is emitted is an excellent approximation to a similar phenomena found in real neurons (Koch 1999). Third, the sub-threshold leaky integration of the model is produced by a simple passive resistance-capacitance circuit whose elements have physiological correlates. We introduce and discuss this circuit in more detail in the next section.

As we note in sections 4.1.3 and 4.5, there are a number of shortcomings of the LIF model as well. But the realism, or lack thereof, of LIF neurons is itself of little importance to us because we are only interested in the model to the extent that it allows

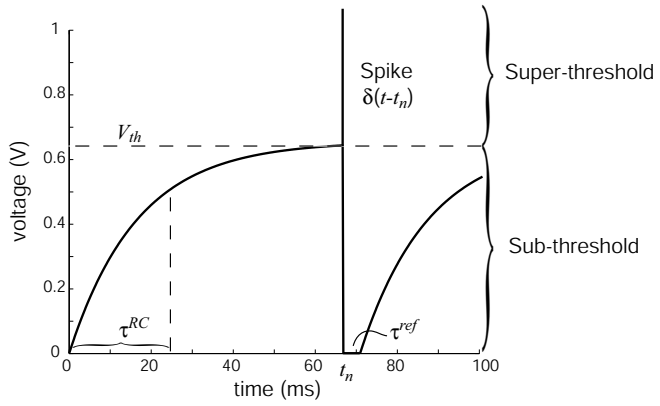

Figure 4.1: Leaky integrate-and-fire (LIF) neuron with constant current input. When the sub-threshold soma voltage reaches the threshold voltage, V_{th}, the model emits a delta function spike, $\delta(t - t_n)$. The model is then reset for a time, τ^{ref}, before again integrating its input. As discussed in section 4.1.2, the sub-threshold regime is well-characterized by an RC circuit, parameterized by τ^{RC} .

us to construct a useful formalism for understanding neural representation. Showing that the formalism we construct using LIF neurons is generally applicable is the burden of sections 4.3–4.5.

4.1.2 Characterizing the LIF neuron

The LIF neuron is best understood as a passive RC circuit coupled to an active spike (i.e., delta function) generator (see figure 4.2). This circuit provides a simple means of describing the time course of the membrane potential, V. The membrane potential is the result of the interaction between charges, Q, in the fluids inside and outside the cell. These charges are separated by the cell membrane, which acts as a dialectric layer. The capacitor, C, in the LIF circuit accounts for the charge build-up on either side of the bilipid layer that comprises the membrane. Notably, there is never any movement of charge across the lipid membrane itself (Jack et al. 1975, p. 13). Rather, the capacitive current J_C results from a change in the amount of charge separated by the membrane. Given that the voltage across the capacitor is $V = Q/C$, we can find J_C by differentiating:

$$J_C = C\frac{dV}{dt}. \tag{4.1}$$

In addition to the capacitance of the bilipid layer, the LIF model includes a leak resistance, R, that models the effects of (some of) the proteins embedded in the lipid membrane. These particular proteins act as ion channels, allowing sodium, potassium,

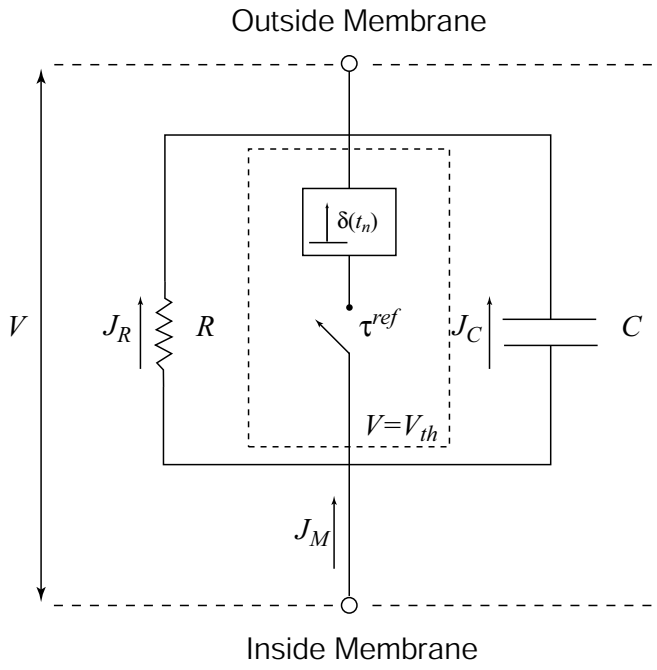

Figure 4.2: An RC circuit that implements the LIF neuron. The standard RC circuit, describing the sub-threshold behavior of the neuron is that part outside of the dashed box. It consists of the membrane potential, V, the leak resistance, R, and the capacitance due to the bilipid layer, C. The active, super-threshold behavior is described by the additional components inside the dashed box. When the membrane potential is equal to the voltage threshold, V_{th}, at a time, t_n, the short-circuit switch is closed, resulting in a 'spike', $\delta(t_n)$. The switch remains closed, resetting the circuit and holding $V = 0$, until it is opened after the refractory period, τ^{ref}.

and chloride ions to pass through the otherwise impervious membrane. The concentration gradients across the neuron membrane of the various ions results in the movement of ions into and out of the cell. The ionic current, J_R, accounts for this passive 'leak' of charge across the membrane.[1] Ohm's law tells us that the ionic current is

$$J_R = \frac{V}{R}. \tag{4.2}$$

The final current of importance to the LIF model is the membrane current, J_M. In effect, this current represents the input to the model. It can be thought of as the

[1]Because we are assuming that the circuit is passive, R is constant and we can assume that linear cable theory is applicable (see Jack et al. 1975, chapters 8, 9, and 10 for a discussion of nonlinear cable theory applied to cellular models).

somatic current resulting from all of the postsynaptic currents (PSCs) generated at the dendrites. As mentioned previously, we consider this current to be comprised of two distinct components, the bias or background current, J^{bias}, and the drive current, J^d. The bias current is an ever-present, steady state current, whereas the drive current accounts for the rapid current fluctuations due to dendritic inputs.

Since the flow of charge must be conserved between the inside and outside of the membrane (i.e., Kirchoff's law applies), we know that

$$J_M = J_C + J_R. \tag{4.3}$$

Substituting equations (4.1) and (4.2) into (4.3) and rearranging gives

$$\begin{aligned} J_M &= C\frac{dV}{dt} + \frac{V}{R} \\ \frac{dV}{dt} &= -\frac{1}{\tau^{RC}}\left(V - J_M R\right), \end{aligned} \tag{4.4}$$

where $\tau^{RC} = RC$.

Recall that this ordinary, first-order, partial differential equation only describes the passive behavior of the LIF model. In figure 4.2, the distinction between the active and passive components of the circuit is denoted by a dashed box. Once the membrane potential, V, crosses the LIF neurons threshold, V_{th}, the components in this box control the model's behavior. In particular, the gate denoted by τ^{ref} closes and a delta function, $\delta(t_n)$, spike is generated. By short-circuiting the capacitor and resistor, the gate sets the potential across the membrane to zero (i.e., the assumed resting potential) since there is no way for a difference in charge to build up. This gate stays closed for a length of time, τ^{ref}, equal to the absolute refractory period of the neuron being modeled.

Solving equation (4.4) for V is examined in some detail in appendix B.2. The result, which can be verified by differentiating, is

$$V(t) = J_M R\left(1 - e^{-t/\tau^{RC}}\right). \tag{4.5}$$

From this equation, we can determine the effects of past input on the model's behavior. In effect, we can begin to understand when the model's 'memory' is important, and when it is not. To do this, let us consider changes to the membrane time constant, τ^{RC} (i.e., changes to either the resistance or capacitance). First, we can see that under a constant input current, J_M, and after a length of time equal to τ^{RC}, $V(t)$ will be equal to approximately two-thirds (i.e., $1 - e^{-1}$) of the steady state input, $J_M R$. So, a larger time constant means both that it takes longer to get up to threshold for above-threshold inputs, and that a slower 'forgetting' is occurring when there is no input. Notably, τ^{RC} becomes much less important if the input current is extremely high. In this case, $V(t)$ becomes nearly linear between the resting voltage and the threshold voltage for most values of τ^{RC}. Given the fact that the circuit is reset after the threshold is passed, high input current results in very little effect of past states on current behavior. Thus, there is an interplay between τ^{RC} and the magnitude of the input current that determines the relevance of past input to the LIF neuron behavior.

In real neural systems, of course, the input current, J_M, is not static, but time varying. As we discuss in appendix B.2, the model's behavior under these conditions

can be understood as a convolution between the input signal and the exponential leak. Under that characterization, the voltage right now (at t) depends on all past current input, $J_M(t)$, where the input is weighted by a function that exponentially decays as that input gets further away (in the past) from the present time. In other words, the present voltage depends most strongly on the most recent inputs, and exponentially 'forgets' about past inputs as they get farther away. It is, in fact, this latter description that most accurately captures the nonlinearity inherent in the LIF model (and which we use in our simulations). However, equation (4.5) is easier to handle analytically because J_M is assumed to be relatively static.

In particular, we can use (4.5) to derive an expression for the neuron response rate curve that we used extensively in chapter 2. We know that the steady-state firing rate of an LIF neuron is inversely proportional to the length of time it takes the RC circuit to pass V_{th}. Note that because we are here concerned with steady-state firing (i.e., constant input current), this derivation is based on the assumption that the input current is changing slowly compared to the interspike interval as assumed when finding (4.5). Under this assumption, the time to threshold is equal to the length of time, t_{th}, it takes the circuit described by equation (4.5) to pass V_{th} plus the absolute refractory period, τ^{ref}:

$$a(t_{th}) = \frac{1}{t_{th} + \tau^{ref}}. \tag{4.6}$$

From equation (4.5) we can find t_{th} as follows:

$$\begin{aligned} V_{th} &= J_M R \left(1 - e^{-t_{th}/\tau^{RC}}\right) \\ t_{th} &= -\tau^{RC} \ln\left(1 - \frac{V_{th}}{J_M R}\right). \end{aligned} \tag{4.7}$$

Substituting (4.7) into (4.6), we find that the firing rate as a function of the input current J_M is

$$a(J_M) = \frac{1}{\tau^{ref} - \tau^{RC} \ln\left(1 - \frac{V_{th}}{J_M R}\right)}. \tag{4.8}$$

Note that $V_{th} = J_{th}R$, so we can cancel the resistance terms. As well, if we know what the input current is as a function of some 'external' variable, x, we can re-write equation (4.8) in the form we first encountered it in section 2.1.2, i.e.,

$$a(x) = \frac{1}{\tau^{ref} - \tau^{RC} \ln\left(1 - \frac{J_{th}}{J_M(x)}\right)}, \tag{4.9}$$

where $J_M(x) = \alpha x + J^{bias}$. Recall from section 2.1.2 that α is both the gain and a unit conversion factor, and J^{bias} accounts for the steady background input to the cell. Figure 4.3 demonstrates the effects of changing various of the parameters in this equation on the shape of the neuron response function.

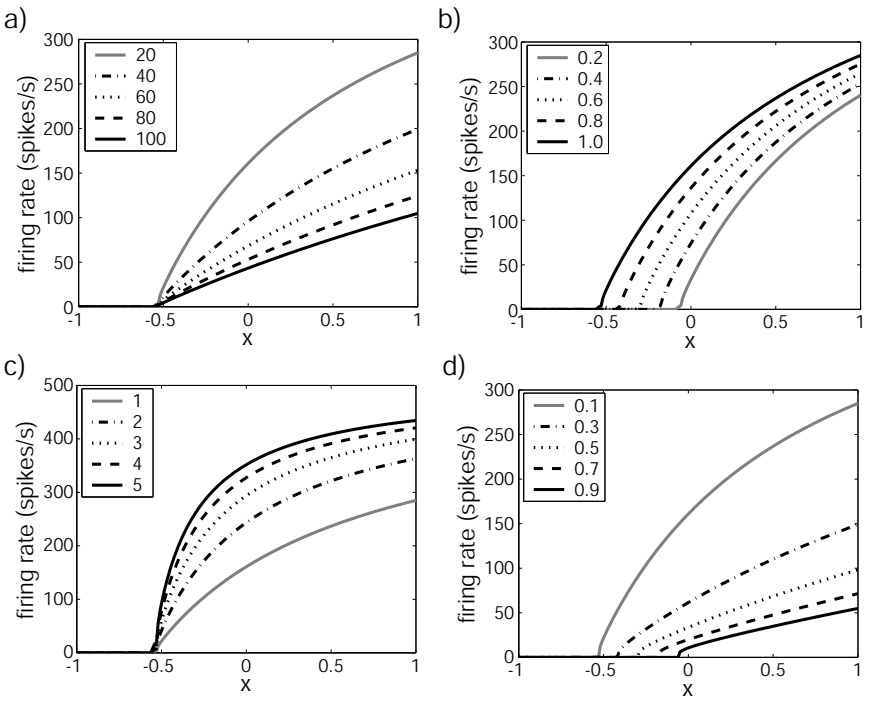

Figure 4.3: The effects of changing various parameters on the LIF response function. a) Varying τ^{RC} between 20 and 100 ms. b) Varying J^{bias} between 0.2 and 1.0 nA. c) Varying τ^{ref} between 1 and 5 ms. d) Varying J_{th} between 0.1 and 0.9 nA. Note that $\alpha = 17$, and unless explicitly changed, $J^{bias} = 10$, $J_{th} = 1$, $\tau^{RC} = 20$, $\tau^{ref} = 1$. Changing these parameters is partially redundant as demonstrated by comparing a), b), and d). Specifically, varying J_{th} is like varying both τ^{ref} and J^{bias}, thus in subsequent models we do not vary J_{th}.

4.1.3 Strengths and weaknesses of the LIF neuron model

We have now progressed from basic considerations of neuron physiology and simple circuits through a derivation of the spiking LIF model to arrive at a derivation of a rate LIF model. In order to motivate our focus on the LIF model, we have already discussed many of its strengths. To summarize, the LIF neuron model:

1. naturally incorporates a number of physiological parameters, including membrane capacitance, membrane (passive leak) resistance, and absolute refractory period;

2. is a good approximation over the normal operating range (i.e., low firing rates) of most neurons; and

3. introduces the 'important' nonlinearity of real neurons, i.e., the neural spike.

Essentially, we agree with Koch (1999) that "such single cell models represent the most reasonable trade off between simplicity and faithfulness to key neuronal attributes" (p. 335). In addition, it is important for the characterizations of representation we have introduced previously that there is an analytic expression for the rate curve of such a neuron. As can be seen from section 2.1.2, this approach depends on our being able to define $a_i(x)$ in order to minimize the error expressed by equation (2.4). Being able to write $a_i(x)$ explicitly as in equation (4.9) makes it easier to solve this error for large populations of neurons. In particular, it makes it possible to generate large populations of model neurons whose tuning curves can be easily manipulated.[2]

The weaknesses of the LIF model have been extensively discussed in the neuroscientific literature (Koch 1999; Softky and Koch 1995; Jack et al. 1975; Shamma 1989). Many of the concerns regarding the use of LIF neurons can be summarized by noting that LIF neurons are physiologically unrealistic to some degree. For example, LIF neurons have no spatial extent; i.e., they are 'point' neurons. A real neuron is extended in space, so the membrane potential at one location may be very different from the potential at another location. Furthermore, being a point neuron means that the electrochemical properties of the dendrites are completely neglected in the LIF neuron. And, as a final example, the complex time courses of membrane ion conductances are not accounted for. Contrary to what is assumed by equation (4.2), the membrane ionic currents tend to be nonlinear functions of both the membrane potential and time (Jack et al. 1975). All of these sorts of physiological facts are simply ignored by the LIF model.

Why do we think ignoring all of these important details will not hamper our attempts at understanding neural coding? The answer is simple: our approach does not depend on how spikes are generated. It only depends on the statistics of spike generation. Of course, including more physiological detail will make our model statistics more closely match the actual statistics of spike generation. For this reason, detailed models are eventually important (see Deco and Schurmann 1998); we discuss a number of them in section 4.5). However, our method for analyzing *some statistics or other* remains the same.[3]

It has been suggested by Softky and Koch (1993) that the problems with models like the LIF go much deeper. They claim that such models do not even have the right kinds of statistics. In particular, they show (using LIF neurons as an example) that models which assume linear summation of dendritic inputs are inconsistent with the highly irregular firing statistics found in most cortical neurons. Given the standardly assumed (i.e., Poisson) statistics of spikes impinging on a LIF neuron with linear dendrites, the output of that neuron will be *less* variable than its input. This is simply a consequence of the 'law of large numbers' (or 'Bernoulli's theorem') familiar to probability theorists. This law states that the variance of a random variable linearly comprised of other

[2]It is also possible to generalize the approach we take to neurons that have dynamic tuning curves (e.g., adapting neurons). We do not discuss this generalization in any detail, although see section 4.5 for an approach to understanding such neurons that does not depend on this generalization.

[3]Our approach is certainly not unique and the independence of this kind of approach from particular models has been noted before (de ruyter van Steveninck and Bialek 1988).

random variables, each with a variance σ^2, is equal to $\frac{\sigma^2}{n}$, where n is the number of random variables. Clearly, as n becomes large the variance of the resulting variable becomes small. This means that standard LIF models that sum dendritic PSCs will have a lower variability in their firing rate than their randomly firing input neurons. However, this is *not* a problem with the LIF neuron itself, but a problem with how the input current, J_M, is determined. Suffice it to say that there are a number of ways that J_M can be determined such that the output from a LIF neuron is sufficiently variable. So, there are ways to make LIF neurons have the relevant statistics, which means that the statistical analyses we perform can generalize properly.

4.2 Temporal codes in neurons

When discussing temporal representation (a.k.a. temporal coding) in neurons, it is difficult to avoid the vigorous debate between those who take the code to be a rate code (Shadlen and Newsome 1994; Shadlen and Newsome 1995; Buracas et al. 1998), and those who take it to be a timing code (de ruyter van Steveninck et al. 1997; Softky 1995; Rieke et al. 1997). In this section we show why this debate is, from our perspective, irrelevant to a good understanding of temporal representation in neurons.

Both rate codes and timing codes are clearly time dependent codes. A rate code, as proposed by Adrian (1928) is one that takes the information about a stimulus to reside in the mean firing rate of a spike train over a relatively long time window (about 100 ms). From this perspective, which still finds favor with some neuroscientists, the observed variability about this mean firing rate is considered to be noise (see, e.g., Shadlen and Newsome 1994; Shadlen and Newsome 1995). So defined, there are a wide variety of problems with adopting such a rate code as being a general feature of neural systems. First, most animals are embedded in highly dynamic environments and encounter signals that change very rapidly. If these animals needed to integrate information over an extended period (of even 100 ms), they would have little chance of survival. Indeed, there is plenty of evidence that many behavioral decisions are made on the basis of one or two neural spikes, which can arrive only a few milliseconds apart (see Rieke et al. 1997, pp. 55–63 for a good review). Second, there are limitations of rate coding that clearly do not affect some neural systems. For example, the frequency/amplitude ambiguity[4] can be resolved by a timing code, but not by a rate code (Bialek and Rieke 1992). Real neural systems seem to have no problem with this ambiguity. Third, there is experimental evidence that different input spike trains with the same mean rate, but different temporal structure produce significantly different output from the same cell (Segundo et al. 1963). Fourth, Zador (1998) has shown that the existence of a mean rate code contradicts the observed redundancy of synaptic connections. In particular, he has shown that the information transmission of such a code *falls* with redundant synaptic connections. However, such connections are common in neural systems. Fifth, and lastly, rate codes cannot support the information transmission rates observed in real neurons (Rieke et al. 1997), although it has long

[4]This ambiguity arises in auditory neurons because sounds with high amplitude that are not at the preferred frequency of a neuron can result in the same spike rate as sounds with low amplitude at the neuron's preferred frequency.

been known that a timing code can (MacKay and McCulloch 1952). In conclusion, there are many reasons to think that the neural code is not a mean rate code.

So, is the neural code obviously a timing code? Unfortunately not. For instance, there is evidence that the precise timing of spikes is not mandatory for the successful transmission of neural signals (Bialek et al. 1991).[5] More generally, whether or not the neural code is a timing code depends on what we mean by 'timing code'. The standard timing code is one that takes spike train *variability* to encode the stimulus signal (MacKay and McCulloch 1952; de ruyter van Steveninck and Bialek 1988; Softky 1995; Rieke et al. 1997). One way of expressing this kind of code is to take the inverse of the interspike intervals (i.e., 1/ISI) as a measure of the variations in a single trial. Because the same stimulus commonly elicits different spike trains, this kind of measure is often averaged over a number of trials. The averaged measure is sometimes called the 'instantaneous' rate code for the neuron (Buracas et al. 1998; Rieke et al. 1997). The term 'rate code' should not be too surprising here, as the measure is equivalent to a rate code where the window size approaches a limit of zero. However, the fact that the window size is so small has prompted many to consider this code a timing code.

There are other timing codes, as well. For example, Optican and Richmond (1987) suggest that information about spatial patterns can be found in an 'onset' timing code (see also Richmond and Optican 1990). They argue that the placement of spikes *relative to stimulus onset* carries information about the stimulus, and that this information can be about non-temporal features, such as shape (Optican and Richmond 1987). Although more recent results have contradicted this interpretation (Tovee et al. 1993), the ubiquity of adaptation in excitatory cortical neurons (i.e., the slowing of firing rates given sustained, super-threshold input) suggests that it may be the case that spike times relative to stimulus onset are important. The main difference between this kind of code and the instantaneous rate code is that non-temporal features are thought to be multiplexed into the time course of the neurons. So, traditional rate codes suffer a number of limitations, instantaneous rate codes do not seem to be rate codes, and onset timing codes are not empirically well supported.

This brief review gives some sense of the kinds of approaches to neural coding available, and why they are at odds. But, why did we say that choosing amongst these different codes is irrelevant? There are three reasons. The first is semantic. That is, there is little agreement as to whether instantaneous rate codes are rate codes (Buracas et al. 1998; Stevens and Zador 1995) or timing codes (Rieke et al. 1997; de ruyter van Steveninck and Bialek 1988). So, it would be unclear what we meant if we simply claimed that we were exploring a rate code or a timing code. The second reason is that it seems likely that the brain uses different codes for different problems (Zador 1998; Rieke et al. 1997); perhaps rapid sensory processing is more likely to use a timing code and static sensory processing is more likely to use a rate code (Buracas et al. 1998). This leads us to our third, and far more important reason for not 'choosing' a code: we simply don't have to. Given the statistical methodology that we adopt (section 4.3), the 'appropriate' code is determined by the signals that are represented and the properties of the neurons which represent those signals. In cases where the signals are rapidly varying relative to the interspike interval, something more like a timing

[5] Actually, for those who like timing codes, this is evidence of the 'robustness' of a timing code. For those who like rate codes, this is evidence of the lack of importance of precise timing.

code is appropriate. In cases where the dynamic range of the signals is large, but the correlation time is fairly long relative to the interspike interval, something more like a rate code is appropriate (see section 4.4). The method we discuss is effective for both of these cases. As well, in cases where the neurons adapt, thus changing the placement of spikes relative to stimulus onset, an appropriate code can be found again using the same method (see sections 4.5.1 and 4.5.3). In other words, we do not need to commit ourselves to one particular kind of coding as being central to neural representation. The approach we use here is general enough to characterize the kind of coding *appropriate to the problem at hand*. This understanding of neural coding thus transcends concerns about whether neurobiological systems use rate or timing codes.

4.3 Decoding neural spikes

4.3.1 Introduction

Recently, there has been a large amount of attention given to the information theoretic properties of neural spike trains (Bialek et al. 1991; Miller et al. 1991; Koch 1999; Stevens and Zador 1996; Richmond and Optican 1990; Roddey and Jacobs 1996; Bialek and Rieke 1992). The resulting attempts to decode neural spike trains have been largely successful, and provided many insights into neural coding. Given this success, there is no need to develop a completely new means of understanding neural representation. As a result, the methods we discuss in this section are a variation on past themes, to which we introduce only modest improvements. Thus, the importance of this section largely lies not in its novelty, but in the fact that it is an integrated part of a general, unified framework for modeling large-scale neural systems.

To begin, then, recall that a LIF neuron provides a good characterization of the temporal encoding process that is found in nervous systems (see figure 4.4). That is, a LIF neuron produces a series of stereotypical output spikes, $\delta(t - t_n)$, given some real-valued input signal, $x(t)$. To put this more precisely, and in the same form as encountered in chapter 2, we write

$$\begin{aligned} a(x(t)) &= G[J(x(t))] && (4.10)\\ &= \sum_n \delta(t - t_n), && (4.11) \end{aligned}$$

where

$$J(x(t)) = \alpha\tilde{\phi}x(t) + J^{bias}.$$

Here, the encoding function $G[\cdot]$ is defined by the parameters of the LIF neuron. As a result of these parameters, the model produces spikes (i.e., $\delta(t - t_n)$) at times t_n. In order to understand this output as a representation, we must find the relevant decoder.

In real nervous systems, it makes the most sense to think of peripheral neurons (e.g., retinal ganglion cells) as encoding some external, time-varying signal. This is because the encoded signals are relatively easy to pick out: they can be light intensities, velocities, pressure changes, or any other time-dependent physical magnitude. Such signals

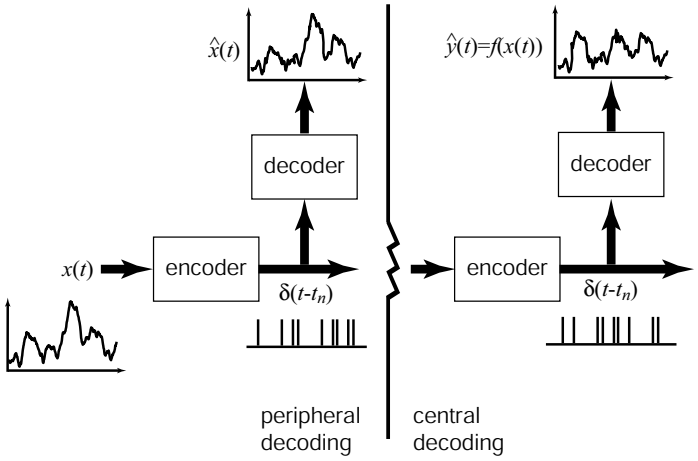

Figure 4.4: The temporal encoding and decoding process. Some physical signal, $x(t)$, such as light intensity is encoded by peripheral neurons into a series of spikes, $\delta(t - t_n)$. These can be passed to a (non-neural) decoder to give an estimate, $\hat{x}(t)$, of the original signal and thus help quantify the signal processing characteristics of the neural system. In more central neurons, the characterization is the same, but the signal being estimated, $y(t)$, will be some complex function of the input, i.e., $y(t) = f(x(t))$. Thus the decoder is a means of characterizing all of the encoding steps that precede that decoding. A long-standing problem in computational neuroscience is determining the relevant decoder (FitzHugh 1961).

are fairly directly encoded into a series of neural spikes, $\delta(t - t_n)$, in nervous systems. As neurons become farther removed from the periphery, it is often less clear what the signal is that is being encoded. Nevertheless, a 'central' neuron is clearly part of *some* encoding process. The fact that the neuron is only one of many elements that give rise to the encoding process simply means that when we attempt to decode that neuron's spike train, we should not associate the decoder with that particular neuron regardless of its relations to other neurons. Rather, such decoders must be associated with the entire encoding process that they complement (see figure 4.4). But, importantly, the problem of characterizing neural representation remains the same.

Despite the fact that a central neuron is only one part of some highly complex encoding process, we can still uniquely identify each neuron with one encoding process. As a result, we can assume that the decoder associated with a neuron is the decoder for the encoding process for which that neuron is the terminus. With this in mind, we will not be lead too far astray if we think of decoders as being associated with a particular neuron. However, we must remember that neurons do what they do (i.e., encode what they encode) because of their relations to other neurons, and not solely in virtue of their intrinsic properties.

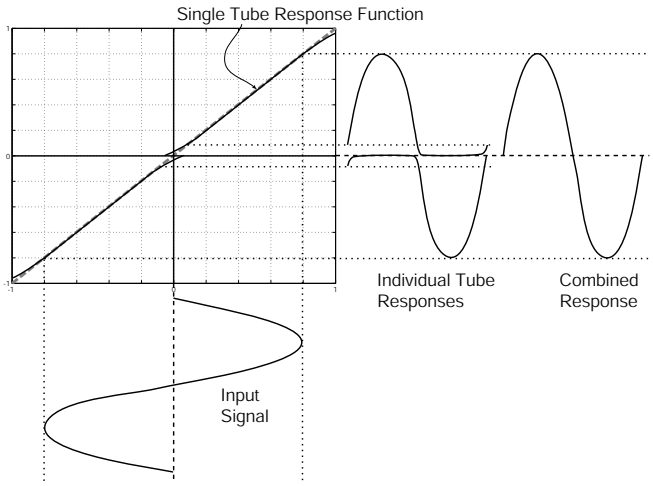

Figure 4.5: A push-pull amplifier. The input signal has nonlinear distortions introduced by each of the elements, but when the responses of the elements are summed those distortions cancel. The result is a highly linear reproduction of the original input signal over a wide range, despite nonlinear elements.

4.3.2 Neuron pairs

In order to successfully find the decoders in this picture of neural representation, it helps to make things a bit more complicated. That is, it helps to consider two neurons at a time instead of just one. This may seem strange, as the fundamental unit of signal processing in the nervous system is often thought to be the single neuron. However, given the considerations of the previous chapters, it is clear that we must understand representation in *populations* of neurons in order to understand the behavior of neural *systems*. The simplest possible population is a pair of neurons. So, while being simple, understanding temporal representation in pairs of neurons constitutes a significant first step towards understanding temporal representation in populations. And, importantly, some of the more troublesome properties of single neurons (e.g., their highly nonlinear responses) become far less troublesome when we consider neuron pairs.

This might come as no surprise to communications engineers. It has been known since the 1920s that superior amplifiers can be built by using complementary pairs of tubes or transistors. This kind of amplifier is commonly known as a push-pull amplifier (see figure 4.5). These amplifiers are significantly more efficient than single element amplifiers that achieve the same linearity (Sawdai and Pavlidis 1999). In general, the push-pull topology provides higher efficiency, higher power output, greater symmetry in the output signal, and a wider range of linear responses compared to a single element circuit.

For our purposes, the increase in linearity is of the most interest. The reason that the push-pull arrangement provides this benefit is that the nonlinearities in the elements

are used to offset one another near the midpoint of the range of operation (usually 0). In effect, each element is used to code half of the input signal. However, the halves overlap such that their sum in the overlapping region serves to linearize the total response, despite each element having significant nonlinearities in the same region (see figure 4.5). Thus, the total range over which the response of the system is reasonably linear is greater than could be achieved with two independent (i.e., averaged) elements.

So what does this have to do with neural coding? There is evidence that nervous systems often use a similar strategy. It is not unusual to find 'on' and 'off' cells in a population that codes a single physical quantity. Most obviously, 'on' and 'off' retinal ganglion and lateral geniculate nucleus cells are used to code light intensities (Hubel and Wiesel 1962). In the neural integrator circuit, the neurons can be divided into two groups; those whose response increases during leftward movements and those whose response increases during rightward movements (Fukushima et al. 1992). The same is true of velocity sensitive neurons in the head-direction system of mammals (Redish 1999) and angular acceleration sensitive cells in the vestibular system (Wilson and Jones 1979). In motor cortex, cells are broadly tuned and have a wide range of preferred directions (Georgopoulos et al. 1993). This is simply a generalization to higher dimensions of the same kind of strategy. That is, for most one-dimensional slices that go through zero in this two-dimensional space, we observe the same kinds of opponency cells we see in these other systems; that is, some increase firing for movement in one direction and some increase firing for movement in the opposite direction.[6] In many cases, then, cells have an opponency relation analogous to that of push-pull amplifiers. And we think the reason might be generally the same—this is a more effective means of constructing a system with a broader range of linear responses.[7] In appendix B.1 we explain why opponency provides more linear responses.

There is another reason that opponency is important. This second reason is unique to neurons. It is crucial for neurons to have this kind of arrangement because they do not encode their input with a *continuous* output signal. Amplifiers have continuous output. Neurons, in contrast, have an 'all-or-none' output in the form of a voltage spike. In order to understand the consequences of this difference, consider a simple code in a perfectly linear neuron: the more intense the stimulus, the higher the firing frequency, with zero intensity equal to zero firing frequency. Significant problems arise with this kind of code for low stimulus intensities. At low intensities, the neuron fires at low frequencies so it will take a *very long time* to determine what the frequency is. A mechanism decoding the output would have to wait until it sees enough spikes to guess what the firing frequency is. In other words, the lower the stimulus intensity, the longer it takes for the neuron to transmit information about stimulus intensity. This is not acceptable for a system attempting to operate in a rapidly changing environment.

However, this problem is solved with an opponency arrangement. By adding a second neuron that increases firing with decreasing stimulus intensity, we have guaranteed

[6] This is the result of the large number of neurons randomly distributed over the hypersphere in the relevant space. We do not mean to insinuate that neurons are actually *paired* (i.e., for each neuron there will always be a neuron tuned to the opposite direction).

[7] Neural systems are often quite linear over a broad range. For instance, in some vestibular systems there is only a 10% deviation from linearity over a 16 fold change in stimulus intensity (Wilson and Jones 1979, p. 99).

that the average firing rate (so the average time to transmit information about stimulus intensity) is constant. If we are not too concerned about having a constant average firing rate, so long as it is above a certain value, we can increase the slope of our encoding neurons. This gives us a better signal-to-noise ratio over the same range since a smaller change in stimulus intensity leads to a large change in the output firing rate. A similar argument holds in the case of nonlinear neurons like the LIF neuron.

In sum, this on-off opponency arrangement provides a temporally efficient and more linear encoding than is evident from examining single neurons. Since we are interested in constructing linear decoders for neural encodings, it is likely that the more linear response from a pair of neurons will serve our purposes better.[8] As well, since we are ultimately interested in understanding temporal codes in neural populations, it makes sense to start with one of the simplest populations—just two neurons. Given these considerations, we take it that neuron *pairs* are at least as fundamental for understanding the properties of neural representation as individual neurons.

4.3.3 Representing time dependent signals with spikes

Taking opponent neurons as the fundamental unit of signal processing in neural systems, figure 4.6 shows what we take to be the basic encoding and decoding steps that are needed to characterize neural representation. As mentioned earlier, the encoding process is determined by the neural model we have chosen (LIF in our case) and so we are largely concerned with the task of determining the appropriate decoder. This is much like the problem we originally faced in chapter 2, except that we are interested in understanding the representation of a continuously changing function of time, rather than a static scalar or vector.

As a result, we begin by making the same well-supported assumption about the linearity of the decoding as we did for the population code (see Rieke et al. 1997 for an excellent review, especially pp. 76–98 and 168–172; see also Bialek et al. 1991; Bialek and Zee 1990). In particular, we presume that we can decode the neural spike train by taking a (continuous, time-shifted) *sum* of some decoding function, $h(t - t')$, weighted by the encoded signal (see figure 4.7 for an intuitive picture of linear decoding). We know, given the LIF model, that the signal, $x(t)$, is encoded as a series of delta functions, which we can write as $\sum_n \delta(t - t_n)$. Thus, our decoding of the encoded function, $x(t)$, can be expressed as

$$\hat{x}(t) = \int_0^T h(t - t') \sum_n \delta(t' - t_n) dt'. \tag{4.12}$$

This kind of expression is known as a convolution integral. This equation says that our best guess as to what $x(t)$ is, can be found by adding up the occurrences of a time-shifted decoding function (analogous to the decoding weights, ϕ_i, in the population code). Another way of describing this decoding process is to say that we are assuming that there is some linear system that can decode the spike train, $\delta(t - t_n)$, and success-

[8] Notably, Rieke et al. (1997) also find optimal decoders by examining two neurons, although they do not make much of the point.

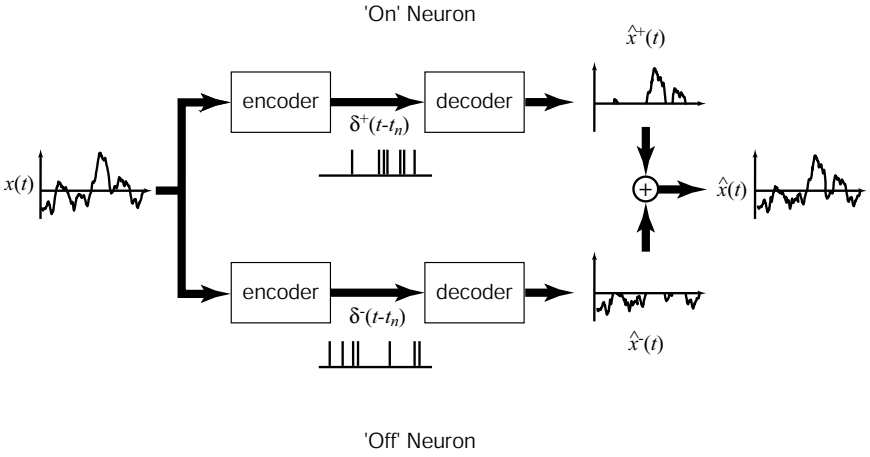

Figure 4.6: Encoding and decoding with an on-off neuron pair. Just like the push-pull amplifier, each neuron encodes part of the signal, and their decoded sum provides a good representation of the whole signal.

fully reconstruct the original input signal that was presented to the neuron. Our job, then, is to find this linear system.

Because a linear system can be completely characterized by its impulse response, $h(t)$, finding the linear system is the same as finding, $h(t)$. In particular, the output from any linear system, $y(t)$, can be written as a convolution integral of the input, $f(t)$, and the system's impulse response, $h(t)$:

$$y(t) = \int h(t-t')f(t')dt'.$$

This is clearly analogous to equation (4.12) above. So what we mean when we say that we assume a linear decoding, is that we can find an impulse response, $h(t)$, that reconstructs the original signal. Because the function $f(t)$ in equation (4.12) is a series of delta functions, we can simplify the integral by evaluating it analytically to give

$$\hat{x}(t) = \sum_n h(t-t_n). \tag{4.13}$$

Note that calling $h(t)$ a linear decoder (or linear filter) does not mean that we assume that we can find $h(t)$ by a linear analysis of the neuron's response. In fact, we have shown explicitly elsewhere that a linear analysis will not allow us to find this decoding filter (Anderson et al. 2000). In some ways, this should not be surprising given how nonlinear neurons really are.

Now that we have a characterization of the general nature of temporal representation, we need to consider if there are any special properties of the signals themselves. This is because any constraints on what is actually represented by nervous systems

can help us find the relevant decoders. To this end, it is important to emphasize that we do not want a linear decoder that will decode one particular signal. Rather, we want a linear decoder that is useful for decoding all signals of interest; i.e., we want to find a linear decoder that is good for an *ensemble* of signals, just as we earlier wanted decoders for an ensemble of functions. There is good evidence that real neurons are optimized for passing information about particular ensembles—namely, *natural* signals (de ruyter van Steveninck et al. 1997). It would be very surprising if biological relevance *had not* played a role in constraining the representational characteristics of neural systems, so this is just what we might expect.[9]

One general property of naturally occurring signals that we assume is that they are not unique in time. This property variously goes by the name of 'stationarity', 'ergodicity', or 'time independence'. By definition, if a signal (or process or system) is stationary, then the statistics of any sample of the signal does not depend on the temporal placement of that sample. So, for example, signals consisting of natural images are stationary because they have the same kinds of statistics *now* as they did *yesterday*, and as they will *tomorrow*. In general, sensory signals do not have clearly defined beginnings and endings, so they can be usefully treated as stationary.

Being able to ascribe the property of stationarity to natural signals is important because it allows us to define the relevant ensembles of signals using statistically independent coefficients. In particular, we can write the ensemble that we take to be represented as a Fourier series:[10]

$$x(t;\mathbf{A}) = \sum_{n=-(N-1)/2}^{(N-1)/2} A(\omega_n)e^{i\omega_n t}, \tag{4.14}$$

where $\mathbf{A}$ is the vector of frequency amplitudes and ω_n is $n\Delta_\omega$ where Δ_ω is the frequency step size. The series of frequency components, ω_n, consists of N complex frequency components where $A(\omega_n) = A^*(-\omega_n)$, and N is odd. It is important to note that we always assume a finite N. The reason we do this is because we are always interested in characterizing systems that operate in the real world, under tight implementational constraints. Under such conditions, there will always be some finite time, T, that is the maximum length of a signal in our ensemble. Given the duality between the length of time of a signal and its resolvability in the frequency domain, we know that $\Delta_\omega = 2\pi/T$. This guarantees that we can faithfully represent our signal with a finite number of frequency components.

As before, choosing a particular set of amplitude coefficients (i.e., a specific $\mathbf{A}$ vector) picks out one signal in this ensemble (see section 3.2). So, in order to properly define an *ensemble* of signals, we must specify what values the vector $\mathbf{A}$ can assume. Again, we can do this by specifying the probability distribution of $\mathbf{A}$. Assuming that each component of $\mathbf{A}$ is an independent Gaussian random variable whose variance is given by the power spectrum $\mathcal{P}(\omega_n)$, we can write a Gaussian distribution[11]

[9] This makes it rather unfortunate that many experiments in neurophysiology use very unnatural signals (e.g., step functions) to characterize neuron responses. Using signals drawn from a more natural ensemble often shows the neurons to be more adept at passing signals that was previously thought (Mainen and Sejnowksi 1995).

[10] This is equivalent to our definition in equation (3.3), but is in a more convenient form for the subsequent analysis.

[11] Notably, the experiments that we later compare our results to draw signals from signal ensembles with the same structure (Bialek et al. 1991).

$$\rho(A(\omega_n)) = \frac{1}{\sqrt{2\pi\mathcal{P}(\omega_n)}} e^{-A(\omega_n)^2/2\mathcal{P}(\omega_n)}. \tag{4.15}$$

Given stationarity, we know that these distributions define the distribution of all of the amplitudes, $\mathbf{A}$, namely,

$$\rho(\mathbf{A}) = \prod_{n=0}^{(N-1)/2} \rho(A(\omega_n)), \tag{4.16}$$

and $A(\omega_n) = A^*(-\omega_n)$.

Now that we have a means of describing the relevant signal ensembles, we can return to the problem of finding an optimal linear filter. Recall that we are interested in analyzing pairs of neurons, and take the system of interest to be comprised of two complimentary neurons, whose response profiles mirror one another (see figure 4.6). Appropriately extending equation (4.11), we can write the encoding of the *pair* of neurons of some signal (picked out by $\mathbf{A}$) into M spikes as

$$\begin{aligned} R(t;\mathbf{A}) &= \sum_{k}^{M_{on}} \delta(t - t_k^+(\mathbf{A})) - \sum_{l}^{M_{off}} \delta(t - t_l^-(\mathbf{A})) \\ &= \sum_{i=1}^{2}\sum_{k=1}^{M} \phi_i \delta(t - t_{ik}(\mathbf{A})), \end{aligned}$$

where $\phi_i = 1$ for the 'on' spikes and -1 for the 'off' spikes as before. We use $R(t;\mathbf{A})$ to indicate the *response* of the pair of neurons; that is, all of the spikes produced by both neurons given the input (this is *not* the same as the estimate of the signal in (4.13)). In order to find the optimal linear filter for decoding this signal, we want to minimize the mean square error, as we did in (2.4):

$$\begin{aligned} E &= \left\langle [x(t;\mathbf{A}) - h(t) * R(t;\mathbf{A})]^2 \right\rangle_{t,\mathbf{A}} \\ &= \left\langle \left[x(t;\mathbf{A}) - \sum_{i,k} h(t - t_{ik}(\mathbf{A})) \right]^2 \right\rangle_{t,\mathbf{A}}, \end{aligned} \tag{4.17}$$

where $*$ indicates convolution. Rather than attempting to minimize this error in the time domain, we can use the Fourier transform to express the error in the frequency domain. This has the immediate advantage of turning the convolution in (4.17) into a multiplication. As a result, we can write the error for each frequency channel as (see appendix B.3)

$$E(\omega_n) = \left\langle \frac{1}{2\pi} |A(\omega_n) - h(\omega_n) R(\omega_n;\mathbf{A})|^2 \right\rangle_{\mathbf{A}}. \tag{4.18}$$

Unfortunately, this is a very high-dimensional integral. Specifically, it will have as many dimensions as there are frequency components needed to define the ensemble of signals. It is very difficult to directly evaluate this kind of integral, so instead we perform a Monte Carlo estimate of its solution. In other words, we generate a large number of actual signals, $x(t; \mathbf{A}^\alpha)$ (where α indexes a particular set of values for our amplitudes, $\mathbf{A}$), find the spike train responses from the pair of neurons, $R(t; \mathbf{A}^\alpha)$, and then determine the linear filter, $h(t)$, that minimizes the average error between the linear estimate of the signal and the original signal (see appendix B.3 for details).

This analysis, though more intuitive in the time domain, is more efficient in the frequency domain. This is because the number of degrees of freedom (i.e., the number of free parameters used to estimate our linear filter) is much smaller in the frequency domain, even though the results are equivalent. To see why this is the case, consider the following example of finding the optimal filter for the retinal ganglion parvo cells. The peak frequency response of these cells is at about 30 Hz (Van Essen and Anderson 1995). Assuming that the membrane leakage time constant is approximately 20 ms (and thus that effects of inputs about 100 ms in the past can be ignored (i.e., $5\tau_{RC}$)), we can determine the number of frequency components needed to characterize the filter. As mentioned earlier,

$$\begin{aligned} \Delta_\omega &= \frac{2\pi}{T} \\ &= \frac{2\pi}{100\,\mathrm{ms}} \\ &= 10\,\mathrm{Hz}. \end{aligned}$$

So the number of components needed is

$$N = \frac{30}{10} = 3.$$

Each of these components is complex, so the problem of describing the linear filter $h(\omega)$ has approximately 6 degrees of freedom. In contrast, if we were to try and estimate the filter in the time domain, where sample times necessary for good approximations are on the order of .1 ms or less, we would have approximately 1000 points in $h(t)$ to estimate (i.e., almost three orders of magnitude more degrees of freedom). Of course we could put various constraints on how these parameters were estimated in order to reduce the complexity of the problem. However, in the frequency domain, we have a built-in, unbiased means of reasonably limiting the number of parameters that need to be estimated to find the linear filter.

Theoretically speaking, in order to estimate this filter, we need only run many short trials (of size $T = 5\tau_{rc}$ or so). However, large numbers of short trials is expensive both experimentally and numerically, and startup transients are likely to influence the results of such experiments. So, as also done by Rieke et al. (1997), we have developed a means of 'windowing' signals of several seconds so as to analyze these longer experiments as if they were many shorter experiments. As discussed in more detail in appendix B.3, there are several advantages to our method over existing ones. Most importantly, our windowing method makes more efficient use of the data from short

trials, decreasing the likelihood of error from transients. This has important practical consequences since many experimental preparations cannot be maintained for the long run times needed to employ previous methods.[12]

As shown in appendix B.3, minimizing the windowed version of (4.18) gives

$$h(\omega_n) = \frac{\langle A(\omega_n) R^*(\omega_n; \mathbf{A}) \rangle_{\mathbf{A}}}{\left\langle |R(\omega_n; \mathbf{A})|^2 \right\rangle_{\mathbf{A}}}, \tag{4.19}$$

where the convolution with the window is included in the averaging, $\langle \cdot \rangle_{\mathbf{A}}$, to emphasize that this is a means of approximating the average over the stochastic variables, $\mathbf{A}$. From the numerator of (4.19) we can see that the filter will have a low amplitude if there is not much correlated power between the input amplitudes and the spike train amplitudes. The denominator tells us that this is also true if the average power of the spike trains is large at a particular frequency. This can occur if the neurons have a high mean background firing rate that generates power that is uncorrelated with the input signal.

Those familiar with past analyses of this kind may be wary of certain limitations of this kind of filter. In particular, Rieke et al. (1997) note that linear filtering is far more likely to work if the correlation time of the signals being encoded are significantly shorter than the mean interspike interval (ISI) (p. 86). This is because having a short correlation time relative to mean ISI ensures that individual spikes are providing independent information about the signal. If this was not the case, then there may be significant interactions between neighboring spikes, i.e., both spikes may be providing information about the same parts of the signal. However, there is nothing in the preceding analysis that makes assumptions about the relation between the mean ISI and the correlation time of the input signal. So, it is merely a *possibility* that linear decoding will not work in the case where correlation times are long compared to mean ISI. In fact, as we show in section 4.4.1, linear decoding performs very well even for long correlation times.

Fortunately, we are now in a position to quantify precisely what it means for our decoder to work well. To determine how accurately our optimal linear filtering approximates the original signal, we can substitute (4.19) back into (4.18) to obtain the residual error, E_r:

$$E_r(\omega_n) = \left\langle |A(\omega_n)|^2 \right\rangle_{\mathbf{A}} - \frac{\left| \langle A(\omega_n) R^*(\omega_n; \mathbf{A}) \rangle_{\mathbf{A}} \right|^2}{\left\langle |R(\omega_n; \mathbf{A})|^2 \right\rangle_{\mathbf{A}}}.$$

This error will only be zero when there are no interharmonic distortions that generate spurious power at frequency ω_n. Nonlinear processes, even those as simple as the LIF neuron, generally do suffer from these kinds of distortions (Anderson et al. 2000). The residual error is a good measure of how well our linear decoding works, and is analogous to what we called 'static error' in chapter 2.

We have now shown how to define the representation of time dependent signals in

[12] The method used by (Rieke et al. 1997) was employed in the context of experimental run times as long as 2 days.

pairs of spiking neurons. Specifically, we have defined the encoding

$$R(t;\mathbf{A}) = \sum_{i,k}^{M} \phi_i \delta(t - t_{ik}(\mathbf{A})), \tag{4.20}$$

and decoding

$$\hat{x}(t) = h(t) * R(t;\mathbf{A}), \tag{4.21}$$

where

$$h(\omega_n) = \frac{\langle A(\omega_n) R^*(\omega_n;\mathbf{A})\rangle_{\mathbf{A}}}{\left\langle |R(\omega_n;\mathbf{A})|^2 \right\rangle_{\mathbf{A}}}. \tag{4.22}$$

4.3.4 Discussion

Now that we know how to find the optimal temporal filter, and have a means of determining how good it is, let us consider the linear decoder, $h(t)$, in more detail. As just noted, our estimate of the signal that is encoded by the neurons into the spike train is given by (4.21). Substituting (4.20) into (4.21), and recalling the result in (4.13), we can write

$$\hat{x}(t) = \sum_{i,k}^{M} \phi_i \delta(t - t_{ik}) * h(t) \tag{4.23}$$

$$= \sum_{i,k}^{M} \phi_i h(t - t_{ik}). \tag{4.24}$$

In essence, this equation says that our estimate is found by putting a waveform in the shape of the linear filter at each spike time and summing the results (see figure 4.7).

This estimate is closely related to the population estimates we used in the previous chapters. In particular, we could re-write (4.24) as

$$\hat{x}(t;\mathbf{A}) = \sum_{i}^{M} a_i(x(t;\mathbf{A}))\phi_i(t), \tag{4.25}$$

where M is the number of time-steps we have divided the signal into, the a_i are 1, 0, or -1, depending whether or not a neuron emitted a spike, and the $\phi_i(t)$ are all time-shifted versions of $h(t)$ (i.e., $h(t - t_i)$). This notation is unusual because the neuron activity in the population code is mapped onto activity (the presence of a spike) at some time t_i, so, as in the case of the population code, for each 'active element' we need one decoder. This results in a large number, M, of temporal decoders, all of which are essentially the same. While awkward, this notation shows exactly how the temporal decoders and the population decoders perform the same function—both serve to translate an activity back into the original domain that was encoded (i.e., either x for the population code or $x(t)$ for the temporal code). As an aside, it is interesting to note that most of the coefficients, a_i, in (4.25) will be zero for any particular signal. In this sense, the representation is a 'sparse' representation. It makes sense that the neural

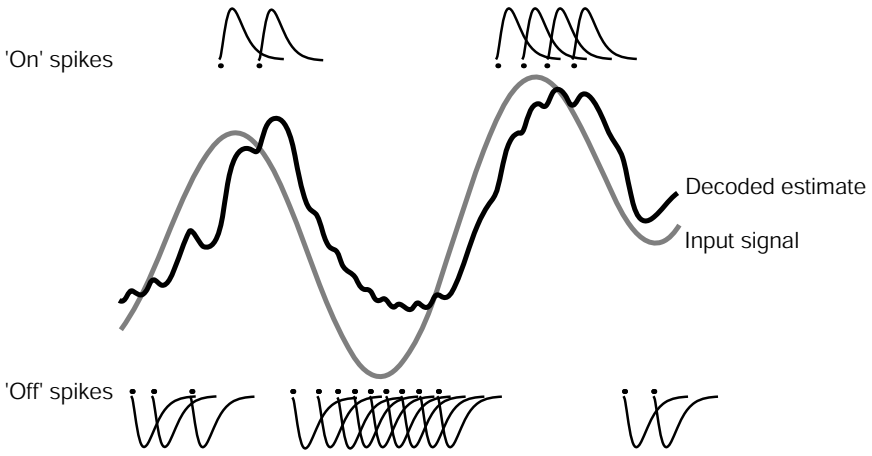

Figure 4.7: An example of linear filtering. The input signal, $x(t)$, is fed into the somas of a pair of on-off neurons which encode the signal into 'on' and 'off' spikes. To get an estimate, $\hat{x}(t)$, of that signal, we can linearly filter those spike trains by effectively placing the filter at the time of occurrence of each spike and summing the result. When the on and off neurons are symmetrical, their respective filters will be 'mirror images', as shown in the figure.

code is sparse, as this results in more efficient coding (Olshausen 2000) and a better use of available energy (Baddeley 1996).

Because $x(t)$ is a *function*, its representation is much like the representation of $x(\nu)$ that we discussed in section 3.2. Looking again at equation (3.6) we see that it is indeed very similar to (4.25). However, there is also an important difference between population and temporal encoding that becomes evident from this comparison. Namely, there is no temporal encoding function in the sense of $\tilde{\phi}_i(\nu)$. This is because the temporal encoding is defined completely by the intrinsic properties of the neuron, which are captured by $G_i[\cdot]$. This difference means that it is much more difficult to derive the same kinds of analytical results for understanding temporal coding as we do for population coding (see section 7.3).

Nevertheless, it proves to be useful that both temporal and population codes in neurons can be characterized using linear coding, since it allows us to unify these two kinds of coding (as we discuss in section 5.1). Before doing so, however, let us consider a number of examples of how to use this characterization of temporal coding to measure the efficiency of information transmission in neural models. We begin with the simple LIF neuron and progress to more complex models. Perhaps the most important lesson to be learned from these examples is that the basic LIF model has just about the same information transmission characteristics as its more complex counterparts. And, both kinds of models perform comparably to real neurons.

4.4 Information transmission in LIF neurons

While we now have a method for characterizing temporal representation, there are two related issues that we have so far ignored or only partially addressed. First, we suggested that this analysis applied to spike trains with both short and long-correlation times, unifying rate and timing codes—we must show that this is the case. And second, we did not discuss how these optimal decoders (or 'filters') relate to their biophysical counterparts, the postsynaptic currents (PSCs). Although these subsequent discussions rely solely on applying our method to the LIF model, in section 4.4.2 we relate these results to the information characteristics of real neurons. As well, in section 4.5 we compare LIF results with the same measures in more complex neuron models. These comparisons show that it is reasonable to use simple LIF neurons in simulations of neural information processing.

4.4.1 Finding optimal decoders in LIF neurons

It is important that nothing in our discussion of optimal linear decoders depends on the nature of the encoder. Of course, choosing a specific encoder greatly affects the particular optimal decoder that we find, but the methods and analyses we discuss are independent of the encoder. In this section we present results using LIF neurons as our encoder. For these simulations, and all similar ones in this chapter, we assume that our signal ensemble is band-limited Gaussian white noise (an assumption shared with the experiments on real neurons that we compare our results to). As well, the encoding in each case is done by a symmetrical pair of 'on' and 'off' neurons. This ensures that the optimal filter is the same for both neurons (as we have assumed to this point, see (4.24)). The LIF neurons we use for our analyses have a background firing rate of 40 Hz, a refractory period of 2 ms, and an RC time constant of 20 ms.

In figure 4.8, the optimal linear decoder is shown in both the frequency and time domains. The decoder displayed here was found using the sliding window analysis described in section 4.3.3 and appendix B.3 on the signal shown in figure 4.9a. Using four seconds of data, we are able to find a filter that very successfully decodes the original input signal, as well as other signals that were randomly drawn from the same ensemble (as shown in figures 4.9b and 4.9c).

We can see from figure 4.10 that the encoding of the original signal via the LIF neuron into a spike train introduces spurious power, especially at high frequencies. This is not surprising given the 'spiky' nature of the encoding. The power at frequencies that are actually in the signal, however, are also well-preserved. Thus, one of the main functions of the decoding filter is to remove these spurious high frequency components while not otherwise altering the spectrum; i.e., it acts as a low pass filter. As can be seen in figure 4.8, the filter does indeed resemble a low pass filter. Thus it is localized in time (figure 4.8b) and nearly flat for frequencies below some cutoff (50 Hz in this case) in the frequency domain (figure 4.8a). The precise width of the optimal filter in the time domain depends on the kinds of signals that were used to find it. For higher frequency signals, it will be thinner, increasing the cutoff frequency of the low-pass filter. Conversely, for a lower frequency set of signals, it will be wider in the time domain and have a lower cutoff frequency. It is also worth noting at this point that

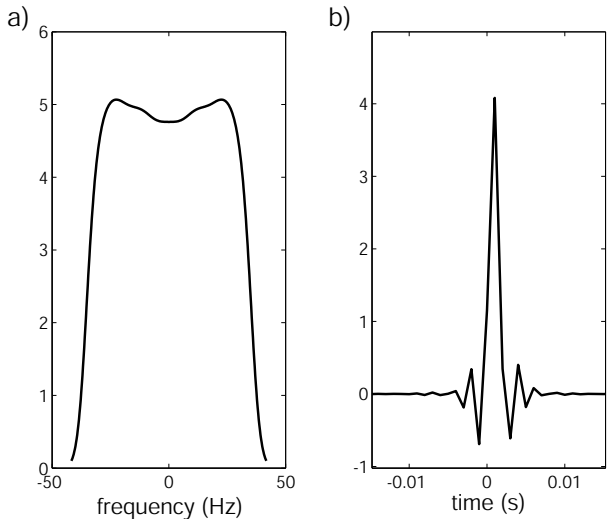

Figure 4.8: An optimal linear decoder for a pair of LIF neurons in a) the frequency domain and b) the time domain.

the filter somewhat resembles a dendritic PSC. Specifically, it is localized in time and has a large, rapidly decaying initial response. However, it is clearly non-causal as it is defined over both negative and positive parts of the time axis.

As mentioned earlier, there is some concern about being able to use such decoders for signals with long correlation times (see Rieke et al. 1997, p. 86). However, in figure 4.11, it is evident that the optimal filter found for the signal with a short correlation time (figure 4.9) works equally well for a signal with a long correlation time. This is true in general. A thousand trials of both slow (i.e., frequency channels between 0 and 4 Hz) signals with long correlation times and fast signals (i.e., frequency channels between 0 and 30 Hz) with short correlation times results in a mean RMSE for the signals with long correlation times of 0.123 and mean RMSE for those with short correlation times of 0.147. This is true *despite* finding and using the optimal filter for the *fast* signals only (i.e., short correlation times). Thus, the same filter not only works for signals with long correlation times, it actually works better than for signals with low correlation times. This is to be expected, since RC circuits (like the LIF neuron) function as low pass filters, and thus are better able to preserve lower frequencies. However, as shown in figure 4.12, finding the optimal filter using a long correlation time ensemble results in an even better estimate of such signals, as expected.

By comparing figures 4.9 and 4.11, we can see how the same neuron is using something more like a 'timing' code in the first case, and something more like a 'rate' code in the second case (look especially at the spike times, indicated by small dots). By comparing figures 4.11 and 4.12, we can see that both seem to use a kind of rate code. However, in figure 4.12, the optimal filter is much wider in the time domain because

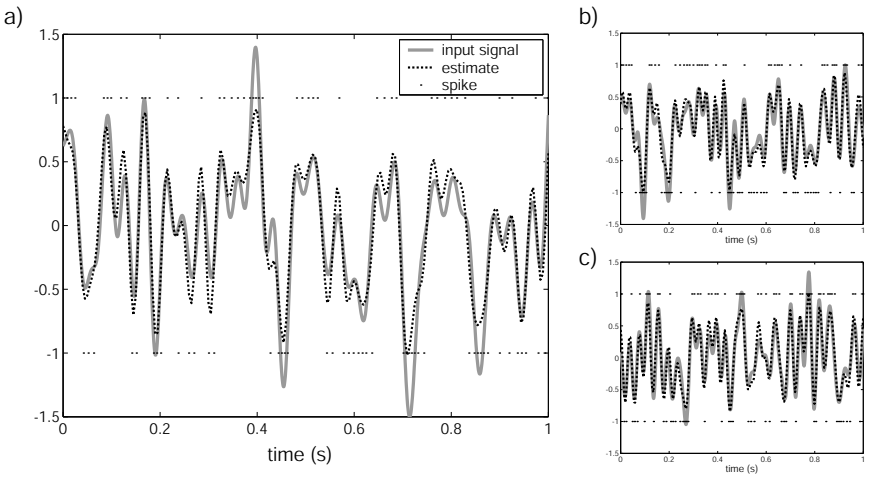

Figure 4.9: a) Reconstruction of the randomly generated signal used to find the optimal filter in figure 4.8 (RMSE = 0.153). Figures b) and c) show reconstructions of randomly generated signals from the same distribution as the signal in a). These were reconstructed using the filter found using the signal in a). RMSE = 0.149 for b) and RMSE = 0.142 for c).

it was constructed from a set of signals with more similar statistics. If the optimal filter gets wider (while remaining equally smooth), then the precise timing of spikes becomes less important for decoding the signal, thus acting more like a rate code. We have verified this by introducing random jitter in the spike times that encode either the fast (short correlation time) or slow (long correlation time) signal. As expected, the RMS error increases significantly more when decoding the jittered spike trains for fast signals (using the 'fast' optimal filter) than for slow ones (using the 'slow' optimal filter), despite the fact that the spike trains have approximately the same total number of spikes. As well, the RMS error increases significantly more for the slow signals decoded with the 'fast' optimal filter. Thus, the average firing rate is more informative about the slow signals, as long as we know it is drawn from a distribution with long correlation times.

So, in conclusion, we have found a means for determining an optimal decoder that works well under a variety of conditions. Not surprisingly, the optimal filter generally works better for a signal drawn from an ensemble with characteristics similar to those used to find the optimal filter. Nevertheless, filters found from ensembles that generally result in short correlation times work very well for signals with longer correlation times. So, this method of finding optimal filters bridges the gap between timing and rate codes because it works effectively in either realm.

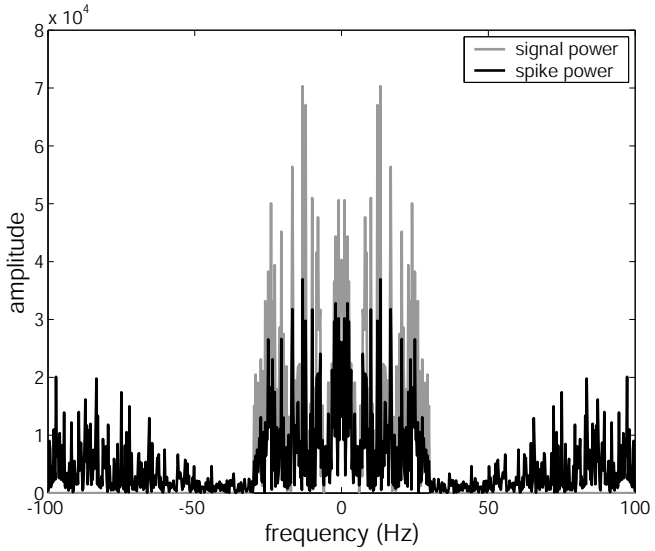

Figure 4.10: Comparisons of the power of the original signal (depicted in figure 4.9a) and the spike train which encodes the signal (also depicted in figure 4.9a). Note that all power in the spikes signal above 30 Hz is not in the original signal.

4.4.2 Information transmission

In order to compare the behavior of LIF information processing to that of natural neural systems, let us briefly review the information characteristics observed for real neurons. Chief amongst the measures of information processing is information transmission rates, measured in bits per spike or bits per second. These measures are surprisingly consistent across many different neural systems. In the cricket cercal system, which measures wind velocity, information rates of between about 150 (Roddey and Jacobs 1996) and 300 (Bialek and Rieke 1992) bits per second have been reported.[13] These rates are equivalent to between 1.1 and 3 bits per spike. In the bullfrog sacculus, which senses ground-borne vibrations, Bialek et al. (1991) report transmission rates of about 3 bits per spike. As well, Bialek et al. (1991) show that motion-selective H1 neurons in the blowfly visual system carry about 3 bits per spike. In salamander retina, recent results suggest that information is transmitted at a rate of about 3.4 bits/spike (Berry II and Meister in press). In primate visual area V5, information transmission rates of 1–2 bits per spike have been observed (Buracas et al. 1998). The highest transmission rates we have seen reported are for the bullfrog auditory neurons, which reach rates as high as 7.8 bits per spike (Rieke et al. 1997, p. 185). Notably, these rates were only achieved for stimuli with frequency spectra of naturally occurring bullfrog calls. Broadband stimuli had transmission rates of about 1.4 bits per spike. In sum, natural

[13] Although Miller et al. (1991) calculate the rate to be about 40 bits per second, they used a 100 ms binned rate code to calculate information.

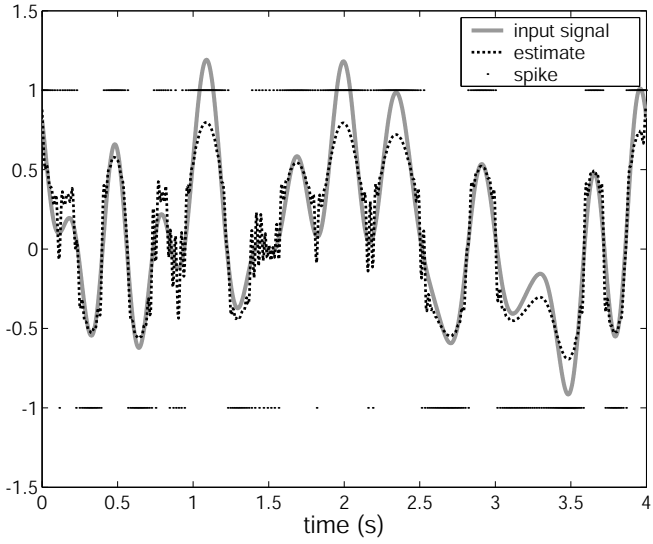

Figure 4.11: Reconstruction of a randomly generated signal from a distribution guaranteeing a low frequency signal, using the optimal filter from figure 4.8. This filter was found using a signal distribution with high frequencies as well. Nevertheless, this reconstruction works very well (RMSE = 0.143). Note that the time scale has been lengthened.

sensory systems are generally able to encode stimuli with between about 1 and 3 bits of information per spike (see also Rieke et al. (1997) for a review).

These are impressively high transmission rates that approach the optimal possible coding efficiencies (de ruyter van Steveninck and Bialek 1988; Rieke et al. 1997). In the frog sacculus, the cricket cercal system, the bullfrog auditory system, and the electric fish electrosensory system, the codes are between 20 and 60% efficient (Wessel et al. 1996; Rieke et al. 1997, p. 180). And, efficiency significantly increases when stimuli are restricted to be more like naturally occurring stimuli of interest to the organism (Rieke et al. 1997, p. 185).[14] All of these measures of information transmission performance in natural systems are found by closely related methods. The researchers use opponent neurons, assume stationarity, and find optimal linear filters, just as we have done. What is of interest to us, as modelers, is to see how these measures in neurobiological systems compare to those for model neurons. So, in the remainder of this section, we perform a similar analysis for the LIF neuron.

To begin, we must realize that model neurons are entirely deterministic. Thus, un-

[14] Also note, that the estimation of information transmission rates using this method places a *lower* bound on the amount of information transmitted by the code. There are reasons to think this bound generally *under*estimates the actual amount of information transmitted (Stevens and Zador 1996). Thus, efficiencies are likely even higher.

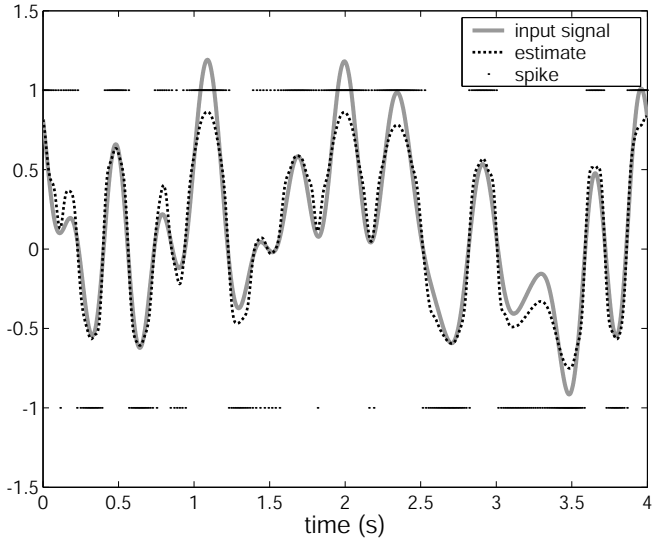

Figure 4.12: Reconstruction of a the same signal from figure 4.11 using an optimal filter based on a long correlation time (i.e., low-frequency) ensemble (RMSE = 0.122). The error is slightly improved, and the high-frequency aspects of the reconstruction in figure 4.11 are removed.

like real neurons, they can be run under conditions of no noise. Technically, then, information transmission rates can be unlimited. However, we can still find an analogous information measure because we linearly decode a nonlinear system. In particular, we can compare the total variance to the variance that is unexplained by our linear decoding procedure. Usually noise is the source of unexplained variance, but in this case, it is the result of our linear filtering.

We can derive and express the information transmission per frequency channel as (see appendix B.4 for the derivation)

$$\mathrm{Info}(\omega_n) = \frac{1}{2}\log_2\left[\frac{\left\langle |A(\omega_n)|^2\right\rangle_{\mathbf{A}}}{\left\langle |A(\omega_n)|^2\right\rangle_{\mathbf{A}} - \frac{\left|\langle A(\omega_n)R^*(\omega_n;\mathbf{A})\rangle_{\mathbf{A}}\right|^2}{\left\langle |R(\omega_n;\mathbf{A})|^2\right\rangle_{\mathbf{A}}}}\right] \tag{4.26}$$

$$= \frac{1}{2}\log_2\left[\frac{\left\langle |R(\omega_n;\mathbf{A})|^2\right\rangle_{\mathbf{A}}}{\left\langle |R(\omega_n;\mathbf{A})|^2\right\rangle_{\mathbf{A}} - \frac{\left|\langle A(\omega_n)R^*(\omega_n;\mathbf{A})\rangle_{\mathbf{A}}\right|^2}{\left\langle |A(\omega_n)|^2\right\rangle_{\mathbf{A}}}}\right]. \tag{4.27}$$

In (4.27), the numerator expresses the total variance at the output, while the denominator expresses the variance that is not explained by our assumption that the output is linearly related to the input. Equation (4.26) has a similar interpretation, though in

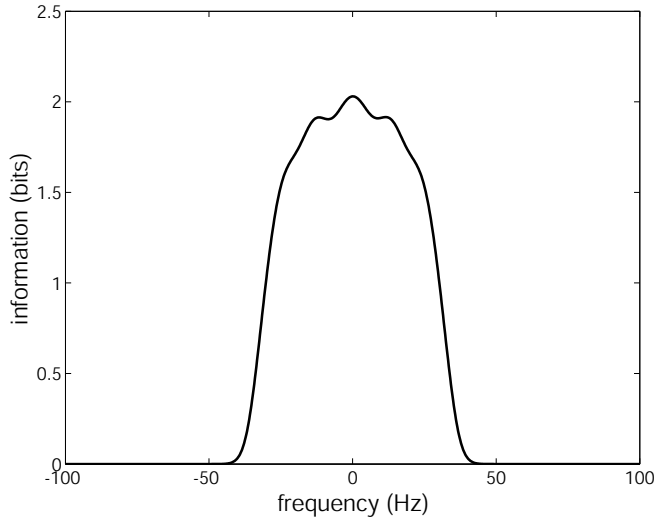

Figure 4.13: Information transmission for the example in section 4.4.1 (see figures 4.9, 4.10, and 4.11). This shows the amount of information decodable from the original signal at each frequency band using the optimal filter.

terms of the input signal. Because we assume linear decoding, these equations express only a lower bound on the amount of information that could be transfered through this system (figure 4.13 depicts the application of these equations to the example in the section 4.4). The resulting information transmission rates for this example are 1.24 bits per spike or 114 bits per second. Both of these measures are comparable to those reported earlier for real neurons.

If we had a more sophisticated, nonlinear decoding technique, we could get a better estimate of the input signal given the output, and thus increase the information transmission. However, Rieke et al. (1997) argue that nonlinear decoding only provides about a 5% improvement in information transmission in real neurons. And, more importantly, it is unclear how such nonlinear techniques relate to biophysical mechanisms.

Linear filtering does not suffer this same lack of biological realism because it can be related to the production of postsynaptic currents (PSCs) in the postsynaptic cell. In fact, we can use (4.27) to determine the information transmission using PSCs instead of the optimal filter. Assuming a simple PSC model, $h_{psc}(t) = e^{-t/\tau_{syn}}$, where τ_{syn} is the synaptic time constant, we can perform an analogous analysis on the resulting decoding.[15] These results are shown in figure 4.14. As can be seen from this figure, information transmission using PSCs compares very favorably to that using the optimal filter. Over the entire range, there is a 6% decrease in the information transmitted using

[15]The analysis is not strictly identical because, in order to determine the PSC information transmission, we replace the spike train with the PSC filtered signal. That is, R in (4.26) and (4.27) is the frequency domain representation of the PSC filtered signal for this analysis.

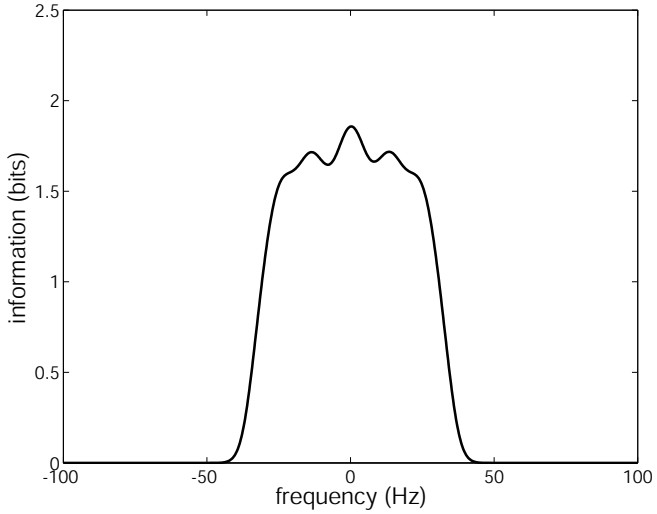

Figure 4.14: Information decodable using the postsynaptic current (PSC) as a temporal filter instead of the optimal filter as in figure 4.13.

the PSCs compared to the optimal filter. This is a very small loss considering the vast increase in neurological plausibility when using PSC decoding.

Figure 4.15 provides an example of decoding a signal with PSCs, which can be compared to the optimal decoding in the previous section (see figure 4.9a). Here, we can see that the decoding is good, though not as good as for the optimal case. However, information transmission rates are similar at 1.17 bits per spike or 108 bits per second; again similar to what is observed in real neural systems. As well, despite the fact that the RMS error is about twice that of the optimal decoders, it is still reasonably low. This is not too much of a concern since we are only using two neurons in this example. Given what we know about the effects of increasing the number of neurons in the population, we can be reasonably confident that this error can be reduced by adding neurons (in chapter 5, we explicitly show this to be the case). Given these promising results, and the vastly increased physiological plausibility of models that rely on PSCs as their temporal filters, all of the simulations we present in subsequent chapters use PSCs as the temporal decoder.

It is worth noting that there are other methods for finding information transmission rates that often result in higher transmission rates (Stevens and Zador 1996; Buracas et al. 1998). However, by far the majority of researchers who have found information rates in real neural systems have used reconstruction methods like the one we have adopted. Thus, taking this approach facilitates comparisons with the analyses done on real neurobiological systems. Nevertheless, Stevens and Zador (1996) have made it clear that such reconstruction methods do not provide the highest possible lower bound on information transmission.

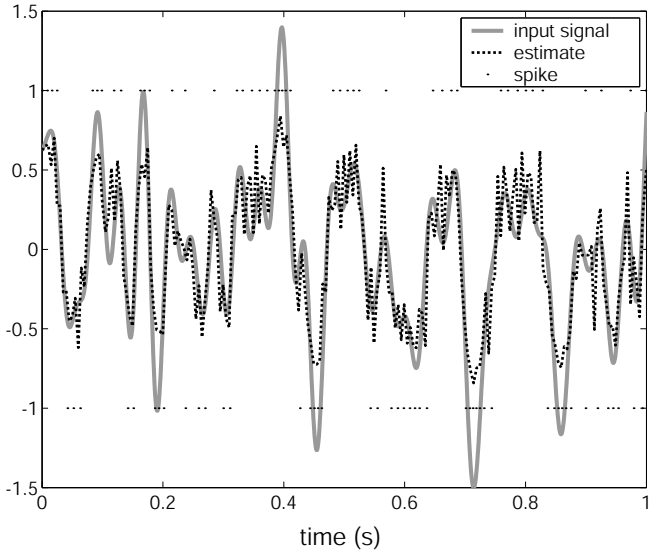

Figure 4.15: Signal reconstruction using PSC (RMSE = 0.287). This compares favorably to the reconstruction using the optimal filter in figure 4.9a.

4.4.3 Discussion

Our central focus to this point in the chapter was on finding optimal decoders for characterizing temporal representation in neurobiological systems. However, we noted along the way that we do not expect there to actually be such optimal decoders in real neural systems. So why bother finding optimal decoders? There are a number of reasons. First, optimal decoders allow us to calculate quantitative measures of the system's performance. This means we can objectively compare the information processing characteristics of neural models to actual neurons. Second, developing such measures provides bounds on the performance of systems that might use non-optimal decoders, like PSCs. Third, if the optimal linear decoder does a good job of decoding, and the non-optimal decoder is substantially like the optimal decoder, then it is reasonable to assume that neural systems can be well-characterized as doing linear filtering of spike trains. Fourth, once we have used the optimal decoder to show that linear decoding is useful, we can more easily combine our characterization of the temporal code with that of the population code (as we do in section 5.1). Finally, getting a general handle on how to understand filtering in neural systems (optimal or not) should ultimately allow us to predict what kinds of filters (i.e., the characteristics of the PSCs) we expect to see given the task of certain neural systems. This is not an avenue we have explored in detail, although finding and quantifying optimal filters is a first step along this path.

4.5 More complex single neuron models

It is important to compare the LIF neuron to more complex models because there are serious concerns regarding the neurobiological plausibility of the LIF model. For example, the LIF model is very limited with respect to its modeling of the active spiking behavior of neurons. In particular, the so-called 'spike' in LIF models is simply 'stuck on' to the output when the membrane voltage crosses threshold. This is done essentially to make the output *look* like that produced by real neurons. More complex models include detailed descriptions of the active mechanisms that actually produce the voltage changes that occur during the spiking process. As well, LIF models include only a single ion current and thus do not have adapting firing rates. In mammalian cortical neurons, there are at least 12 different ion currents (Gutnick and Crill 1995; Johnston and Wu 1995). Furthermore, in a large subset of these neurons (e.g., the 'regular-spiking' cells), some of these currents result in spike adaptation.

Our strategy in this section is to consider successively more complex models. Thus we begin by introducing an extra ion current needed to explain adaptation into an LIF model (section 4.5.1). However, this model still does not explain spiking. Next, we consider a model that has been developed as a canonical model of a large class of neurons (section 4.5.2). This model is called the θ-neuron, and includes the spike generation process as part of its intrinsic dynamics (Ermentrout 1996; Gutkin and Ermentrout 1998a). However, this model does not include adaptation and makes mathematical reductions unfamiliar to most neurophysiologists. For these reasons, we conclude by considering a conductance-based model, which we call the Wilson model, that includes a number of currents that account directly for adaption and the relevant dynamics of the spiking process (section 4.5.3).

During our consideration of each of these models, we analyze their information processing behavior using the methods developed previously. As a result, we conclude by comparing all four models to show that the various increases in complexity do not significantly affect information transmission. And, just as important when constructing large simulations, we show that the LIF neuron is a far less computationally demanding model to run.

4.5.1 Adapting LIF neuron

Adaptation, or slowing of the spike rate, is seen prominently in what are called 'regular spiking' cortical neurons (Connors and Gutnick 1990). When these neurons are injected with a depolarizing step function, they spike rapidly at first, but quickly slow their firing rate to a much lower steady-state firing rate (see figure 4.16). In order to capture this behavior in a leaky integrate-and-fire (LIF) model, we can incorporate the voltage dependent resistance as shown in figure 4.17 (Wehmeier et al. 1989; Koch 1999). This mimics the effects of the slow hyperpolarizing potassium current (J_{adapt} in figure 4.17) found in regular-spiking cells.

We can write the equations governing this circuit as follows:

$$\frac{dV}{dt} = -\frac{1}{\tau^{RC}}\left(V(1+\frac{R}{R_{adapt}}) - J_M R\right) \tag{4.28}$$

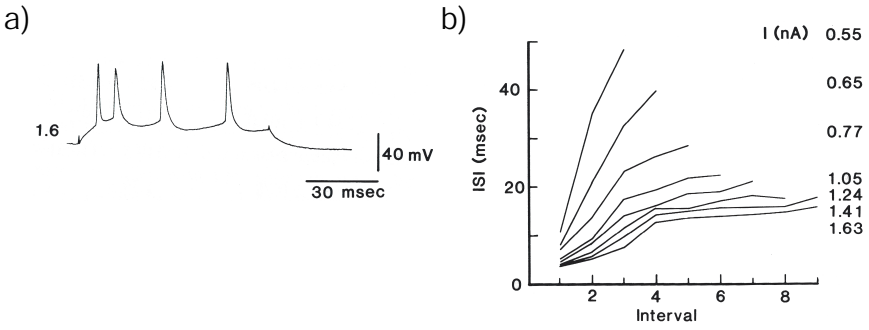

Figure 4.16: a) An adapting spike train recorded from a human regular spiking cortical neuron with an input current of 1.6 nA. b) Adaptation at different input strengths. ISI is the interspike interval, which increases with interval number for every input current. (From McCormick et al. 1985 © The American Physiological Society, reproduced with permission.)

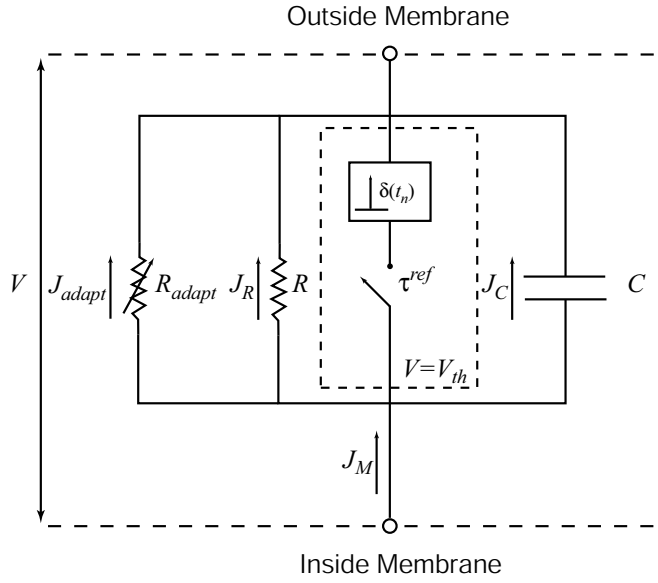

Figure 4.17: RC circuit for LIF with adaptation. The dynamics of variable resistor, R_{adapt}, are controlled by an adaptation time constant, τ_{adapt}. (See figure 4.2 for the circuit with no adaptation).

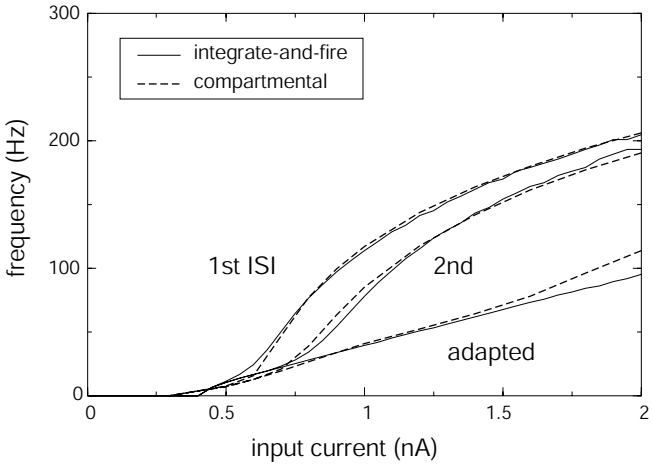

Figure 4.18: Comparison of an adapting LIF neuron with a biophysically detailed, conductance-based model of a layer 5 pyramidal neuron. (From Koch 1999, p. 337 © Oxford University Press, reproduced with permission.)

$$\frac{dR_{adapt}}{dt} = \frac{R_{adapt}}{\tau_{adapt}}.$$

Equation (4.28) is identical to (4.4) with the addition of the time varying resistance, R_{adapt}. Notably, the time constant that controls the speed of adaptation, τ_{adapt}, will be large compared to the RC time constant of the circuit, τ^{RC}. Assuming the cell is initially at rest (i.e., $V = 0$ and $R_{adapt} = \infty$), once V_{th} is passed and an action potential is produced, the voltage dependent resistance, R_{adapt}, begins to decrease (i.e., the conductance increases) by some value, G_{inc}.[16] This introduces an extra current, J_{adapt}, which acts to lessen the effects of the input voltage, J_M. Thus, it takes longer for the next action potential to be generated because there is a larger difference between $V_{threshold}$ and V_{reset} than there was at rest. When there is no input, the resistance drifts exponentially towards its resting state.

Perhaps surprisingly, this simple addition to the LIF model makes it behave quite similarly to detailed conductance-based models, as shown in figure 4.18. As can be seen in this figure, the strong nonlinearity of the LIF model near threshold is also removed by the inclusion of adaptation. In fact, Ermentrout (1998a) has shown that, in general, adaptation in highly nonlinear neurons serves to linearize the response function. Notably, this linearization of the response function makes it more likely that we can find a good linear decoder.

Comparing figures 4.19 and 4.9a is instructive regarding the effects of including adaptation in our model neuron. As can be seen, the resultant decoding is very sim-

[16] For mathematical convenience, it is easier to model the decreasing resistance as an increasing conductance. Note that the conductance is 0 when the resistance is ∞.

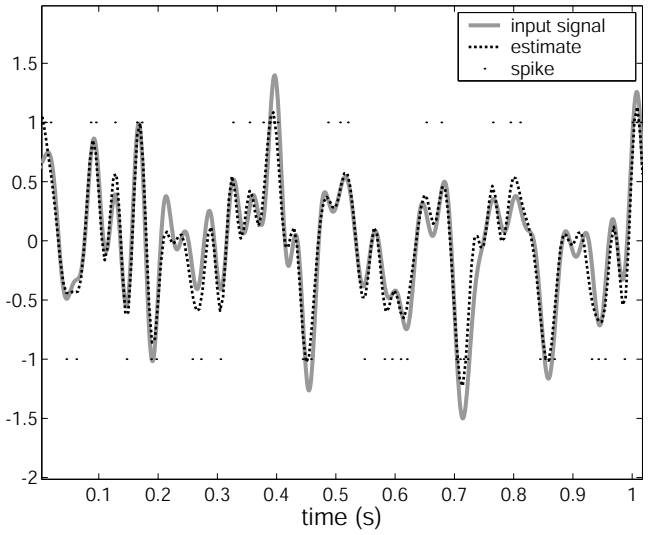

Figure 4.19: Signal reconstruction using the optimal filter as the decoder and the adapting LIF model as the encoder ($\tau_{adapt} = 55$ ms, $G_{inc} = 20$ nS, RMSE = 0.153). This can be compared to figure 4.9a, where the same signal and decoder are used but the encoder is the non-adapting LIF model.

ilar to the original LIF. In fact, both the RMSE and the information transfer rate are the same. However, the efficiency of the adapting LIF is significantly higher (2.23 bits/spike) than the standard LIF (1.24 bits/spike). This suggests that the ubiquity of adaptation in cortical neurons might serve to improve the efficiency of information transfer.

So, the adapting LIF model is just as good, if not better than the standard LIF model. However, there is also a 25% increase in the length of time it takes to simulate the adapting model, so the computational costs are significantly higher.

4.5.2 θ-neuron

Recently, there have been a number of attempts to generate simple, nonlinear, spiking models that effectively capture the behaviors of an entire *class* of neurons (Gutkin and Ermentrout 1998a; Ermentrout 1996; Hoppensteadt and Izhikevich 1997; Hoppensteadt and Izhikevich in press). Most of these are focused on understanding the dynamics of what are called 'class I' neurons. Hodgkin (1948) proposed a distinction between two classes of neurons, where class I neurons are those that can spike at arbitrarily low frequencies and steadily increase their frequency as a function of input

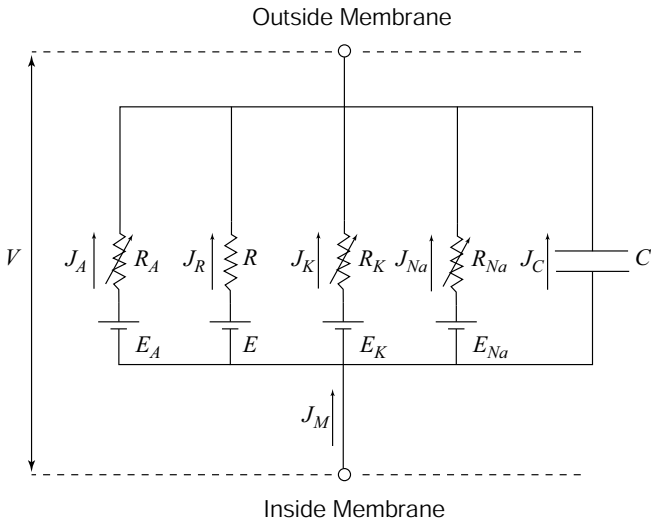

Figure 4.20: RC circuit for class I neurons. The parameters R_{Na}, R_K, R_A, and R are the sodium, slow potassium, fast potassium (A-current), and membrane leakage resistances, respectively. The equilibrium potentials, E, for the respective currents are shown as batteries. For further related discussion see section 4.5.3.

current.[17] Notably, mammalian neurons are almost all class I neurons (Wilson 1999b, p. 149).

Class I neurons are very similar to Hodgkin-Huxley neurons, except that they incorporate an extra, very fast, voltage dependent potassium current, called the A-current (J_A in figure 4.20). Notice that the circuit describing class I behavior (figure 4.20) no longer incorporates a delta function generator as in the case of the LIF neuron. This is because the time courses of the voltage dependent potassium (R_K) and sodium (R_{Na}) channels are responsible for the generation of the neural spike (for a discussion of the dynamics of these currents see Nelson and Rinzel 1995).

It is natural to characterize the dynamics of class I cells in the language of nonlinear systems theory (see Rinzel and Ermentrout 1989 for a discussion of nonlinear systems theory in the context of neural models). In particular, the class I cells contain what is known as a saddle-node bifurcation (Wilson 1999b; Hoppensteadt and Izhikevich in press). This occurs because of the change in recovery dynamics due to the nearrest threshold of the A-current activation and inactivation. Being able to describe the behavior of class I neurons in this general way has lead to the development of canonical

[17] In fact, Hodgkin suggested that there are three classes, but the third is a class of neurons that do not fire repetitively at all; presumably this is a methodological artifact. Class II neurons are those, like the famous Hodgkin-Huxley neuron (Hodgkin and Huxley 1952), that have a non-zero minimum spiking frequency, can have graded spike size, and whose firing rates are less sensitive to changes in input current strength. Arvanitaki (1938) earlier presented a similar classification.

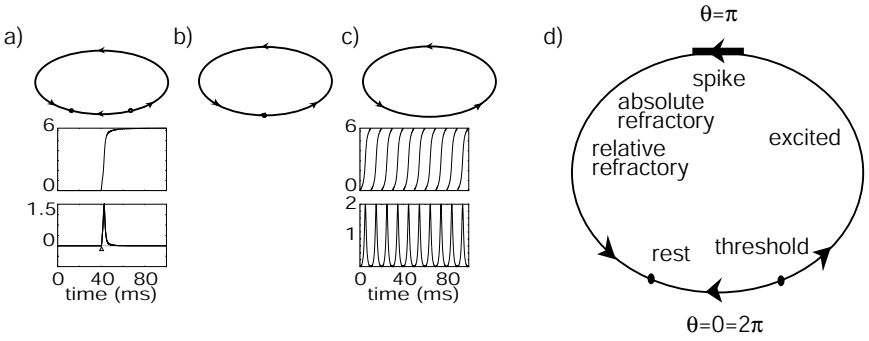

Figure 4.21: Theta-neuron behavior. In (a) and (c) the upper diagram shows the location of critical points on the invariant circle, the middle graph shows the behavior of θ, and the lower graph is the trace of $(1-\cos\theta)$ showing the spikes. (a) Excitable regime with $\beta+\sigma=-0.3$. The stable state is the node on the right. The single spike is evoked by a pulse stimulus (marked by the triangle) that forces θ past the saddle fixed point on the right. (b) Meeting of the saddle and node points with $\beta+\sigma=0$. The trajectory has an infinite period. (c) Oscillatory regime where the stable state is now a limit cycle with $\beta+\sigma=0.3$. Periodic behavior of the phase variable and spikes in $(1-\cos\theta)$ are present. (d) Phase evolution and its analog to membrane voltage states. Note that the spike occupies a small region near π. A strong enough stimulus will push θ past the threshold and into the excited region. Here the regenerative dynamics that summarize active conductances carry the phase through the spike. (Adapted from Gutkin and Ermentrout 1998a and Hoppensteadt and Izhikevich in press both © MIT Press, reproduced with permission.)

models of such neurons.[18]

In order for canonical models to be useful, they need to be simple. This way, the universal properties of the entire family of models can be studied more easily. Recently, Bard Ermentrout and his colleagues have developed a simple canonical model for class I neurons called the θ-neuron (Ermentrout and Kopell 1986; Ermentrout 1996; Gutkin and Ermentrout 1998a; Gutkin and Ermentrout 1998b). The θ-neuron is a 1-dimensional model that preserves the dynamics of a saddle-node bifurcation. Essentially, the model describes the location of the neural state vector along the spike trajectory with a single phase variable, θ. The model can be written as

$$\frac{d\theta}{dt} = (1-\cos\theta) + (1+\cos\theta)(\beta+\sigma) \quad \text{for } \theta \in [0, 2\pi], \tag{4.29}$$

where β is a bias (analogous to J^{bias} in equation (2.3)) and σ is the input (analogous to J^d).

[18] A canonical model is one which any member of a family of models can be transformed into, using a continuous change of variables (Hoppensteadt and Izhikevich 1997).

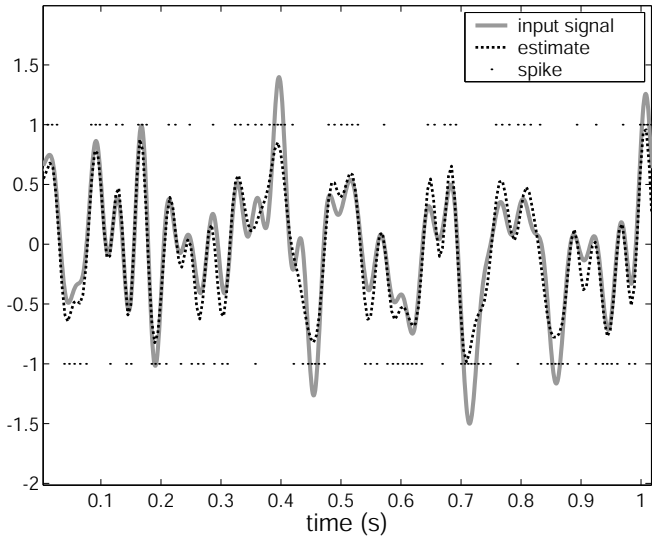

Figure 4.22: Signal reconstruction using an optimal filter and the θ-neuron model as the encoder (RMSE = 0.160).

The behavior of this model is summarized in figure 4.21. As can be seen from this figure, the θ-neuron displays the main qualitative features of a class I spiking neuron, including distinct subthreshold and superthreshold regions, an all-or-none spike, a steady-state resting point, and an absolute and relative refractory period.[19]

The θ-neuron more than just captures the qualitative features of spiking neurons. (Hoppensteadt and Izhikevich) (in press) rigorously show how complex conductance models can be reduced to models like the θ-neuron. Thus, determining the information transmission characteristics of this neuron is very useful for providing insight into the behavior of a wide variety of neurons. Figure 4.22 shows the decoded θ-neuron spike train for the same signal used in past examples. Note that the background firing rate and maximal firing rate over the range were matched the LIF models using β and a gain, α.

The θ-neuron compares favorably to the LIF neurons in most respects. The RMS error and information transmission rates are within 5% of each other. There is a slightly larger difference in efficiency, where the LIF (1.24 bits/spike) outperforms the θ-neuron (.96 bits/spike), but the values are comparable. The biggest difference between the models is that it takes approximately 100 times longer to run the θ-neuron. This, of course, is a major drawback when trying to run models of large, complex systems.

Given the generality of the dynamics of a canonical model like the θ-neuron, we take these results to be indicative of what should be found in all class I models. So,

[19]The presence of the absolute refractory period is evident from (4.29). While the neuron is spiking, the effect of the input will be minimal since $1 + \cos\theta \approx 0$ for values of θ near π.

it is reassuring to see just how similar the information transmission characteristics are to a simple LIF model. However, despite the generality of the θ-neuron, it does suffer some important limitations. For one, the output does not map directly onto voltage changes in real neurons, so spike shape is not modeled (although spike timing is). More importantly, adaptation is not included in the θ-neuron.[20] Given the ubiquity of adaption in mammalian cortical cells, it is important that we consider models that both spike, and adapt. As well, on a more methodological note, the θ-neuron, and the methods used to reduce conductance-based models to it, are unfamiliar to most neuroscientists. In contrast, more direct reductions, like those found in the FitzHugh-Nagumo (FitzHugh 1961), Morris-Lecar (1981), and Rinzel (1985) models are likely to be more familiar. These models explicitly describe voltage dynamics, and thus produce true action potential traces, unlike the θ-neuron.

4.5.3 Adapting, conductance-based neuron

In this section, we consider the most realistic of the models we have seen so far; we call it the Wilson neuron (Wilson 1999b; Wilson 1999a). The Wilson neuron is a conductance-based model of the regular spiking neuron in mammalian neocortex. This model includes adaptation but, unlike in the adapting LIF model, the dynamics of the adaptation current are modeled directly after the calcium-dependent potassium current thought to be responsible for adaptation in class I neurons (Wilson 1999b). Like the θ-neuron, this model includes spiking dynamics. However, the reduction of this model from conductance-based models is very direct, resulting in a model that also captures the voltage dynamics observable in real neurons. As a result, this model captures features of neural spiking not addressed by the previous models. For example, changes in spike height with frequency, spike shapes, and after-hyperpolarization are captured by this model.

Let us briefly consider the reduction of complex conductance models to the Wilson neuron to show its relation to more empirically generated models (see Wilson 1999b; Wilson 1999a). To begin, consider the famous Hodgkin-Huxley (1952) model whose parameters were derived directly from experimental observations of the giant squid axon:

$$C\frac{dV}{dt} = -g_{Na}m^3h(V-E_{Na}) - g_K n^4(V-E_K) - g(V-E) + J_M \quad (4.30)$$

$$\frac{dm}{dt} = \frac{1}{\tau_m(V)}(-m + M(V)) \quad (4.31)$$

$$\frac{dh}{dt} = \frac{1}{\tau_h(V)}(-h + H(V)) \quad (4.32)$$

$$\frac{dn}{dt} = \frac{1}{\tau_n(V)}(-n + N(V)). \quad (4.33)$$

The circuit for this model is identical to figure 4.20, with the A-current removed. The parameters g_{Na}, g_K, and g are the sodium, potassium and membrane leakage conductances ($\frac{1}{R}$), respectively. The parameters m, h, and n are the sodium activation,

[20] Although it could be included, by adding a second dimension to the model analogous the slow A-current.

sodium inactivation, and potassium activation parameters, respectively. These parameters model the dynamics of the opening and closing of ion channels in the cell membrane. Notably, the equilibrium values of these parameters (M, H, and N) and their respective time constants (τ_m, τ_h, and τ_n) are functions of the membrane potential, V. Finally, the equilibrium potentials (shown as batteries in figure 4.20), E_{Na}, E_K, and E are the potentials for which the net ionic current across the membrane is zero (these are largely due to the ion concentration gradients across the membrane). Thus, these equations capture a fourth-order nonlinear system. Analyses of such complex systems are extremely difficult.

Fortunately, Rinzel (1985) noticed two very useful simplifications. First, he pointed out that τ_m is extremely small, so (4.31) can be eliminated by approximating m as $M(V)$ (since the equilibrium, $M(V)$, is reached quickly). Second, he realized that sodium channels close at approximately the same rate, but in the opposite direction as the potassium channels. Thus, (4.32) can be eliminated by letting $h = 1 - n$. Notably, there is now a single 'recovery' variable that results from the amalgamation of h and n, call it R. These simplifications mean that an accurate approximation to the Hodgkin-Huxley equations can be found in a two-dimensional system.

As mentioned in the previous section, the introduction of the A-current accounts for the differences between the Hodgkin-Huxley and class I neurons. A direct introduction of this current makes the two-dimensional model a three-dimensional one. However, Rose and Hindmarsh (1989) showed that a good approximation to the effects of this current is found by making the equation for the recovery variable, R, quadratic.[21] Thus, a good approximation to a class I neuron can be achieved in a two-dimensional system.

In order to introduce adaptation into the model, we must add a slow potassium current (analogous to J_{adapt} in the adapting LIF neuron), governed by the conductance variable H. Using parameters found to produce good approximations in human neocortical neurons, we can write the final Wilson model as (after Wilson 1999a; Wilson 1999b, p. 157)

$$\begin{aligned} C\frac{dV}{dt} &= -\left(1781 + 4758V + 3380V^2\right)(V - 48) \\ &\quad -26R(V + 95) - 13H(V + 95) + J_M \qquad (4.34) \\ \frac{dR}{dt} &= \frac{1}{5.6}\left(-R + 129V + 79 + 330(V + 38)^2\right) \qquad (4.35) \\ \frac{dH}{dt} &= \frac{1}{99.0}(-H + 11(V + 75.4)(V + 69)). \qquad (4.36) \end{aligned}$$

Equation (4.34) incorporates the resting potentials of sodium and potassium ions at 48 and -95 mV. The quadratic in V and the constants multiplying R and H in (4.34) are found by fitting the nullclines of this system to the full conductance model. In (4.35), the contributions of the standard potassium current (linear) and the A-current (quadratic) have been amalgamated, as suggested by Rose and Hindmarsh (1989). Finally, (4.36) includes a term with the resting potential, -75.4 mV, so as to ensure the current has no effect at rest. Notably, the time constant of this current is very long (99.0 ms) so as to produce an appropriate adaptation without affecting the shape of the spike.

[21] See Rush and Rinzel 1994 for reservations regarding this interpretation of the A-current.

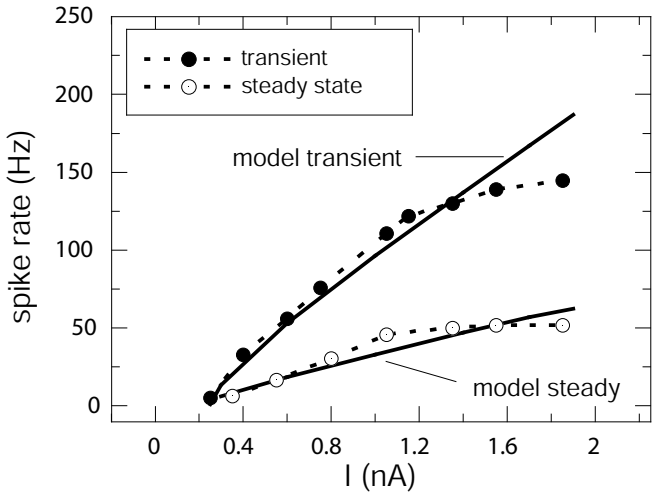

Figure 4.23: Wilson's conductance-based model compared to real data (from Avoli et al. 1994 © Springer-Verlag as in Wilson 1999b). Both transient and steady state (i.e., after adaptation) properties of a human regular spiking cortical neuron are reproduced well, and over a range of inputs.

As can be seen from figure 4.23, these equations do an excellent job of approximating the behavior of real regular-spiking neurons.

As shown in figure 4.24, decoding the spike train of this model again works well. The neurons used here are again matched to the original LIF model for background and maximal firing rates. In this case, the information transmission rate (91 bits/s) and RMS error are about 20% worse than for the LIF model. However, the efficiency is improved (2 bits/spike, a 30% increase). Given the results of the adapting LIF model this is likely due to inclusion of the adapting current. Again, by far the greatest difference between this model and the LIF model is that it takes approximately 600 times longer to run the same problem.

4.5.4 Discussion

In this section, we have progressed from the simple LIF model to a conductance-based, adapting model that is known to capture a wide-variety of detailed biophysical results (Wilson 1999b; Wilson 1999a). What we have shown through this discussion, is that the information transmission characteristics of a variety of single-cell models are not only very similar to one another, but also to neurons found in real neurobiological systems. Table 4.1 summarizes the results from the various models we have explored. As this table shows, there is not a very large spread in either the information transmission rates or efficiencies, although the adapting neurons are consistently more efficient. It

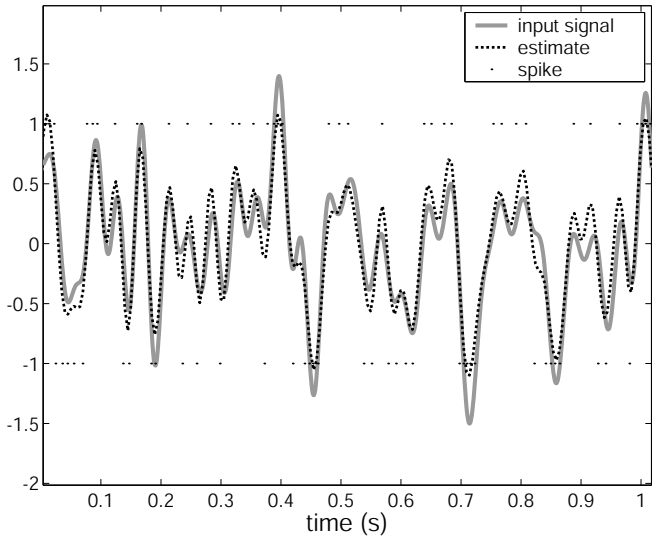

Figure 4.24: Signal reconstruction using an optimal filter and the conductance-based model as the encoder (RMSE = 0.186).

is also important to note that the efficiencies of the models lies comfortably within the range of efficiencies reported for actual neurobiological systems, between about 1 and 3 bits/spike (see section 4.4.2). By far the greatest differences between the models lies in their run times. All in all, this tables shows that the simple LIF model is a good trade-off between a realistic neural encoder and a computationally tractable model. As models become large, computational tractability becomes extremely important. Because our focus in this book is on large-scale modeling, we adopt the LIF as our neural model of choice.[22]

4.6 Summary

In this chapter, we have been largely concerned with introducing and formally characterizing neural spiking, the most salient neural nonlinearity. We began with a study of the leaky integrate-and-fire (LIF) model, which is simple, yet incorporates this central nonlinearity. We then turned to a general discussion of temporal coding in neurons and argued that 1) it is not necessary to understand neural coding in terms of a rate code/timing code dichotomy, and 2) considering pairs of neurons as fundamental to temporal coding is highly productive. We then described a means of finding optimal temporal decoders for pairs of neurons. This analysis is very similar to those that have been done on neurons in the past, with some slight improvements (i.e., Gaussian windowing).

[22]However, the software package associated with this book allows the user to pick any of these models when constructing a simulation.

Neuron	Rate	Bits/spike	RMSE	Run time (s)
LIF	114	1.24	0.153	0.18
Adapting LIF	114	2.23	0.153	0.24
θ-Neuron	109	0.96	0.160	20.1
Wilson Model	91	2.00	0.186	125.2

Table 4.1: Comparison of information transmission characteristics of various model neurons.

We employed this analysis on the LIF neuron to show that these methods effectively bridge the gap between timing and rate codes, allowing us to remain agnostic as to what 'the' code of neural systems is. As well, we showed that the simple LIF neuron has approximately the same information transfer characteristics as both real neurons and more complex neuron models. Our examination of these more complex models both showed how they could be included in this general framework and demonstrated that the LIF model strikes a convenient balance between neural plausibility and computational cost.

In addition, we addressed the issue of the biophysical relevance of the optimal decoders. We showed that the post-synaptic currents (PSCs) could be used in place of the optimal decoders with little loss in information transfer. The gain in neural plausibility is, of course, enormous. As a result of these considerations, all of our subsequent models use the LIF model and PSC temporal filtering.