Spatio-temporal control of network activity through gain modulation in cortical circuit models

Jake P. Stroud

Wadham College
University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity term 2018

To Mary Harrison, who loved stories, but sadly
only heard the start of this one.

Acknowledgements

I would like to thank the Doctoral Training Centre for giving me the opportunity to study for a DPhil at Oxford University and the Engineering and Physical Sciences Research Council for funding my research. I also thank Wadham College for providing an amazing college environment and enabling me to thrive during my time at Oxford.

Many people have provided invaluable scientific guidance to me during my DPhil. Firstly, I express deep gratitude to my three supervisors: Tim P. Vogels, Mason A. Porter, and Guillaume Hennequin. Tim encouraged me to freely explore research questions and to push ideas as far as possible until we had exhausted all potential avenues. He guided me on how to be both meticulous and creative when designing figures and how to tell an engaging story with my scientific results. Mason taught me new levels of precision in both my mathematical and non-mathematical exposition, and Guillaume taught me to always question my results and revealed to me just how much I have left to learn.

I also thank everyone in the Vogels lab. I enjoyed every minute of being there; from lab meetings to conferences, New College lunches to lab retreats, or just chatting over a cup of tea in the office. I looked forward to coming into the lab every day and I will be sad to leave. In particular, I thank Georgia Christodoulou, Rui P. Costa, Yayoi Teramoto Kimura, and Friedemann Zenke. I am also especially grateful to Everton J. Agnes and William F. Podlaski who were unfortunate enough to endure sitting next to me for over 3 years. Both have contributed greatly to my scientific thinking.

I am also in great debt to the love and support from my friends and family. My parents have always supported my ambitions, no matter how big or small. In particular, my mum stopped at nothing to give me the great education that I have been lucky enough to have. Finally, I thank Kathi for being a wonderful and constant source of joy and support.

Abstract

Animals perform an extraordinary variety of movements over many different time scales. To support this diversity, the motor cortex (M1) exhibits a similarly rich repertoire of activities. Although recent neuronal-network models capture many qualitative aspects of M1 dynamics, they can generate only a few distinct movements all with the same duration. Therefore, these models can still not explain how M1 efficiently controls movements over a wide range of shapes and speeds. In this thesis, we demonstrate that simple modulation of neuronal input–output gains in recurrent neuronal-network models with fixed connectivity can dramatically reorganise neuronal activity and thus downstream muscle outputs. Consistent with the observation of diffuse neuromodulatory projections to M1, our results suggest that a relatively small number of modulatory control units provide sufficient flexibility to adjust high-dimensional network activity on behaviourally relevant time scales. Such modulatory gain patterns can be obtained through a simple reward-based learning rule. Novel movements can also be assembled from previously learned primitives, thereby facilitating fast acquisition of hitherto untrained muscle outputs. Moreover, we show that it is possible to separately change movement speed while preserving movement shape, thus enabling efficient and independent movement control in space and time. Our results provide a new perspective on the role of neuromodulatory systems in controlling cortical activity and suggest that modulation of single-neuron responsiveness is an important aspect of learning.

Contents

We will never have a perfect world, and it would be dangerous to seek one. But there is no limit to the betterments we can attain if we continue to apply knowledge to enhance human flourishing.

— Steven Pinker

CHAPTER 1

Introduction

Most animals are endowed with a relatively dense collection of interconnected nerve cells (i.e., neurons) that we call a brain. This collection of neurons enables processing of sensory information, decision-making, and subsequent motor output commands. Such neural computations are normally more elaborate than merely sensing and reacting to the environment. For example, sponges can react to environmental changes even though they have no nervous system and bacteria can detect chemical gradients and navigate accordingly.

The complexity of brains and the functions they can perform vary widely across the animal kingdom. For example, the nematode worm (*C. elegans*) has only 302 neurons in its nervous system, yet it can remember and respond to many different odours and also exhibits social behaviours (de Bono and Villu Maricq, 2005). In contrast, an adult human brain contains on the order of $100{,}000{,}000{,}000$ neurons. Such a large network of neurons enables us to, among other things, perform complex skilled movements such as hitting a tennis ball consistently and accurately over a long period of time and in such a way that you increase the probability of your opponent losing the point (and hopefully the match!). In fact, it has been

proposed that the only reason that we have a brain is to produce adaptable and complex movements (Wolpert, 2011). Additionally, some strategy games, such as the board game 'Go' (which many thought could only be played to a high level by humans because of their capacity for complex cognitive decision-making), can now be played at superhuman levels by computers using modern machine learning algorithms. However, it has proved incredibly challenging for machines to perform dexterous movements such as pouring water from a bottle into a glass — which we might consider is a substantially easier task than playing the world's best Go player.

Even though technology and fields such as machine learning have advanced significantly over recent decades, we are still unable to accurately and thoroughly reproduce the activity of the nervous system of even *C. elegans*, which is one of the simplest nervous systems in the animal kingdom (see www.openworm.org for current progress). Although the entire connectome of *C. elegans* has been mapped (White et al., 1986), the approximate $3{,}000$ connections (called *synapses*), enable complex dynamical interactions between the neurons and it is difficult to understand how stimulating one region affects another and ultimately how the animal's behaviour is impacted.

This difficulty arises in part because there are many complex phenomena involved in the interactions between neurons. We now give a very brief overview of such phenomena. Neurons are effectively small electrical input–output units and they primarily communicate with each other by sending voltage spikes called 'action potentials'. When a neuron receives a sufficient input current caused by incoming spikes arriving from other neurons, the neuron's membrane voltage suddenly spikes itself and then ultimately returns to a 'resting' voltage. This voltage spike then propagates down the neuron's axon to all synapses with which the neuron is a presynaptic partner. At most synapses, the presynaptic arrival of an action potential triggers the release of certain chemicals called neurotransmitters. These chem-

icals move across the synapse to the postsynaptic neuron and bind with receptor molecules. These receptors control the flow of small charged particles (called ions) across the membrane of the postsynaptic neuron. Certain combinations of neurotransmitters and ions will cause either the postsynaptic neuron's membrane voltage to increase (called excitation) or decrease (called inhibition). Depending on the timing of arriving spikes, the type of ions and neurotransmitters involved, the location on the cell of the arriving spikes, the presence of neuromodulators (e.g., dopamine), the type of neurons interacting and many other phenomena, one can obtain a large range of different neuronal activities. Additionally, most synaptic connection strengths can change over time in what is called *synaptic plasticity*. For example, if two connected neurons are active at the same time, one typically observes an increase in their connection strength (Bliss and Collingridge, 1993; Cruikshank and Weinberger, 1996; Hebb, 1949).

This is just a small glimpse into the complexity of neuronal circuits. To aid our understanding of this complexity, over approximately the last 50 years, researchers in the field of computational neuroscience have built models that help to explain existing experimental data and can also generate new predictions for future experiments. Such models typically involve sets of differential equations which can be studied either analytically or by running simulations on a computer.

This thesis builds on such computational approaches for studying the dynamics of networks of neurons. We focus in particular on neuronal-network models that exhibit dynamics reminiscent of cortical circuits involved in motor control (e.g., primary motor cortex). In Section 1.1, we introduce how one can model the firing rate of a single neuron receiving multiple presynaptic inputs. In Section 1.2, we show how one can model the firing rate of a recurrently connected network of neurons and we introduce several important aspects of such models that are relevant for this thesis. In Section 1.3, we discuss cortical motor circuits and finally, in Section 1.4, we provide some motivation of the main topic of research in this thesis.

1.1 Modelling neuronal firing-rate dynamics

In this thesis, we use relatively abstracted models of neuronal activity in which we model neuronal firing-rates (we call these 'rate models' in the following discussion). We do this for several reasons. One reason is that such simplified models can enable greater analytical tractability; this becomes particularly beneficial when studying networks of model neurons (which we discuss in the next paragraph). Although many methods for analysing the dynamics of networks of spiking neurons have been developed over the last few decades (Dayan and Abbott, 2001; Gerstner and Kistler, 2002; Gerstner et al., 2014), many of these methods rely on mean-field analyses, which themselves depend upon several assumptions (such as assuming a large number of neurons or random connectivities). For the rate models that we study in this thesis, many useful methods from dynamical systems (Strogatz, 2014; Teschl, 2012; Wiggins, 2003) and linear algebra (Strang, 2016) can generate precise predictions of the behaviour of such simplified neural models. There is the additional possibility that some of the conclusions that we reach regarding the simplified rate models may also extend to more biologically realistic spiking models.

Another reason for using rate models is that many of the key experimental results regarding cortical motor circuit dynamics, which is the main topic of this thesis, pertain to low-dimensional activity patterns of motor cortex (Afshar et al., 2011; Churchland et al., 2012; Gallego et al., 2017; Shenoy et al., 2013) and fundamental computational mechanisms operating at the neural population level (Russo et al., 2018). Furthermore, there is growing experimental evidence that motor cortex operates as an autonomous (i.e., time-invariant) dynamical system (Churchland and Cunningham, 2014; Churchland et al., 2010, 2012; Kao et al., 2015) where pre-movement activity sets the initial condition for the neural system whose subsequent evolution produces desired muscle activity. (At least for fast ballistic movements

where proprioceptive feedback does not affect the movement.) That is, many experimental results suggest that the population activity of cortical motor circuits appears more relevant for understanding how the brain performs motor tasks than single-neuron firing rates or even single spike events. Therefore, modelling motor cortex as an autonomous dynamical system and using methods from dynamical-systems theory to study the behaviour of such a system seems natural. In other words, we may not be sacrificing much in terms of relevant biological phenomena by using such a simplified model. In fact, it is more likely that much will be gained from taking such a perspective, particularly given the recent experimental findings mentioned above.

Before we introduce the type of models that we primarily use in this thesis, we first present how one can model the firing rate of a single postsynaptic neuron that receives multiple independent inputs from N presynaptic neurons. We follow the derivation given in Dayan and Abbott (2001), but many similar derivations exist (e.g., see Gerstner et al. (2014); Miller and Fumarola (2012); Wilson and Cowan (1972)).

To understand how a neuron's firing rate depends on its synaptic inputs, we must first determine how a neuron's total synaptic current (which we denote by I_s) depends on such synaptic inputs. Consider a neuron receiving input from a single presynaptic neuron i that has a synaptic connection 'weight' (i.e., the strength of the synaptic connection) w_i (see Fig. 1.1a). Although there is a rich literature regarding the dynamics of how neurons influence one another's activities through complex synaptic connections (Abbott and Nelson, 2000; Dayan and Abbott, 2001; Gerstner et al., 2014; Montgomery and Madison, 2004; Redondo and Morris, 2011; Ziegler et al., 2015), firing-rate models typically omit describing such complex processes that occur at synapses. Instead, a synaptic connection is commonly represented by a single scalar weight value. The weight w_i can either be positive (for excitatory synapses) or negative (for inhibitory synapses). Biologically, neurons

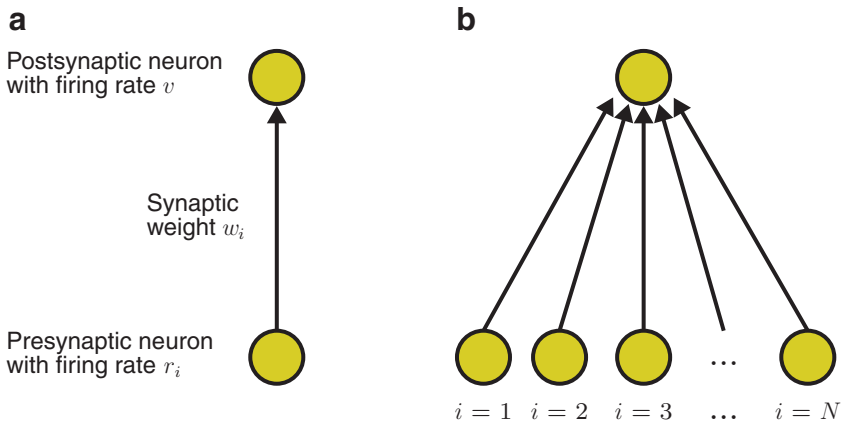

Figure 1.1: An illustration of one postsynaptic neuron receiving presynaptic inputs. **a**, *One postsynaptic neuron receiving one presynaptic input.* **b**, *One postsynaptic neuron receiving multiple independent presynaptic inputs from N other neurons.*

either have only positive outgoing weights or only negative outgoing weights; this is commonly called *Dale's law* (Strata and Harvey, 1999).

Getting back to our example — if an action potential (i.e., a 'spike') arrives at this synapse from the presynaptic neuron at time $t = 0$, the synaptic current at time $t \geq 0$ generated at the soma of the postsynaptic neuron is commonly written as $w_i K(t)$, where $K(t) = e^{-t/\tau}/\tau$ is the *synaptic kernel* (Dayan and Abbott, 2001). The synaptic kernel $K(t)$ describes the temporally evolving synaptic current at the soma in response to a presynaptic spike arriving at time $t = 0$. There are many anatomical and molecular phenomena that affect this temporal response. However, for simplicity — and following a large body of previous literature (Dayan and Abbott, 2001; Gerstner et al., 2014) — we assume such a functional form for K and we also assume the same form for all synapses. This allows greater analytical tractability and still provides accurate descriptions of the current at the soma due to synaptic inputs under many conditions (Dayan and Abbott, 2001; Gerstner et al., 2014).

We now consider a sequence of multiple spikes that arrive at this synapse at times t_j. Assuming that the current generated by independent spike arrivals at the

synapse sum linearly, the total synaptic current at time t arising from this sequence of presynaptic spikes is given by

$$w_i \sum_{t_j<t} K(t-t_j) = w_i \int_{-\infty}^{t} K(t-s)\rho_i(s)ds\,, \tag{1.1}$$

where $\rho_i(s) = \sum_j \delta(s-t_j)$ is the *neural response function*, which describes the spike sequence as a sum of Dirac δ functions (Dayan and Abbott, 2001).

For a firing-rate model, the next step is to replace the neural response function $\rho_i(s)$ by the firing rate $r_i(s)$ of the presynaptic neuron, so that the postsynaptic input is now given by

$$w_i \int_{-\infty}^{t} K(t-s)r_i(s)ds\,. \tag{1.2}$$

If this is the only input to the postsynaptic neuron, the total synaptic current I_s is described by Eqn. (1.2). However, if this neuron receives multiple synaptic inputs from synapses $i = 1,\ldots,N$ (see Fig. 1.1b), then, assuming that the effect of currents arriving from multiple synapses sum linearly, we can sum over them and their associated presynaptic firing-rate activities by writing

$$I_s = \sum_{i=1}^{N} w_i \int_{-\infty}^{t} K(t-s)r_i(s)ds\,. \tag{1.3}$$

To describe the temporally evolving synaptic current I_s, we take the time derivative of Eqn. (1.3) to obtain

$$\begin{aligned}
\frac{dI_s}{dt} &= \frac{d}{dt}\sum_{i=1}^{N} w_i \int_{-\infty}^{t} K(t-s)r_i(s)ds \\
&= \frac{1}{\tau}\sum_{i=1}^{N} w_i \frac{d}{dt}\int_{-\infty}^{t} e^{-\frac{(t-s)}{\tau}} r_i(s)ds \\
&= \frac{1}{\tau}\sum_{i=1}^{N} w_i \left(\frac{dt}{dt}\left[e^{\frac{(s-t)}{\tau}} r_i(s)\right]\Big|_{s=t} - \frac{d(-\infty)}{dt}\left[e^{\frac{(s-t)}{\tau}} r_i(s)\right]\Big|_{s=-\infty} + \int_{-\infty}^{t} \frac{\partial e^{\frac{(s-t)}{\tau}}}{\partial t} r_i(s)ds \right) \\
&= \frac{1}{\tau}\sum_{i=1}^{N} w_i \left(r(t) - \int_{-\infty}^{t} K(t-s)r_i(s)ds \right) \\
&= \frac{1}{\tau}\left(\sum_{i=1}^{N} w_i r_i(t) - I_s \right)\,,
\end{aligned} \tag{1.4}$$

where we used the Leibniz integral rule to go from the second to the third line and we used Eqn. (1.3) to get to the final line. We therefore obtain

$$\tau \frac{dI_s}{dt} = -I_s + \sum_{i=1}^{N} w_i r_i(t)\,, \tag{1.5}$$

with some initial condition $I_s(t=0) = I_0$.

The final step we need is to model how the firing rate of the postsynaptic neuron depends on the total synaptic current I_s at the soma. For a constant synaptic current, the firing rate v of the postsynaptic neuron can be expressed as $v = f(I_s)$, where f is called the input–output *gain function* or f-I curve (Dayan and Abbott, 2001). This function describes how the steady-state firing rate depends upon the synaptic current and is often taken to be a sigmoid or a rectified linear function (Dayan and Abbott, 2001; Gerstner et al., 2014). See Section 1.2.1 for a more detailed discussion of choosing such a function.

For currents that change with time, a common assumption is that $v(t) = f(I_s(t))$ still approximately holds with a time-dependent current $I_s(t)$. This leads to the following equation for how the firing rate $v(t)$ of a single postsynaptic neuron depends on the firing rates r_i of its N presynaptic partners:

$$\tau \frac{dI_s(t)}{dt} = -I_s(t) + \sum_{i=1}^{N} w_i r_i(t)\,, \text{ with } v(t) = f(I_s(t))\,, \tag{1.6}$$

with some initial condition $I_s(t=0) = I_0$.

In Section 1.2, we slightly adapt Eqn. (1.6) to model the firing rates of a population of recurrently connected neurons.

1.2 Modelling recurrent neuronal networks

In this thesis, we focus on studying recurrent networks of model neurons (i.e., networks in which any neuron can connect to any other neuron). In Section 1.1, we obtained an equation that describes the firing rate of a single neuron receiving feedforward inputs (see Eqn. (1.6) and Fig. 1.1b). We now slightly adapt Eqn. (1.6)

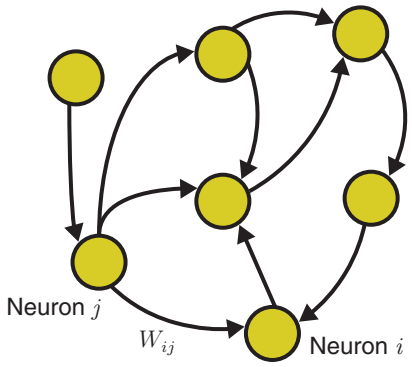

Figure 1.2: An illustration of a recurrently connected network of neurons. Any neuron can connect to any other neuron.

so that we can describe the dynamics of a population of N neurons in a recurrent network. We assume that the connection between any two neurons in a recurrent network is encoded by a connectivity matrix $\boldsymbol{W}$ where element W_{ij} is the connection strength from neuron j to neuron i (see Fig. 1.2). Following Eqn. (1.6), the input x_i to neuron i in a recurrent network is governed by the following differential equation

$$\tau \frac{dx_i}{dt} = -x_i + \sum_{j=1}^{N} W_{ij} f(x_j)\,, \tag{1.7}$$

where the firing rate of neuron i is $f(x_i)$ and the initial condition is $\boldsymbol{x}(t=0) = \boldsymbol{x}_0$, where $\boldsymbol{x}(t) = (x_1(t), \ldots, x_N(t))^\mathsf{T}$. It will sometimes be more convenient to write Eqn. (1.7) using vector and matrix notation:

$$\tau \frac{d\boldsymbol{x}}{dt} = -\boldsymbol{x} + \boldsymbol{W} f(\boldsymbol{x})\,. \tag{1.8}$$

The dynamical system in Eqn. (1.8) can give rise to rich dynamics that resemble experimentally measured neuronal activity in many different brain regions (Breakspear, 2017; Hennequin et al., 2014; Mante et al., 2013; Sussillo et al., 2015; Wang et al., 2018). Clearly, depending on our choices of the synaptic weight matrix $\boldsymbol{W}$ and the input–output gain function f, which converts neuronal activity into firing rates, we can obtain a multitude of different dynamics. For the remainder of Section 1.2, we provide a brief discussion of some typical choices of f and $\boldsymbol{W}$ and the effects that they can have on the neuronal dynamics.

1.2.1 The input–output gain function f

The input–output gain function f in Eqn. (1.8) is central to this thesis. Therefore, in this section, we provide a discussion regarding particular aspects of the function f that will be important for later sections.

1.2.1.1 A linear input–output gain function

If the input–output function f in Eqn. (1.8) is linear (e.g., $f(\boldsymbol{x}) = \boldsymbol{x}$), we obtain

$$\tau \frac{d\boldsymbol{x}}{dt} = -\boldsymbol{x} + \boldsymbol{W}\boldsymbol{x}\,, \tag{1.9}$$

and the system in Eqn. (1.9) can be solved exactly (Strang, 2016; Teschl, 2012). The solution of Eqn. (1.9) is

$$\boldsymbol{x}(t) = e^{\frac{t}{\tau}(\boldsymbol{W}-\boldsymbol{I})}\boldsymbol{x}_0\,, \tag{1.10}$$

where $\boldsymbol{I}$ is the identity matrix, $e^{\boldsymbol{A}}$ denotes the matrix exponential of a matrix $\boldsymbol{A}$ (see (Teschl, 2012, Section 3.1) for an excellent introduction to the matrix exponential), and $\boldsymbol{x}_0$ is the initial condition. In practice, however, computing this solution can produce substantial numerical errors (Higham, 2005), particularly for networks containing many neurons. To minimise numerical errors, one must employ a method such as 'scaling and squaring' when using a Taylor series expansion to compute Eqn. (1.10) (Higham, 2005). Alternatively, one can construct the solution in Eqn. (1.10) in terms of the eigenvalues and eigenvectors of the matrix $\boldsymbol{W}$. However, we must also use a few 'tricks' from linear algebra and complex analysis to ensure that we ultimately obtain a real-valued solution (see Appendix B).

1.2.1.2 Stability of a linear dynamical system

An important concept in this thesis is the *stability* of neuronal activity on a network. We use the standard mathematical definition of the stability of a linear dynamical

system, in that the linear system in Eqn. (1.9) is called *stable* if all solutions remain bounded as $t \to \infty$ (Teschl, 2012). Furthermore, a linear system is called *asymptotically stable* if all solutions converge to 0 as $t \to \infty$.

Using mathematical definitions of the stability of linear dynamical systems, the system in Eqn. (1.9) is asymptotically stable if and only if the real part of all of the eigenvalues of $\boldsymbol{W} - \boldsymbol{I}$ are less than 0 (Teschl, 2012, Section 3.2), or equivalently if the real part of all of the eigenvalues of $\boldsymbol{W}$ are less than 1. This is because the addition of a diagonal matrix only shifts the real part of the spectrum. (See Appendix A for a proof.) We will denote the condition of the real part of all of the eigenvalues of $\boldsymbol{W}$ being less than 1 as $\alpha(\boldsymbol{W}) < 1$, where

$$\alpha(\boldsymbol{W}) = \max\{\operatorname{Re}\lambda : \lambda \text{ is an eigenvalue of } \boldsymbol{W}\}. \tag{1.11}$$

The quantity $\alpha(\boldsymbol{W})$ is called the *spectral abscissa* of the matrix $\boldsymbol{W}$ (Hennequin et al., 2014; Vanbiervliet et al., 2009). Thus, from the above discussion, for the linear recurrent neuronal network Eqn. (1.9), the neural activity $\boldsymbol{x}(t)$ converges to $\boldsymbol{0}$ over (positive) time if and only if $\alpha(\boldsymbol{W}) < 1$.

How useful is the linear network model in Eqn. (1.9) in describing cortical dynamics? Some biological realism is neglected, because neuronal firing rates can become negative in this model. (We will revisit this issue in Section 1.2.1.4.) Additionally, the solution Eqn. (1.10) either decays to $\boldsymbol{0}$ or explodes, depending on the sign of the largest real part of the spectrum of $\boldsymbol{W}$ (except for the special case of $\alpha(\boldsymbol{W}) = 1$; see also Section 1.2.3) (Teschl, 2012). In contrast, real cortical circuits can exhibit complex fluctuations in activity, as well as sustained levels of firing. In Section 1.2.1.3, we discuss how one can overcome some of these difficulties by using a nonlinear gain function.

1.2.1.3 A nonlinear input–output gain function

As mentioned in Section 1.2.1, the gain function f, which resembles the single-neuron input–output (f-I) curve (Thurley et al., 2008), converts neuronal activity

$\boldsymbol{x}$ into firing rates. When experimentally accessing the single-neuron input–output gain function *in vitro* by injecting current into the soma of a neuron and measuring the resulting firing rate, one commonly finds a sigmoidal-shaped input–output curve (Chance et al., 2002; Rothman et al., 2009; Thurley et al., 2008). In line with such observations, one also typically chooses a sigmoidal-shaped function f, such as a $\tanh$ function, in the model in Eqn. (1.8) (Rajan et al., 2010; Sompolinsky et al., 1988). In this thesis, we primarily use the following functional form for f:

$$f(x_i) = \begin{cases} r_0 \tanh(g_i x_i / r_0)\,, & \text{if } x_i < 0\,, \\ (r_{\max} - r_0) \tanh(g_i x_i / (r_{\max} - r_0))\,, & \text{if } x_i \geq 0\,, \end{cases} \tag{1.12}$$

where $r_{\max}$ is the maximum firing rate, and the 'gain' value g_i is the slope of the function f for neuron i at baseline rate r_0 and thus controls the input–output sensitivity of neuron i. The gain g_i of each neuron is conventionally set to 1 in previous models (Hennequin et al., 2014; Hoerzer et al., 2014; Rajan et al., 2010), and the baseline and maximum rates r_0 and $r_{\max}$, respectively, are chosen to be biologically realistic depending upon the cortical area being modelled. See Section 2.2 where we discuss some typical choices of r_0 and $r_{\max}$.

With this setup (i.e., Eqns. (1.8) and (1.12)), $f(x_i)$ describes the firing rate of neuron i relative to the baseline rate r_0. In Fig. 1.3, we plot the gain function $f(x_i)$ from Eqn. (1.12) using gain values of (black) 1 and (blue) 2, together with the linearised version of the gain function (i.e., $f_{\mathrm{l}}(x_i) = g_i x_i$) (see the dashed lines).

The function in Eqn. (1.12) ensures that the solution of Eqn. (1.8) is always bounded. However, guaranteed boundedness of the neuronal dynamics is not the same as stability of the equilibrium point at $\boldsymbol{x} = \boldsymbol{0}$. To study the (local) stability of the equilibrium point $\boldsymbol{x} = \boldsymbol{0}$ in Eqn. (1.8) when one uses the nonlinear function Eqn. (1.12), we must linearise Eqn. (1.8) around $\boldsymbol{x} = \boldsymbol{0}$ (Teschl, 2012). See Chapter 2 for a detailed discussion.

It has been shown that the system in Eqn. (1.8) together with a $\tanh$ gain function, such as the one in Eqn. (1.12), has a positive maximum Lyapunov exponent if

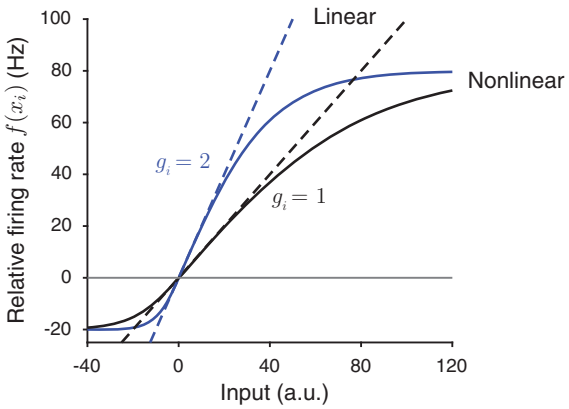

Figure 1.3: Examples of a $\tanh$ *gain function and the corresponding linearised versions. We show the gain function* f *from Eqn.* (1.12) *(solid curves) with (black)* $g_i = 1$ *and (blue)* $g_i = 2$*. We also show the linearised versions (i.e.,* $f_1(x_i) = g_i x_i$*) using the dashed lines with the same two gain values.*

$\alpha(\boldsymbol{W}) > 1$ (Sompolinsky et al., 1988). Because of this, if $\alpha(\boldsymbol{W}) > 1$, the neuronal dynamics governed by Eqn. (1.8) are commonly called 'chaotic', and the dynamics rapidly fluctuate between maximum and minimum firing rates (Hoerzer et al., 2014; Rajan et al., 2010; Sompolinsky et al., 1988; Sussillo and Abbott, 2009; Vogels et al., 2005).

1.2.1.4 A strictly positive input–output gain function

Eqn. (1.8) together with the gain function Eqn. (1.12) describes how neuronal firing rates can be modelled relative to a baseline rate r_0. Although such a setup can receive criticism because neuronal firing rates appear to be negative and thus the model is biologically unrealistic (Dayan and Abbott, 2001), we show that one can obtain identical neuronal firing-rate activity by using a strictly positive gain function f and including a constant input $\boldsymbol{h}$ in Eqn. (1.8). Specifically, given a desired baseline firing rate r_0, one models the neuronal activity as

$$\tau \frac{d\boldsymbol{x}}{dt} = -\boldsymbol{x} + \boldsymbol{W} f(\boldsymbol{x}) + \boldsymbol{h}\,, \tag{1.13}$$

from some initial condition $\boldsymbol{x}_0$, with $h_i = -r_0 \sum_j W_{ij}$ and

$$f(x_i) = \begin{cases} r_0 \tanh(g_i x_i / r_0) + r_0\,, & \text{if } x_i < 0\,, \\ (r_{\max} - r_0) \tanh(g_i x_i / (r_{\max} - r_0)) + r_0\,, & \text{if } x_i \geq 0\,, \end{cases} \tag{1.14}$$

where $r_{\max}$ is the maximum firing rate.

1.2.2 Network architecture

Thus far, we have not discussed how one chooses the network architecture (i.e., the connectivity between the neurons), which is encoded by $\boldsymbol{W}$. There is a rich literature discussing the choice of the connectivity matrix $\boldsymbol{W}$ and in particular the biological plausibility of $\boldsymbol{W}$ and how it affects the resulting neuronal dynamics (Dayan and Abbott, 2001; Gerstner et al., 2014; Hennequin et al., 2012, 2014; Rajan and Abbott, 2006; Rajan et al., 2010; Sompolinsky et al., 1988; Sussillo and Abbott, 2009; Vogels et al., 2005). One typically sets the probability that any two neurons are connected to be small, in line with experimentally observed sparse cortical connectivity (Braitenberg and Schüz, 1991; Lefort et al., 2009). Connection weights are typically chosen independently and identically from a probability distribution. A Gaussian distribution is commonly used with a mean of 0 and a standard deviation of $w_0/\sqrt{N}$ where N is the number of neurons in the network and w_0 is a constant (Rajan et al., 2010; Sompolinsky et al., 1988; Sussillo and Abbott, 2009). This leads to complex eigenvalues of $\boldsymbol{W}$ that, in the limit of large N, are distributed uniformly within a circle centred at $[0, 0]$ with a radius equal to w_0 (Girko, 1983) (see Fig. 1.4). Thus, for linear neuronal dynamics governed by Eqn. (1.9), with $w_0 < 1$, the neuronal dynamics will likely decay to $\mathbf{0}$ over time because the real parts of the eigenvalues of $\boldsymbol{W}$ are likely to be less than 1. In practice, there is a chance that the real parts of some eigenvalues are larger than 1 if w_0 is close to 1 because this result holds only in the limit of large N. For example, see the the rightmost eigenvalue in Fig. 1.4.

If one chooses connection weights as described above from a Gaussian distribution, Dale's law — which states that all outgoing synaptic weights from a neuron are either excitatory (i.e., positive) or inhibitory (i.e., negative) (Strata and Harvey, 1999) — is violated. However, this biological implausibility can be resolved by suggesting that the units in the network do not represent individual neurons. Rather, the units can be interpreted as representing the activity of a population of

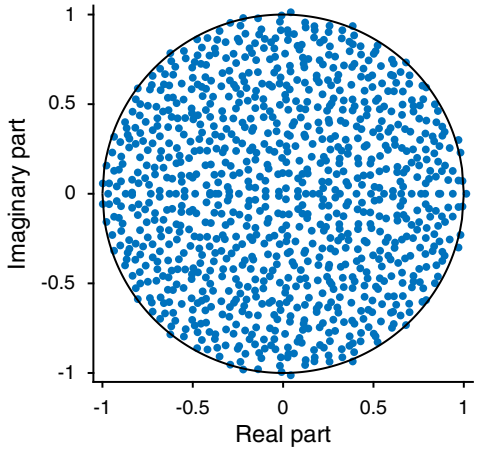

Figure 1.4: Spectrum of a connectivity matrix $\boldsymbol{W}$ of 1000 neurons with elements drawn independently and identically from a Gaussian distribution with a mean of 0 and a standard deviation of $1/\sqrt{N}$. We also show the unit circle with the black curve.

(excitatory and inhibitory) neurons. For example, if we consider the case of one population of neurons that connect to another population of neurons, depending on whether there is more net excitation or inhibition from the first population to the other, the overall effect will be either excitatory or inhibitory. Under this interpretation, units (i.e., populations of excitatory and inhibitory neurons) in model neuronal networks can display both positive and negative outgoing connections.

Alternatively, one can comply with Dale's law by enforcing columns of the weight matrix $\boldsymbol{W}$ to contain either all nonnegative or all nonpositive elements (Hennequin et al., 2014; Murphy and Miller, 2009; Rajan and Abbott, 2006). For an excitatory population E and an inhibitory population I, one commonly constructs the weight matrix $\boldsymbol{W}$ as

$$\boldsymbol{W} = \begin{pmatrix} \boldsymbol{W}_{EE} & -\boldsymbol{W}_{EI} \\ \boldsymbol{W}_{IE} & -\boldsymbol{W}_{II} \end{pmatrix}, \tag{1.15}$$

where the elements of $\boldsymbol{W}_{XY}$ are nonnegative and each element represents the connection strength from a unit in population Y to a unit in population X. The elements of $\boldsymbol{W}_{XY}$ are typically set to $w_0/\sqrt{N}$ with probability p and 0 with probability $1-p$ for any two neural populations $X \in \{E, I\}$ and $Y \in \{E, I\}$. Therefore, the left columns of $\boldsymbol{W}$ are nonnegative, and the right columns are nonpositive. The matrix $\boldsymbol{W}$ is thus likely mathematically non-normal (i.e., $\boldsymbol{W}^\mathsf{T}\boldsymbol{W} \neq \boldsymbol{W}\boldsymbol{W}^\mathsf{T}$). In Section 1.2.3, we discuss the effects on the neuronal dynamics of using a weight matrix of

the form in Eqn. (1.15).

1.2.3 Non-normal amplification

For a normal weight matrix $\boldsymbol{W}$ (i.e., $\boldsymbol{W}^{\mathsf{T}}\boldsymbol{W} = \boldsymbol{W}\boldsymbol{W}^{\mathsf{T}}$) that is asymptotically stable with respect to Eqn. (1.9) (i.e., the real part of all of the eigenvalues of $\boldsymbol{W}$ are less than 1, as discussed in Section 1.2.1.1), the resulting linear neuronal dynamics governed by Eqn. (1.9) decay exponentially to $\mathbf{0}$ following any initial condition (Hennequin, 2013; Hennequin et al., 2012; Trefethen and Embree, 2005). The closer the real parts of the eigenvalues of $\boldsymbol{W}$ are to 1 (but with all real parts of the eigenvalues remaining below 1), the more time it takes for the resulting neuronal dynamics to decay to $\mathbf{0}$. However, the linear dynamics governed by Eqn. (1.9) will always decay exponentially to $\mathbf{0}$ for a normal weight matrix — no matter how close the real part of its eigenvalues are to 1 (but always remaining strictly below 1).

Conversely, for a non-normal, (biologically-inspired) weight matrix $\boldsymbol{W}$ given by Eqn. (1.15) that is linearly stable and which contains strong excitatory connections balanced by strong inhibitory connections, one can obtain linear neuronal dynamics given by Eqn. (1.9) that transiently deviate away from 0 for some neurons following an initial condition (see Fig. 1.5) (Hennequin et al., 2012; Murphy and Miller, 2009). Such a transient deviation in neuronal activity is known as *non-normal balanced amplification* (Murphy and Miller, 2009), and it emerges because $\boldsymbol{W}$ is non-normal and thus its eigenvectors are not mutually orthogonal (Trefethen and Embree, 2005). The activity in the non-orthogonal eigenvector basis of $\boldsymbol{W}$ decays monotonically, but the activity of the neurons themselves can transiently amplify (Trefethen and Embree, 2005). The amplification can only be transient due to the linear stability of the weight matrix. Such transiently amplified neuronal activity has been suggested to be similar to the experimentally observed fast, transiently amplified neuronal responses observed in sensory cortices such as V1 (Murphy and Miller, 2009).

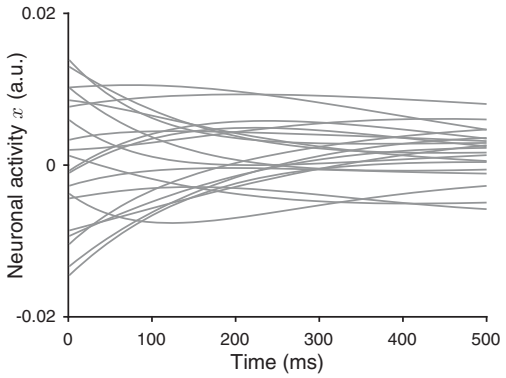

Figure 1.5: Non-normal amplification. We plot the activity of 20 *excitatory neurons in a recurrent neuronal network of* $N = 200$ *neurons where the neuronal activity is governed by Eqn.* (1.9) *with* $\tau = 200$ *ms and with* $\boldsymbol{W}$ *chosen according to Eqn.* (1.15) *with the elements of* $\boldsymbol{W}_{XY}$ *set to either* $3/\sqrt{N}$ *or* 0 *with probabilities* 0.1 *and* 0.9*, respectively, for any two neuronal populations* $X \in \{E, I\}$ *and* $Y \in \{E, I\}$*. The initial condition* $\boldsymbol{x}_0$ *for the neuronal activity is chosen from a uniform distribution over the range* $[-1, 1]$ *and is then rescaled so that* $\|\boldsymbol{x}_0\|_2 = 1$*. We observe that each neuron's activity does not simply decay monotonically to* 0*; some of the neurons' activities transiently deviate away from* 0*.*

1.2.4 Stability-optimised circuits

Neural activity in motor cortex also displays amplified activity transients during movement execution (Churchland et al., 2012). However, the increase in neural activity is often so large that a slightly different model network architecture is required than the one that we used in Fig. 1.5. One needs to somehow balance strong recurrent connections, which are necessary for amplified neuronal activity, with real parts in the eigenvalues of the weight matrix which are all below 1, which is necessary for the dynamics to eventually return to baseline. One approach to this problem is to enforce strong excitatory connections and then optimise inhibitory connections to reduce the spectral abscissa (i.e., the largest real part in the spectrum) of the weight matrix. Networks generated in this way have been called 'stability-optimised circuits' (Hennequin et al., 2014).

To create a stability-optimised circuit, one starts with a weight matrix $\boldsymbol{W}$ of

$N = 2M$ neurons (of which M are excitatory and M are inhibitory) that is constructed similarly to Eqn. (1.15) with the probability p of connection between any two neurons set to $p = 0.1$. Nonzero elements of $\boldsymbol{W}$ are then set to $w_0/\sqrt{N}$ for excitatory connections and $-\gamma w_0/\sqrt{N}$ for inhibitory connections, where $w_0^2 = 2\rho^2/(p(1-p)(1+\gamma^2))$ and the inhibition/excitation ratio γ is set to $\gamma = 3$. This construction results in $\boldsymbol{W}$ having an approximately circular spectrum of radius $\rho = 10$ (see the blue dots in Fig. 1.6a) (Hennequin et al., 2014).

An optimisation algorithm (see Appendix C for a detailed discussion of this algorithm) is then used to change only the inhibitory connections so that the spectral abscissa of $\boldsymbol{W}$ is reduced below 1. The algorithm minimises a quantity known as the 'smoothed spectral abscissa' $\tilde{\alpha}(\boldsymbol{W})$, which is an upper bound of the spectral abscissa $\alpha(\boldsymbol{W})$, that — among other advantages — leads to tractable optimisation (see Appendix C for details) (Vanbiervliet et al., 2009). Briefly, inhibitory weights are iteratively updated to follow the negative gradient of $\tilde{\alpha}(\boldsymbol{W})$ with respect to $\boldsymbol{W}$ subject to three constraints. First, the inhibitory weights remain inhibitory (i.e., negative). Second, a constant ratio ($\gamma = 3$) of mean inhibitory to mean excitatory weights is maintained. This is achieved by multiplicatively rescaling the inhibitory 'blocks' $\boldsymbol{W}_{EI}$ and $\boldsymbol{W}_{II}$ by $-\gamma\overline{\boldsymbol{W}}_{EE}/\overline{\boldsymbol{W}}_{EI}$ and $-\gamma\overline{\boldsymbol{W}}_{IE}/\overline{\boldsymbol{W}}_{II}$, respectively, where

$$\boldsymbol{W} = \begin{pmatrix} \boldsymbol{W}_{EE} & \boldsymbol{W}_{EI} \\ \boldsymbol{W}_{IE} & \boldsymbol{W}_{II} \end{pmatrix}, \tag{1.16}$$

and $\overline{\boldsymbol{W}}_{XY}$ denotes the mean over all elements in the matrix $\boldsymbol{W}_{XY}$. Third, the density of inhibitory connections is restricted to be less than or equal to 0.4. This is enforced so that the density of inhibitory connections remains biologically realistically small. We achieve this by setting the smallest connection strengths to 0. (One also removes any self-loops; that is, $\boldsymbol{W}_{ii} = 0$ for each i because one can trivially shift the spectrum of $\boldsymbol{W}$ to the left by addition of a diagonal matrix with only nonpositive elements.) This constrained gradient descent reduces the spectral abscissa from 10 to approximately 0.15 and we find empirically that approximately

2000 iterations are required (Hennequin et al., 2014). For the precise algorithmic details, see Appendix C.

In Fig. 1.6a, we show the spectrum of a 200-neuron network before (blue) and after (black) stability optimisation. Using the weight matrix $\boldsymbol{W}$ prior to stability optimisation, the neuronal dynamics governed by Eqn. (1.8) and Eqn. (1.12) fluctuate rapidly between maximum and minimum firing rates (see Fig. 1.6b). Following stability optimisation, the dynamics transiently oscillate before decaying to baseline (see Fig. 1.6c). For both simulations, we chose the initial condition $\boldsymbol{x}_0$ uniformly at random from the interval $[-1, 1]$ and then rescale $\boldsymbol{x}_0$ so that $\|\boldsymbol{x}_0\|_2 = 1$. Note that the oscillations are much more prominent than those in Fig. 1.5 even though the spectral abscissa is approximately 0.15 in Fig. 1.6c compared with approximately 0.98 in Fig. 1.5.

Rather than simply choosing the initial condition uniformly at random, one can also find initial conditions that produce large deviations in neuronal activity away from baseline. To find these so-called 'preferred initial conditions' (Hennequin et al., 2014), we study the linear system Eqn. (1.9). For some unit-norm initial condition $\boldsymbol{a}$, we define the 'evoked energy' $\varepsilon(\boldsymbol{a})$ as

$$\varepsilon(\boldsymbol{a}) = \frac{2}{\tau} \int_0^\infty \|\boldsymbol{x}(t)\|_2^2 dt \,. \tag{1.17}$$

The factor $2/\tau$ acts to normalise the evoked energy so that $\varepsilon = 1$ for an unconnected network (i.e., $W_{ij} = 0$ for all i, j) given any (unit-norm) initial condition $\boldsymbol{a}$ (Hennequin et al., 2014).

We then define an 'optimal' initial condition $\boldsymbol{a}_1$ as an initial condition that maximises $\varepsilon(\boldsymbol{a}_1)$ from Eqn. (1.17). (Note that for a weight matrix $\boldsymbol{W}$ with $\alpha(\boldsymbol{W}) < 1$, the envelope of the linear dynamics (1.9) decays exponentially over time, thus ε is also finite.) For the linear dynamics (1.9), we can analytically derive a collection $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_N$ of N orthogonal initial conditions that each maximise the evoked energy ε in the subspace orthogonal to all previous optimal initial conditions. (See Appendix C.2 for further details.)

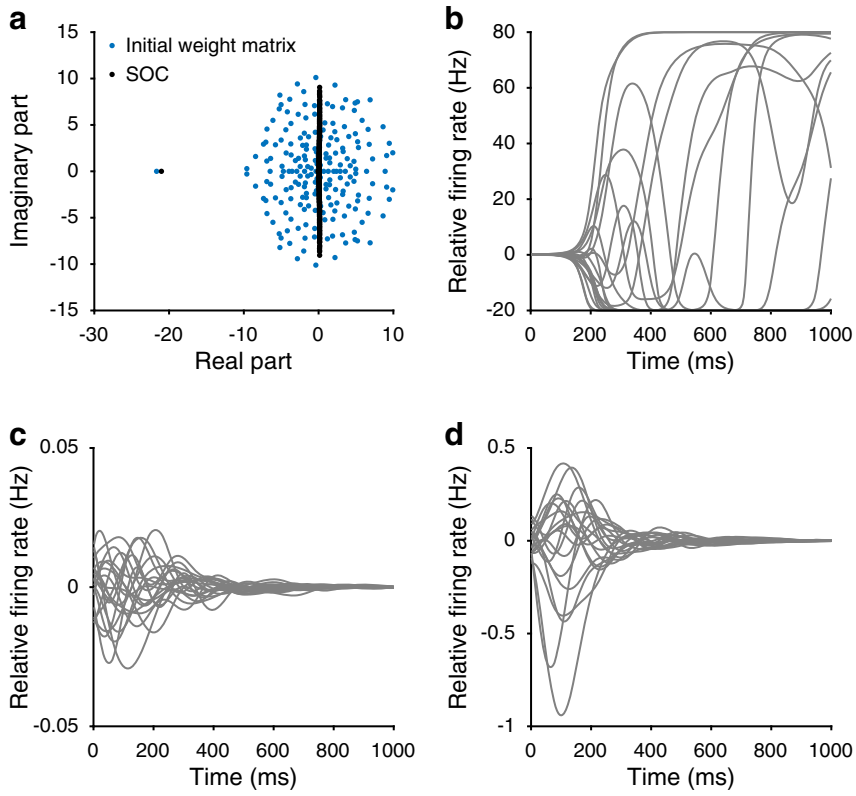

Figure 1.6: Stability-optimised circuits. **a**, *We show the spectrum of a* 200*-neuron weight matrix before stability optimisation (initial weight matrix; blue dots) and the spectrum of the same weight matrix following stability optimisation (SOC; black dots). See the main text and Appendix C for details of the network construction.* **b**, *We plot the neuronal firing rates for* 20 *excitatory neurons for the initial (unstable) network prior to stability optimisation. The neuronal activities are governed by Eqn.* (1.8) *with the nonlinear gain function Eqn.* (1.12) *with* $r_0 = 20$ *Hz,* $r_{\max} = 100$ *Hz, and* $\tau = 200$ *ms. The initial condition* $\boldsymbol{x}_0$ *is chosen from a uniform distribution over the interval* $[-1, 1]$ *and is then rescaled so that* $\|\boldsymbol{x}_0\|_2 = 1$*.* **c**, *Same as panel (b), but we use the weight matrix obtained after stability optimisation to generate the neuronal firing rates. One can compare these dynamics with those plotted in Fig. 1.5.* **d**, *Same as panel (c), but we use the initial condition* $\boldsymbol{a}_1$ *that produces the largest evoked energy* ε *(see Eqn.* (1.17)*) under the constraint that* $\|\boldsymbol{a}_1\|_2 = 1$ *(see the main text). Also note the difference in the vertical axis scale between panels (c) and (d).*

In Fig. 1.6d, we plot the neuronal dynamics in response to the initial condition $\boldsymbol{a}_1$. We see that on average, the neuronal dynamics have larger deviations away

from 0 than for an initial condition that is chosen uniformly at random (compare panels (d) and (c)).

It has been demonstrated that the neuronal dynamics of stability-optimised circuits are qualitatively similar to neuronal activity observed in primary motor cortex during movement execution (see Fig. 1.7) (Churchland et al., 2012; Hennequin et al., 2014).

1.3 Cortical motor circuits

Over the last decade, significant progress has been made on understanding the dynamics of cortical motor circuits and how they control and generate movements. In this section, we briefly summarise some of these key findings.

In a common experimental motor task, a monkey is trained to reach from the centre of a screen to a target location. Electrophysiological recordings of motor cortex are then taken together with electromyogram (EMG) measurements of muscle dynamics during both movement preparation and execution. When preparing to execute a particular movement, movement-specific preparatory activity is observed in premotor and motor cortices (Churchland et al., 2012). One commonly observes neuronal activity approaching a particular state at movement onset that is specific to each movement (Churchland et al., 2012; Shenoy et al., 2013). After movement onset, neurons display quickly-changing multiphasic firing-rate transients that return towards baseline following movement completion (see Fig. 1.7a). Each neuron normally displays unique patterns of activity that are different for different movements, and it has been difficult to understand how the complex patterns of activity observed at the single-neuron level relates to the intended movement (Churchland et al., 2012; Shenoy et al., 2013).

Based on these results and others (e.g., Afshar et al. (2011); Churchland and Cunningham (2014); Churchland et al. (2010); Shenoy et al. (2013)), it has been suggested that it is useful to view motor cortex as an autonomous dynamical sys-

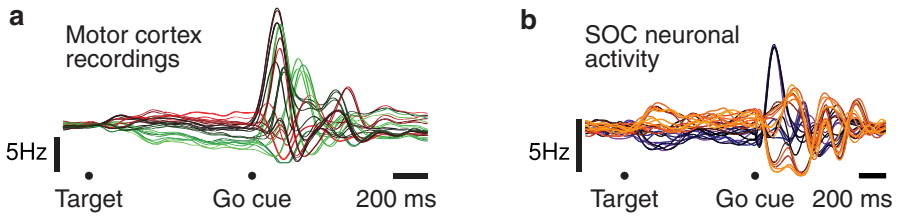

Figure 1.7: Experimental recordings and simulations of single-neuron firing rates in motor cortex. This figure was adapted with permission from Hennequin et al. (2014). **a***, Example firing rates of one neuron in macaque motor cortex for* 27 *different reach conditions during a delayed reaching task.* **b***, Firing-rate activity for one neuron in the stability-optimised circuit model for* 27 *different initial conditions where each initial condition is chosen as a different weighted combination of* $\boldsymbol{a}_1$ *and* $\boldsymbol{a}_2$ *(see Section 1.2.4) with combination weights chosen from the uniform distribution over the domain* $[-1, -0.5] \cup [0.5, 1]$*. Each trace is coloured according to the amount of preparatory activity at the time of the go cue.*

tem, where movement-specific preparatory activity sets the initial condition for the system, whose subsequent evolution drives desired downstream muscle activity. From this perspective, the complex single-neuron firing-rate dynamics provide a flexible basis set for the generation of movements (Churchland and Cunningham, 2014). Furthermore, neural activity in primary motor cortex can be accurately captured by a simple linear dynamical-systems model of a similar form to Eqn. (1.9) (Lara et al., 2018). This suggests that the transient amplification that we observe in primary motor cortex during movement execution (see Fig. 1.7) (Churchland and Cunningham, 2014) is likely a result of non-normal dynamics (although, for example, it is possible that external inputs still affect motor cortex dynamics during movement execution).

In line with this perspective, several recurrent neuronal-network models have been developed to capture neural activity observed during movement execution. One of these models is the stability-optimised circuit (Hennequin et al., 2014), which we discussed in Section 1.2.4. This model exploits inhibition to stabilise neuronal activity and places an emphasis on biologically plausible connectivity.

Another model is a 'chaotic' recurrent network (see Section 1.2.1.3) in which the recurrent connections are optimised using supervised training so that a weighted combination of the neuronal activity produces naturalistic muscle activity (Sussillo et al., 2015). Following training, either model can qualitatively reproduce neuronal activity observed in motor cortex during movement execution. (See Fig. 1.7b for example neuronal dynamics resulting from a stability-optimised circuit.) However, these models cannot explain how new movements are be learned or how their static architecture allows variations in both output trajectories and their speed.

1.4 Why study gain modulation in cortical motor circuits?

There are several possibilities as to how cortical activity can be modified to produce desired motor outputs. One possibility is that the initial condition of neural activity (i.e., $\boldsymbol{x}_0$ in Eqn. (1.8)) is changed depending on the intended movement (Churchland et al., 2012; Shenoy et al., 2013). However, changing the initial condition cannot reliably extend the duration of neural activity transients. Alternatively, when learning new movements, changes often occur in the connections between neurons in several cortical motor areas (Komiyama et al., 2010; Peters et al., 2014; Sanes and Donoghue, 2000; Xu et al., 2009). However, it has been suggested that there is very little circuit reorganisation in motor cortices during the time course of a single experimental session (i.e., on the order of about an hour) (Perich et al., 2017; Sadtler et al., 2014). To quickly switch between movements, rather than employing synaptic plasticity, it is likely that a change occurs in the input to cortical motor circuits, which may affect the neurons' response properties. Indeed, there is experimental evidence demonstrating that changes occur in the input–output sensitivity (i.e., gain modulation) of neurons in motor cortex (Kida and Mitsushima, 2018) and downstream spinal motoneurons (Vestergaard and Berg, 2015; Wei et al., 2014) during skill acquisition and control of muscle activity. Such gain changes are similar to modifying the gain value g_i in Eqn. (1.12).

Gain modulation is a possible mechanism to generate new patterns of neural activity while circuit connectivity remains fixed (Salinas and Sejnowski, 2001; Salinas and Thier, 2000; Swinehart et al., 2004; Zhang and Abbott, 2000). Additionally, gain modulation may play a role in controlling the duration of neural activity transients (Wang et al., 2018). However, there have been only a few theoretical studies investigating the effects of gain modulation in model neuronal networks. One such study has demonstrated that relatively subtle changes in the input–output gain of recurrently connected neurons can cause substantial changes in network output activity (e.g., a linear weighted sum of neuronal firing rates, see Section 2.6) (Zhang and Abbott, 2000). Furthermore, modulating the gain of neurons in the input layer of a feedforward neuronal network can allow neuronal activity in the output layer to approximate a variety of target outputs, such as $\sin$ and $\cos$ functions (Swinehart et al., 2004). However, it remains unclear what the effects of gain modulation are in recurrent neuronal networks that exhibit rich temporal neuronal dynamics (i.e., reminiscent of those observed in motor cortex during movement execution).

Realistically, gain modulation may occur from a variety of mechanisms. It has been shown *in vitro* that background excitatory and inhibitory inputs can act as a gain-control mechanism, in which increasing the synaptic input in a balanced manner reduces the gain of cortical neurons (Chance et al., 2002). Additionally, neuromodulators such as serotonin and dopamine have been demonstrated to affect the gain of neurons in a variety of areas including cortex (Thurley et al., 2008), basal ganglia (Hernandez-Lopez et al., 2000), and the spinal cord (Wei et al., 2014).

In this thesis, we study the effects of gain modulation in recurrent neuronal networks with rich temporal dynamics (such as those we discussed in Section 1.2.4). In Chapter 2, we investigate the effects of changing the gain of all neurons identically in recurrent neuronal networks.

CHAPTER 2

Global gain modulation in recurrent neuronal networks

In this chapter, we demonstrate some of the key effects of identically modulating the gain of all neurons — which we call *global gain modulation* — in recurrent neuronal-network models. We build on material from Chapter 1 regarding the stability of neuronal dynamics by studying how global gain modulation affects stability. We examine the effects of gain changes on the spectrum of the effective network connectivity matrix and how this relates to the frequency and amplitude of the neuronal dynamics. We then investigate how global gain modulation relates to variability in network outputs (which we suggest may relate to variability in muscle activity).

2.1 Introduction

Cortical networks receive a multitude of inputs from many other brain regions. Many such inputs to cortical circuits can affect the intrinsic gain — that is, the input–output sensitivity — of neurons. For example, increasing the amount of balanced synaptic input to a cortical neuron can decrease its input–output gain (Chance

et al., 2002) and excitatory synaptic inputs exhibiting short-term depression can affect neuronal gain sensitivity (Rothman et al., 2009). Additionally, neuromodulators, such as dopamine and serotonin, can affect the input–output gain of neurons in many areas including cortex (Thurley et al., 2008), basal ganglia (Hernandez-Lopez et al., 2000), and the spinal cord (Wei et al., 2014).

Various roles have been suggested for why neuronal gain modulation may be useful. In the following, we describe some studies regarding the potential roles of global gain modulation (i.e., identically modulating the gain of all neurons). Using fluctuations in pupil diameter as a proxy for gain changes in the central nervous system, it has been suggested that when under time pressure during decision making, there is an increase in the gain of the central nervous system in humans (Murphy et al., 2016). From these results, it was hypothesised that increases in gain push neuronal activity closer to a decision threshold, so that a decision can be made within a given time constraint (Murphy et al., 2016).

Gain modulation has also been proposed as a mechanism for flexible control of functional brain connectivity (Salinas and Bentley, 2009). For example, it has been demonstrated that brain-wide gain modulation can reinforce dominant neural pathways in humans (Eldar et al., 2013). The authors observed that temporal correlations between fMRI time series tracked slow fluctuations in the human participants' pupil size (Eldar et al., 2013). Additionally, the larger the participants' neural gain (which was inferred by their pupil size), the more the participants' task performance aligned with their predisposed learning style.

In other research, the gain of motoneurons in the spinal cord was linked experimentally to optimisation of muscular control (Vestergaard and Berg, 2015). In their experiments, the authors observed that the gain of neurons in the spinal cord of turtles is modulated on sub-second time scales in accordance with the required force production of a limb — the larger the required force, the larger the gain of the pool of motoneurons. This led the authors to suggest that the gain of spinal motoneu-

rons is modulated to minimise variability in muscle-force production (Vestergaard and Berg, 2015). Additionally, the gain of motoneurons in the spinal cord of humans can be modulated systematically by serotonin; this also results in increased variability in muscle activity during precision tasks (Wei et al., 2014).

As we mentioned in Chapter 1, there have been few theoretical studies into gain modulation in model neuronal networks (Salinas and Sejnowski, 2001). Previous research suggests that relatively small levels of gain modulation in recurrent neuronal networks allows substantial switching in the activity of output neurons (Zhang and Abbott, 2000). Additionally, independently modulating the gain of neurons in the input layer of a feedforward network can allow units in the output layer to generate a variety of target functions (Swinehart et al., 2004). However, the effects of gain modulation in recurrent neuronal networks with rich temporal dynamics (i.e., similar to those observed in many cortical areas; see Section 1.3) remain unclear.

In this chapter, we study how global gain modulation in recurrent neuronal-network models affects the neuronal dynamics. Using methods from dynamical systems and network theory, we identify conditions that guarantee asymptotic stability of the neuronal dynamics on a network with respect to the largest real part in the spectrum of the connectivity matrix $\boldsymbol{W}$ (i.e., the spectral abscissa of $\boldsymbol{W}$; see Section 1.2.1.2) and the input–output gain function. We show that the level of neuronal gain before instability arises, which we call the *critical gain*, is inversely proportional to $\alpha(\boldsymbol{W})$, where $\alpha(\boldsymbol{W})$ is the spectral abscissa of $\boldsymbol{W}$ (see Section 1.2.1.2). We prove this for both a linear and a nonlinear gain function and provide supporting numerical simulations.

We then investigate the effects of global gain modulation on the resulting neuronal dynamics whilst they remain in the regime of being asymptotically stable. To test our results in networks that exhibit dynamics reminiscent of cortical activity, we follow Hennequin et al. (2014) and create 'stability-optimised circuits'. These networks consist of a set of sparse, strong excitatory recurrent connections that

are stabilised by fine-tuned inhibition (see Section 1.2.4). We demonstrate that changes in neuronal gain produce predictable changes in the frequency and amplitude of the transient neuronal activity. We then show that the global gain in a recurrent neuronal network correlates positively with variability in the network output activity, and we suggest that global gain in cortical motor circuits may correlate positively with variability in muscle activity. Our results build on the prior links already drawn from experimental work regarding neuronal gain modulation and muscle control (Coxon et al., 2005; Salinas and Thier, 2000; Vestergaard and Berg, 2015; Wei et al., 2014).

2.2 Methods

We study recurrent neuronal networks whose state $\boldsymbol{x}(t) = (x_1(t), \ldots, x_N(t))^\mathsf{T}$ evolves according to the dynamical system

$$\tau \frac{d\boldsymbol{x}(t)}{dt} = -\boldsymbol{x}(t) + \boldsymbol{W} f(\boldsymbol{x}(t); g) + \boldsymbol{h}(t)\,, \tag{2.1}$$

from some initial condition $\boldsymbol{x}(0) = \boldsymbol{x}_0$. In Eqn. (2.1), $f(\boldsymbol{x}; g)$ denotes the element-wise application of the static scalar gain function f to the neuronal activity vector $\boldsymbol{x}$. In keeping with Hennequin et al. (2014), we set the single-neuron time constant to be $\tau = 200$ ms, and $\boldsymbol{h}(t)$ denotes an external input. The gain function f, which governs the transformation of neuronal activity $\boldsymbol{x}$ into firing rates relative to a baseline rate r_0, is

$$f(x_i; g) = \begin{cases} r_0 \tanh(g x_i / r_0)\,, & \text{if } x_i < 0\,, \\ (r_{\max} - r_0) \tanh(g x_i / (r_{\max} - r_0))\,, & \text{if } x_i \geq 0\,, \end{cases} \tag{2.2}$$

where the global scalar gain value g is the slope of the function f at $\boldsymbol{x} = \boldsymbol{0}$ and thus g controls the input–output sensitivity of all neurons (Rajan et al., 2010). We use a baseline rate of $r_0 = 20$ Hz and a maximum firing rate of $r_{\max} = 100$ Hz. We choose these values so that when we use stability-optimised circuits (see Sec-

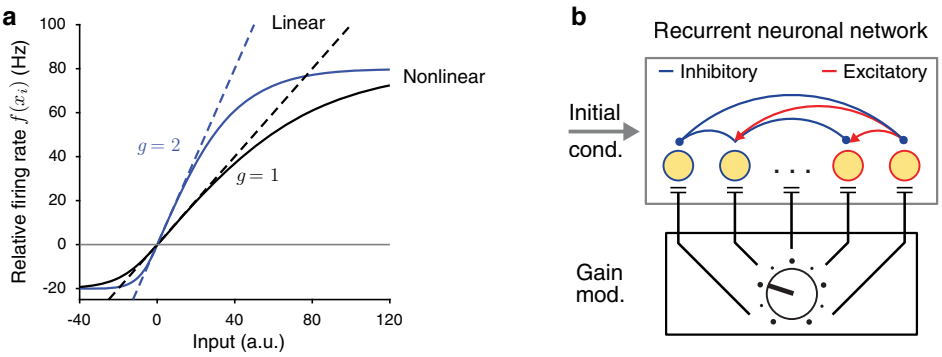

Figure 2.1: **a**, *We show the nonlinear gain function f from Eqn. (2.2) (solid curves) with (black) $g = 1$ and (blue) $g = 2$. We also show the linearised versions (i.e., $f_{\mathrm{l}}(x_i; g) = gx_i$) using the dashed lines with the same two gain values.* **b**, *Illustration of a recurrent neuronal network in which we can control the gain (which we indicate by the dial) of all neurons in the recurrent network. Red arrows denote excitatory connections and blue circles denote inhibitory connections.*

tion 2.5), these values produce neuronal activities that are consistent with observations (Churchland et al., 2012; Kao et al., 2015; Lara et al., 2018). (We also revisit how such choices affect the neuronal activities later in Section 3.7.)

As we mentioned in Section 1.2.1, $f(\boldsymbol{x}; g)$ describes the neuronal firing rates relative to the baseline r_0. We also use a linearised version of the gain function, which we denote by $f_{\mathrm{l}}(x_i; g) = gx_i$. See Fig. 2.1, where we plot both f and f_{l} for two different values of the gain g.

Following Hennequin et al. (2014), we create weight matrices $\boldsymbol{W}$ with $N = 2M$ neurons (of which M are excitatory and M are inhibitory), where the probability of connection between any two neurons is p. Nonzero elements of $\boldsymbol{W}$ are set to $w_0/\sqrt{N}$ for excitatory connections and to $-\gamma w_0/\sqrt{N}$ for inhibitory connections, where $w_0^2 = 2\rho^2/(p(1-p)(1+\gamma^2))$. This construction results in $\boldsymbol{W}$ having an approximately circular spectrum of radius ρ (Hennequin et al., 2014). We set the inhibition/excitation ratio to $\gamma = 1$ unless we state otherwise (e.g., see Section 2.5). In Sections 2.3 and 2.4, we use various different values for N, p, and ρ. In

Section 2.5, we also use stability-optimised circuits to generate neuronal dynamics similar to those observed in cortex (Hennequin et al., 2014). See Fig. 2.1b for an illustration of our model.

2.3 Stability of neuronal activity

A major effect that gain modulation can have in recurrent neuronal networks is to change the (local) asymptotic stability of neuronal activity (Sompolinsky et al., 1988). Briefly, we are interested in whether a small perturbation of the neuronal activity around $\boldsymbol{x} = \mathbf{0}$ grows or decays over time. If the perturbation decays, the baseline firing rate is asymptotically stable. However, if the perturbation grows indefinitely, we say the neuronal dynamics are unstable (see Section 1.2.1.2).

In this section, we build on Section 1.2.1.2, where we discussed asymptotic stability of linear systems, and study the asymptotic stability of the neuronal dynamics governed by Eqn. (2.1) with respect to the gain g and the weight matrix $\boldsymbol{W}$. We utilise methods from dynamical systems to treat both linear (f_{l}) and nonlinear (f) gain functions, and we obtain bounds on the decay of the solution of Eqn. (2.1). Our approach relies on notions of stability for systems of nonlinear autonomous differential equations. There are several excellent textbooks and other materials regarding this topic (Pereira, 2011; Teschl, 2012; Wiggins, 2003). See Appendix 2.A for any mathematical definitions that we use in this section.

2.3.1 Analytical results

Our key analytical result is the following. Suppose that we are given a differentiable (possibly nonlinear) gain function $f(\boldsymbol{x})$ in Eqn. (2.1) with a slope $g \neq 0$ at $\boldsymbol{x} = \mathbf{0}$ and any weight matrix $\boldsymbol{W}$. If

$$g < 1/\alpha(\boldsymbol{W})\,, \tag{2.3}$$

we show that in the absence of external input (i.e., $\boldsymbol{h} = \mathbf{0}$), then $\boldsymbol{x} = \mathbf{0}$ is locally asymptotically stable under the dynamics governed by Eqn. (2.1). Additionally, if Eqn. (2.3) is satisfied, then for a sufficiently small external input $\boldsymbol{h}(t)$ in Eqn. (2.1), for long times we have

$$\|\boldsymbol{x}(t)\| \leq K\|\boldsymbol{h}(t)\|\,, \tag{2.4}$$

for some positive real number K. Therefore, sufficiently small, (possibly time-dependent) external inputs do not affect the stability of the neuronal dynamics. See Appendix 2.B for the mathematical details and proofs of Eqns. (2.3) and (2.4).

For the case of the linear gain function $f_1(\boldsymbol{x}; g) = g\boldsymbol{x}$, we show that if $g < 1/\alpha(\boldsymbol{W})$, the dynamics of Eqn. (2.1) are globally asymptotically stable (see Appendix 2.B for a proof). Therefore, given any initial condition, with no external input, the dynamics decay exponentially to $\mathbf{0}$. We also note that for the linear gain function $f_1(\boldsymbol{x}; g) = g\boldsymbol{x}$, the gain factor g can be interpreted as simply scaling the weight matrix $\boldsymbol{W}$ in Eqn. (2.1) and the stability condition can be re-written as $\alpha(g\boldsymbol{W}) < 1$.

These results imply that given a weight matrix $\boldsymbol{W}$, the dynamics on $\boldsymbol{W}$ can be brought into the asymptotically stable regime by globally modifying the gain of all neurons (e.g., through neuromodulators (Hernandez-Lopez et al., 2000; Thurley et al., 2008; Wei et al., 2014)). Alternatively, the neuronal dynamics on $\boldsymbol{W}$ can be brought into the asymptotically stable regime by reducing the spectral abscissa of $\boldsymbol{W}$ by changing the synaptic connections. For example, Hebbian synaptic plasticity rules can reduce the spectral abscissa of $\boldsymbol{W}$; see (Hennequin, 2013, Chapter 4) for a discussion. Such plasticity rules are similar to rules that can be used to maintain detailed E/I balance in feedforward spiking networks (Vogels et al., 2011).

In Fig. 2.2a, we show the spectrum of a weight matrix $\boldsymbol{W}$ that we generated as detailed in Section 2.2 with $N = 200$, $p = 0.1$, and $\rho = 1$. We show the critical gain (which we define as $g = 1/\alpha(\boldsymbol{W})$), given the spectral abscissa of $\boldsymbol{W}$. In general, increasing the gain results in dynamics that decay more slowly. However, if the gain is increased above $1/\alpha(\boldsymbol{W})$, the dynamics become unstable.

A stability condition related to Eqn. (2.3) was obtained previously (Sompolinsky et al., 1988). However, the authors used a mean-field approach that relied on the assumption that $N \to \infty$, and they did not consider the case of an external input $\boldsymbol{h}$ in Eqn. (2.1). Here, we independently treat the gain value g and the spectral abscissa $\alpha(\boldsymbol{W})$, and for our stability conditions Eqn. (2.3) and Eqn. (2.4), we do not make any assumptions regarding how $\boldsymbol{W}$ is constructed (such as the synaptic weights having a mean of 0). Our approach also focusses on applying bounds on the neuronal activity $\boldsymbol{x}$ over time rather than focussing on the case of long times.

2.3.2 Numerical results

To test our analytical predictions, we construct 200 weight matrices where the number N of neurons, the connection probability p, and the approximate spectral abscissa ρ (see Section 2.2) are drawn from the uniform distribution on the intervals $[50, 1000]$, $[0, 0.5]$, and $[0, 20]$, respectively. We simulate dynamics according to Eqn. (2.2) with $\boldsymbol{h} = \boldsymbol{0}$, and we use both f_{l} and f and the initial condition $\boldsymbol{x}_0$ is drawn randomly from the uniform distribution on the interval $[-0.025, 0.025]$ (i.e., $\boldsymbol{x}_0$ is close to $\boldsymbol{0}$).

For each of the 200 weight matrices, we initially set $g = 0.9/\alpha(\boldsymbol{W})$ and increase g in increments of 10^{-4} until $\|\boldsymbol{x}(T)\|_2 > \|\boldsymbol{x}_0\|_2$ with $T = 10$ s. To numerically estimate the critical gain at which instability arises, we define the first value of g such that $\|\boldsymbol{x}(T)\|_2 > \|\boldsymbol{x}_0\|_2$ as the critical gain. We plot these results in Fig. 2.2b. Regardless of whether we use the nonlinear f (main) or linear f_{l} (inset) gain functions, all simulation results lie very close to the theoretical prediction. (We use the same weight matrices for both f and f_{l}.) The region of asymptotically stable (respectively, unstable) neuronal activity corresponds to being under (respectively, above) the red curve. These results support our analytical predictions by showing that changes in the spectral abscissa of the weight matrix or changes in the global neuronal gain can lead to changes in the local stability of the neuronal dynam-

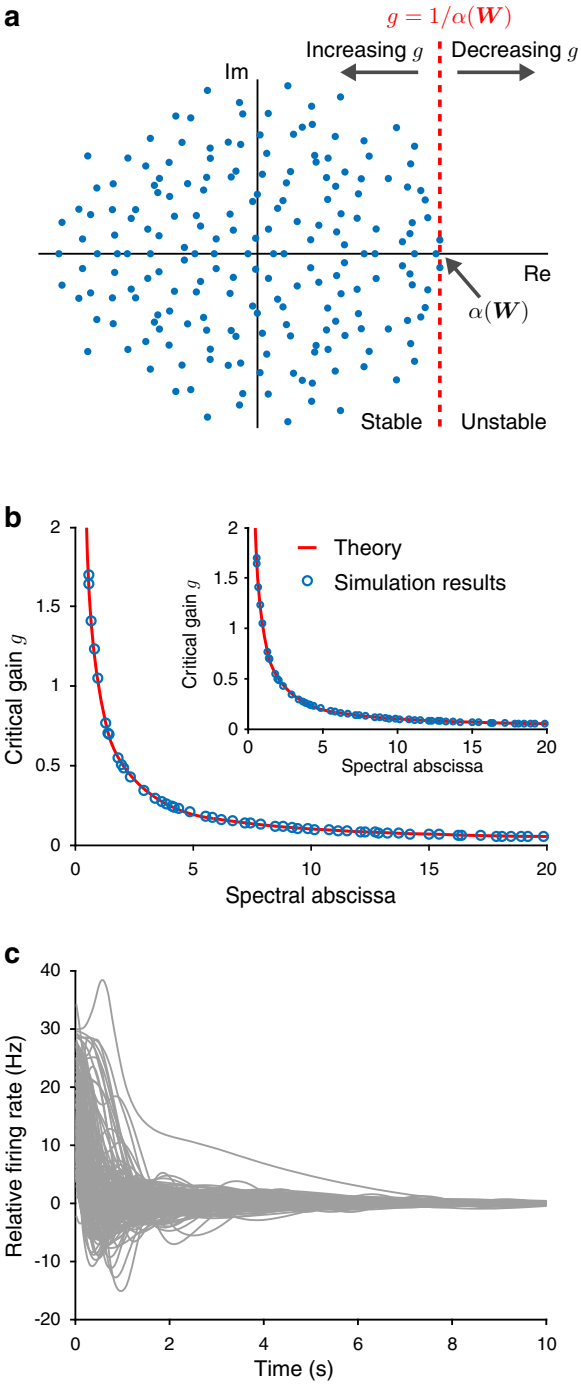

Figure 2.2: **a**, *We plot the spectrum of a weight matrix* $\boldsymbol{W}$*, which is generated as detailed in Section 2.2 with* $N = 200$*, the connection probability* $p = 0.1$*, and approximate spectral abscissa* $\rho = 1$*. The red dashed line is the spectral abscissa* $\alpha(\boldsymbol{W})$*. If* $g < 1/\alpha(\boldsymbol{W})$*, then the dynamics on the network are asymptotically stable. Increasing* g *above* $1/\alpha(\boldsymbol{W})$ *results in unstable neuronal dynamics.* **b**, *Simulation results for the critical gain* g *before instability arises using* 200 *different weight matrices where* N*,* p*, and* ρ *are drawn from the uniform distribution on the intervals* $[50, 1000]$*,* $[0, 0.5]$*, and* $[0, 20]$*, respectively. We show results when using either the nonlinear* f *(main) or* f_1 *(inset) gain functions along with the theoretically predicted inverse relationship between the critical gain* g *and* $\alpha(\boldsymbol{W})$ *(i.e.,* $g = 1/\alpha(\boldsymbol{W})$*) in red (see the main text). For visualisation purposes, we show only* 55 *data points.* **c**, *For each of the* 200 *weight matrices, we generate initial conditions* $\boldsymbol{x}_0$ *from the uniform distribution on the interval* $[-1000, 1000]$ *and plot the mean neuronal firing rate using the nonlinear gain function* f *with* $g = 0.95/\alpha(\boldsymbol{W})$*. Although the neuronal firing rates are initially not close to* 0*, the mean firing rates decay to* 0 *for each weight matrix.*

ics around $\mathbf{0}$. It also shows that changes in stability are accurately described by Eqn. (2.3).

For the numerical simulations that we show in Fig. 2.2b, we use an initial condition $\boldsymbol{x}_0$ for the neuronal activity that is close to $\mathbf{0}$. For the nonlinear gain function, however, the equilibrium point at $\mathbf{0}$ is not guaranteed to be globally attractive (which is guaranteed to be the case with the linear gain function; see Appendix 2.B). Therefore, to examine whether the equilibrium point at $\mathbf{0}$ has a large basin of attraction when using the nonlinear gain function f, we use the same 200 weight matrices that we described above but now we draw the initial condition $\boldsymbol{x}_0$ randomly from the uniform distribution on the interval $[-1000, 1000]$, and we set $g = 0.95/\alpha(\boldsymbol{W})$ (i.e., close to the critical gain level). (Note that if the spectral abscissa of a weight matrix is large, then $g = 0.95/\alpha(\boldsymbol{W})$ is small, so after the initial condition $\boldsymbol{x}_0$ is passed through the gain function f, the resulting firing rate will have a support that consists of a smaller domain than $[-1000, 1000]$.) For each weight matrix, we plot the mean firing-rate activity over time in Fig. 2.2c. We see that the mean activity ultimately decays to 0 for all weight matrices. Therefore, for the simulations that we performed, we find that there is a relatively large basin of attraction to $\mathbf{0}$ when using the nonlinear gain function f (e.g., from Fig. 2.2c, we observe that the initial neuronal firing rates cover approximately half of the total range of possible firing rates (i.e., approximately the interval $[-15, 40]$)).

2.4 Relationship between eigenvalues and changes in neuronal gain and time constant τ

In Section 2.3, we identified the range of allowable gain modulation while the neuronal dynamics remain asymptotically stable (see Eqn. (2.3)). We now investigate the effects of gain modulation whilst the neuronal dynamics remain asymptotically stable. In this section, we consider the case of a linear gain function $f_1(\boldsymbol{x}) = g\boldsymbol{x}$ for

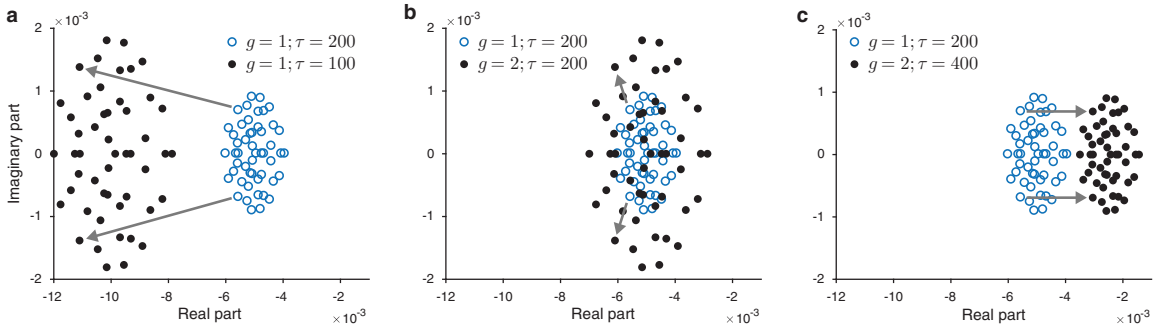

Figure 2.3: Changes in the spectrum of the matrix $(g\boldsymbol{W} - \boldsymbol{I})/\tau$ when changing the gain g and/or the time constant τ. **a**, *Changes in the spectrum when changing τ.* **b**, *Changes in the spectrum when changing the gain g.* **c**, *Changes in the spectrum when changing both τ and g. We indicate example changes for a complex conjugate pair of eigenvalues using the grey arrows.*

analytical tractability; we note that $f(\boldsymbol{x}) = f_1(\boldsymbol{x}) + \mathcal{O}(\boldsymbol{x}^2)$, so f and f_1 are the same to a first order approximation. For the linear gain function f_1 and with $\boldsymbol{h} = \boldsymbol{0}$, we noted in Section 1.2.1.1 that the solution to Eqn. (2.1) is given by

$$\boldsymbol{x}(t) = e^{(g\boldsymbol{W}-\boldsymbol{I})t/\tau}\boldsymbol{x}_0\,. \tag{2.5}$$

By studying changes in the spectrum of the matrix $(g\boldsymbol{W} - \boldsymbol{I})/\tau$, we can understand how the resulting neuronal dynamics are affected. For example, if $\alpha(g\boldsymbol{W} - \boldsymbol{I}) < 0$, then $\alpha((g\boldsymbol{W} - \boldsymbol{I})/\tau)$ describes the decay envelope of the neuronal activity (see Appendix 2.A). Additionally, there is a relationship between the imaginary eigenvalues and the frequency of oscillation of the neuronal activity (Teschl, 2012): the larger the imaginary eigenvalues of the matrix $(g\boldsymbol{W} - \boldsymbol{I})/\tau$, the higher the frequency of oscillation of the resulting dynamics. This is the case for a linear gain function, however, in Section 2.5, we investigate whether these effects hold true when using a nonlinear gain function.

In Fig. 2.3, we compare and contrast the effects of changing the neuronal gain g and changing the single-neuron time constant τ on the spectrum of the matrix $(g\boldsymbol{W} - \boldsymbol{I})/\tau$. We generate a 50-neuron weight matrix as described in Section 2.2

with a connection probability of $p = 0.1$ and an approximate spectral abscissa of $\rho = 0.2$. Thus, if $g < 5$, then $\alpha((g\boldsymbol{W} - \boldsymbol{I})/\tau) < 0$. We note that simply changing τ does not affect the asymptotic stability of the neuronal dynamics.

Halving the time constant τ causes $\alpha((g\boldsymbol{W} - \boldsymbol{I})/\tau)$ to be more negative (implying a faster rate of decay of the dynamics to zero) and increases the imaginary part of the spectrum (implying that the dynamics oscillate with a higher frequency) (see Fig. 2.3a). We expect this result because τ simply scales time in Eqn. (2.1). In contrast, doubling the gain causes the same change in the imaginary part of the spectrum, but the real part of the spectrum is expanded about the centre of the spectrum and $\alpha((g\boldsymbol{W} - \boldsymbol{I})/\tau)$ is increased towards 0 (see Fig. 2.3b).

Finally, we can change both the gain and the time constant such that $\alpha((g\boldsymbol{W} - \boldsymbol{I})/\tau)$ increases and the imaginary part of the spectrum remains unchanged (see Fig. 2.3c). This implies that the resulting linear neuronal dynamics given by Eqn. (2.5) will decay more slowly but oscillate with the same frequency.

2.5 Effects of global gain modulation on transient neuronal activity

We now investigate whether the claims that we made in Section 2.4 regarding the expected changes in neuronal firing rates following gain modulation are valid in recurrent neuronal networks that display realistic firing-rate activity. To study this, we use stability-optimised circuits (SOCs) (see Section 1.2.4) (Hennequin et al., 2014). SOCs display complex multiphasic neuronal activity transients that resemble firing-rate activity in motor cortex during movement execution. To generate a SOC, one starts with a weight matrix as described in Section 2.2 with an approximate spectral abscissa $\rho = 10$. The resulting neuronal activities using this weight matrix are unstable because the spectral abscissa is larger than 1 (see Section 1.2.4). One then stabilises such an initially unstable weight matrix by changing

the inhibitory connections to reduce its spectral abscissa. This is achieved using an optimisation algorithm. (See Section 1.2.4 and Appendix C for a complete description.) This results in a weight matrix $\boldsymbol{W}$ that, when endowed with neuronal dynamics governed by Eqn. (2.1), displays complex multiphasic neuronal activity (due to the strong excitatory connections), which ultimately decays to a baseline firing rate following any initial condition.

To generate neuronal firing rates that have larger amplitude oscillations than would be expected from an initial condition chosen uniformly at random (which we have used so far in this chapter), we can analytically (for the case of a linear gain function and with $\alpha(\boldsymbol{W}) < 1$) determine the initial condition $\boldsymbol{x}_0$ that maximises the 'evoked energy' $\varepsilon(\boldsymbol{x}_0)$ (see Eqn. (1.17) in Section 1.2.4 and Appendix C.2) (Hennequin et al., 2014). Choosing such an initial condition generates neuronal activity with large amplitude deviations away from baseline. We also find empirically that we obtain neuronal activity with similar, large amplitude deviations away from baseline when using a nonlinear gain function (Hennequin et al., 2014).

Using the above-mentioned procedure, we create a SOC consisting of 200 neurons with $\alpha(\boldsymbol{W}) = 0.15$ after stabilisation. We plot the firing rate of one neuron in Fig. 2.4 using the nonlinear gain function f in Eqn. (2.2) and using the initial condition $\boldsymbol{x}_0$ that maximises the evoked energy $\varepsilon(\boldsymbol{x}_0)$. We use the same values of the global gain value g and time constant τ that we used in Fig. 2.3 to generate Fig. 2.4. We see that halving τ causes a doubling of the frequency of the neuronal firing rate (see Fig. 2.4a), which one would naturally expect because the time constant τ scales time in Eqn. (2.1). Doubling the gain g causes an increase in the amplitude of the neuronal firing rate (see Fig. 2.4b); this occurs because $\alpha((g\boldsymbol{W} - \boldsymbol{I})/\tau)$ increases towards 0, which implies that the linear neuronal dynamics decay more slowly. When doubling the gain g, the frequency of the firing rate also approximately doubles. We predicted these effects by studying changes in the spectrum of the associated linear system in Section 2.4, however, we find empiri-

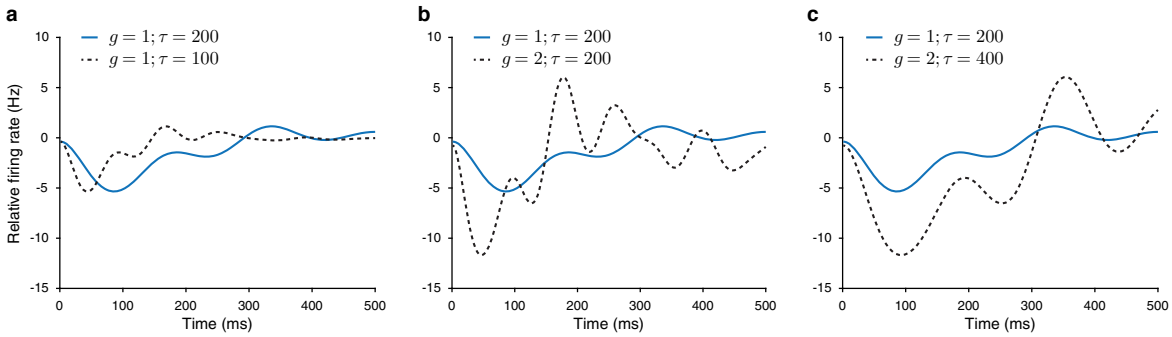

Figure 2.4: Effects of changing the global gain g and the single-neuron time constant τ on neuronal firing rates in stability-optimised circuits when using the nonlinear gain function f in Eqn. (2.2). **a**, *We plot the firing rate of one neuron in a 200-neuron network for two different values of the time constant τ. Only the frequency of the neuronal dynamics is affected (see main text).* **b**, *For the same neuron, we plot its firing rate when changing the global gain g for all neurons. We see that, like panel (a), the frequency of its firing rate is approximately doubled, but the amplitude also increases.* **c**, *For the same neuron, we plot its firing rate when changing both the gain g and τ so that the frequency of the neuronal dynamics remains approximately the same but the amplitude of the activity increases.*

cally that the nonlinear system with the gain function in Eqn. (2.2) displays these effects. Finally, changing both τ and g in concert so that $\frac{\tau}{g} = 200$, can result in neuronal activity with a similar frequency but different amplitudes (see Fig. 2.4c). We only plot the activity of one neuron, but we observe similar effects for all neurons in the network.

2.6 Global gain modulation correlates with network output variability

In this section, we study whether global gain modulation in stability-optimised circuits relates to variability in network outputs (which can act as a proxy for muscle activity). This is motivated both by our results from Section 2.5 and, as we mentioned in the introduction to this chapter in Section 2.1, a recent experimental study

that found that gain sensitivity can be modulated on sub-second time scales in spinal motoneurons to reduce variability in muscle-force production (Vestergaard and Berg, 2015).

To investigate such phenomena, we study the output of a neuronal network (e.g., a weighted linear combination of neuronal firing rates; see Fig. 2.5a). For the case of neuronal network models that exhibit neuronal activity reminiscent of motor cortex (such as SOCs), the output of such a network could represent muscle-force activity (Churchland et al., 2012; Russo et al., 2018; Sussillo et al., 2015), which is typically measured using electromyography (EMG) (Churchland et al., 2012). Additionally, motor cortex is one of the final cortical outputs to downstream spinal motoneurons (Rathelot and Strick, 2009), so it is plausible to assume that a weighted combination of cortical activity may resemble muscle force activity (Churchland et al., 2012; Russo et al., 2018; Sussillo et al., 2015).

Here, we treat the network output as a weighted linear combination of excitatory neuronal firing rates. Such an approach has been widely used (Hennequin et al., 2014; Hoerzer et al., 2014; Russo et al., 2018; Sussillo and Abbott, 2009; Sussillo et al., 2015; Wang et al., 2018). We compute the network output activity $z(t)$ at time t as

$$z(t) = \boldsymbol{m}^{\mathsf{T}} f(\boldsymbol{x}^E(t)) + b\,, \tag{2.6}$$

where $\boldsymbol{m}\,, \boldsymbol{x}^E(t) \in \mathbb{R}^M$, the quantity M is the number of excitatory neurons, $\boldsymbol{x}^E(t)$ is the excitatory neuronal activity, and f is the gain function (see Eqn. (2.2)). Shortly, we will discuss how we fit the readout weights $\boldsymbol{m}$ and the offset b.

To generate target muscle activity of duration $T = 500$ ms, we draw muscle activities from a Gaussian process with a covariance function $K \in [0,T]\times[0,T] \to \mathbb{R}_{\geq 0}$ that consists of a product of a squared-exponential kernel (to enforce temporal smoothness) and a non-stationary kernel that produces a temporal envelope similar to that of real EMG data during reaching (Churchland et al., 2012). Specifically,

$$K(t,t') = e^{-\frac{(t-t')^2}{2\ell^2}} \times E(t/\sigma) \times E(t'/\sigma)\,, \tag{2.7}$$

with $E(t) = te^{(-t^2/4)}$. The parameter σ controls the duration of the generated muscle activity. Based on test simulations, we find that setting $\sigma = 110$ ms is a reasonable choice for muscle activity that lasts 500 ms (see Fig. 2.5b). The parameter ℓ controls the autocorrelation of the activity. Therefore, ℓ affects the variability of the muscle activity; a small value of ℓ implies that the activity is more variable and vice versa (see below for our choice of ℓ and also see Fig. 2.5b for two example muscle activities). We also multiply the resulting muscle activity by a constant to ensure that it has the same order of magnitude as the neuronal firing rates, and we use a sampling rate of 400 Hz (i.e., target muscle activity is a vector of length 200).

In other words, target muscle activity that we generate corresponds to $\boldsymbol{Lu}$ where $\boldsymbol{LL}^\intercal = \boldsymbol{K}$, $\boldsymbol{K} \in \mathbb{R}^{200\times200}$ is the covariance function, $\boldsymbol{u} \in \mathbb{R}^{200}$, and $\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ (i.e., elements of the vector $\boldsymbol{u}$ are drawn independently and identically from a Gaussian distribution with mean 0 standard deviation 1).

We generate 20 muscle activities using 20 different evenly-spaced values of ℓ between $\ell = 35$ ms and $\ell = 80$ ms (see Fig. 2.5b). (See also Fig. 2.7 in Appendix 2.C where we plot the power spectrum of the generated muscle activity for $l = 35$ ms and $l = 80$ ms.) We then generate neuronal dynamics that last $T = 500$ ms using the initial condition $\boldsymbol{x}_0$ that maximises the evoked energy $\varepsilon(\boldsymbol{x}_0)$ in Eqn. (1.17) for gain values between $g = 0.02$ and $g = 3$ in increments of 0.02. We use the nonlinear gain function f in Eqn. (2.2). We fit the readout weights $\boldsymbol{m}$ and offset b in Eqn. (2.6) independently for each g and each of the 20 target muscle activities. To ameliorate any issues of overfitting, we use 100 noisy trials in which we add Gaussian white noise to the initial condition $\boldsymbol{x}_0$ for each trial with a signal-to-noise ratio of 30 dB (Hennequin et al., 2014).

For each g and each ℓ, we compute the the R-squared (R^2) goodness of fit between the network output $z(t)$ and the target activity using the trial with no noise

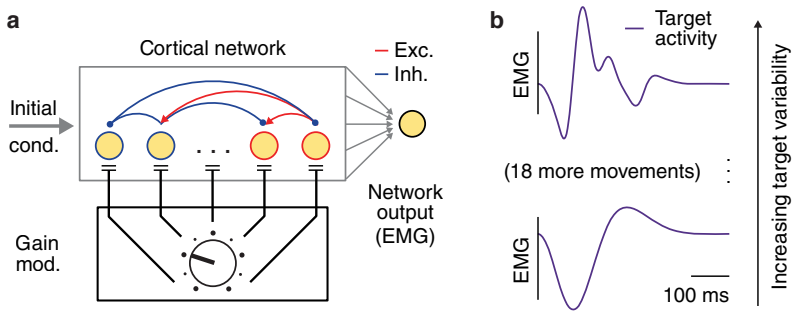

Figure 2.5: **a**, *Illustration of our recurrent neuronal network model, in which we can control the neuronal gains of all neurons and we compute the network output as a weighted linear combination of the excitatory neuronal firing rates. The network output acts as a proxy for muscle electromyogram (EMG) activity (see main text).* **b**, *Examples of simulated muscle activity with (bottom)* $\ell = 80$ *ms and (top)* $\ell = 35$ *ms in Eqn.* (2.7). *We generate* 20 *muscle activity variants using* 20 *different evenly-spaced values of* ℓ *between* 35 *ms and* 80 *ms. We use these muscle activities to generate the results that we show in Fig. 2.6. (See the main text for further details.)*

added to the initial condition. In mathematical terms, the goodness of fit is

$$R^2 = 1 - \frac{\sum_{t=1}^{T}(z(t) - y(t))^2}{\sum_{t=1}^{T}(y(t) - \bar{y})^2}\,, \tag{2.8}$$

where $y(t)$ is the target muscle activity at time t and $\bar{y} = \frac{1}{T}\sum_{t=1}^{T} y(t)$. Therefore, an R-squared of $R^2 = 0$ implies that the performance is as bad as if the output $\boldsymbol{z}$ is equal to the mean of the target $\boldsymbol{y}$ and thus does not capture any variations in output.

In Fig. 2.6a, we show a heatmap of R^2 values for each of the 20 target muscle activities over a relevant range of gains. With red crosses, we indicate the maximum R^2 value for each of the 20 targets. We observe a strong positive correlation between global gain and target variability. In other words, a larger global gain corresponds to larger variability (i.e., a small value of ℓ in Eqn. (2.7)) in target output activity. Because we fit the readout weights independently for each g and each target, this result implies that there are more suitable gains for different variabilities

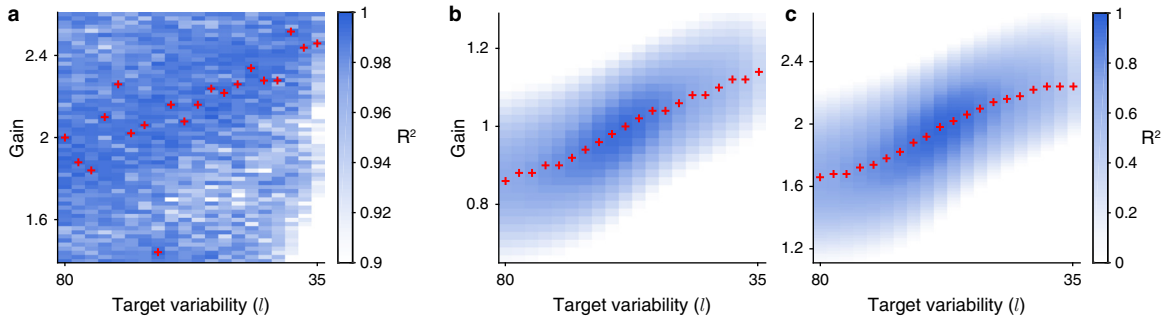

Figure 2.6: Relationship between gain modulation and target output variability. **a***, Heatmap of the goodness of fit (determined by the* R^2*) for* 20 *different variabilities in target muscle activity (which correspond to* 20 *different evenly-spaced values of* ℓ *between* 35 *ms and* 80 *ms) and different levels of neuronal gain. Readout weights are fitted independently for each gain increment and each of the* 20 *targets. Red crosses indicate the maximum* R^2 *for each of the* 20 *targets.* **b***, Same as panel (a) except that we fit the readout weights using a gain of one to all* 20 *targets simultaneously. The heatmap shows the* R^2 *values obtained when varying the neuronal gain for each of the* 20 *targets when keeping the readout weights fixed.* **c***, Same as panel (b) except that we fit the readout weights using a gain of two to all* 20 *targets simultaneously.*

in muscle activity even if we optimise the readout weights for each gain and each target. However, we note that the R^2 values are all above 0.9 in the plot. Such large R^2 values likely arise because we fit the readout weights independently for each gain increment and target.

To demonstrate this relationship between global gain and output variability even more clearly, we now fit the readout weights to all 20 targets simultaneously (using 100 noisy trials of neuronal activity, as mentioned above) with the gain set to 1. Therefore, the resulting readout weights produce a network output with a maximum mean R^2 across the 20 targets when the gain is set to 1. We then vary the gain while keeping the readout weights fixed, and examine the resulting R^2 value for each of the 20 targets. We find a strong positive correlation between gain and target variability, and a gain of 1 produces a maximum R^2 value for the target that

has approximately the mean variability over the 20 targets (see Fig. 2.6b). This relationship appears to be (mostly) invariant to the gain that we use when fitting the readout weights. This is suggested by Fig. 2.6c, where we plot results from a similar simulation but we instead use a gain of 2 to fit the readout weights. However, we do observe a more nonlinear relationship between the optimal gain and target variability compared with when we used a gain of 1. This suggests that changing the gain to relatively large or small values, respectively, while keeping the readout weights fixed does not necessarily lead to improved performance for target outputs that exhibit relatively large or small variability, respectively.

In summary, the results that we present in this section suggest that more intricate muscle activities can be most accurately generated using a larger global gain. Because there is a close relationship between dopamine and neuronal gain (Hernandez-Lopez et al., 2000; Thurley et al., 2008), our results align with observations that dopamine controls the vigour of movements (Niv et al., 2007; Panigrahi et al., 2015) and that fine-tuned motor control becomes harder with decreasing levels of dopamine (for example, as seen with Parkinson's disease (Damier et al., 1999; Panigrahi et al., 2015)).

2.7 Conclusions and discussion

In this chapter, we laid the foundations for understanding how gain modulation affects neuronal dynamics in recurrent neuronal-network models by studying the effects of global gain modulation. We derived a stability condition (see Eqn. (2.3)) relating the neuronal gain to the spectral abscissa of the weight matrix that guarantees that the neuronal dynamics governed by Eqn. (2.1) are locally asymptotically stable around 0. Our stability condition illustrates that there is an inverse relationship between the global neuronal gain g and the spectral abscissa of the weight matrix (see Eqn. (2.3) and the curves in Fig. 2.2b). We provided supporting numerical simulations using both a nonlinear and a linear gain function, and we observed

that for the nonlinear gain function that we used, the dynamics in our simulations always returned to baseline — suggesting that the basin of attraction to $\mathbf{0}$ was large. This result is useful in the context of neuronal networks, because inputs to cortical circuits are not guaranteed to be small. To produce desired patterns of temporal activity, cortical inputs can vary substantially in magnitude yet the neuronal activity typically returns to baseline (Churchland et al., 2012). This is particularly relevant for cortical motor circuits, because these circuits are thought to act as dynamical systems, where the neuronal dynamics relax to baseline once all external inputs are removed (Churchland et al., 2010; Shenoy et al., 2013). Therefore, the baseline firing rate in these circuits appears to be stable and the basin of attraction to baseline is relatively large. However, it may be several years before one can experimentally asses whether the stability condition that we derived is actually satisfied in cortical networks.

One could also extend our stability condition Eqn. (2.3) for the case of a gain function in which each neuron has its own gain value (e.g., see Eqn. (1.12); also see Chapter 3). (Note that the gain function that we used in this chapter (see Eqn. (2.2)) is a special case of the gain function in Eqn. (1.12) because we fix all gains to the same value.) From studying Eqn. (2.13), the condition guaranteeing asymptotic stability of the neuronal dynamics when using such a gain function f in Eqn. (1.12) with $f'(\mathbf{0}) = \boldsymbol{g}$, where the vector $\boldsymbol{g}$ consists of the gain values for each neuron, would be $\alpha(\boldsymbol{W} \times \mathrm{diag}(\boldsymbol{g})) < 1$. We omitted using such a gain function in our analysis because we devoted this chapter to investigations of global gain modulation.

We also studied the relationships between modifying the time constant τ in Eqn. (2.1) and modifying the gain g, and we examined the resulting spectrum changes. From our analysis and simulations, we suggested that by changing τ and g in concert so that $\frac{\tau}{g} = 200$, one can modify only the amplitude of the resulting linear neuronal dynamics while their frequency of oscillation is unaffected. We then

provided supporting numerical simulations using a nonlinear gain function and a recurrent neuronal-network model and observed such changes in the frequency and amplitude of the neuronal dynamics that we expected from changes in the neuronal gain and time constant τ. Following these results, and in line with recent experimental evidence suggesting that neuronal gain is associated with variability in muscle force activity (Vestergaard and Berg, 2015; Wei et al., 2014), we investigated the relationship between global gain modulation in model cortical circuits and variability in target network outputs. We found that the global gain of a recurrent neuronal network correlates positively with variability in the target output. That is, there are more suitable gains for different variabilities in target activity (independent of the readout weights), where a large gain is associated with large variability in the target. This suggests the possibility that the global gain of cortical motor circuits is modulated in accordance with the variability in the desired muscle activity; in line with experimental observations of gain changes in spinal motoneurons (Vestergaard and Berg, 2015). Additionally, our results align with experimental observations that dopamine (which can affect neuronal gain (Hernandez-Lopez et al., 2000; Thurley et al., 2008)) controls the vigour of movements (Niv et al., 2007; Panigrahi et al., 2015) and that decreased levels of dopamine lead to deficits in fine-tuned motor control (as is the case in Parkinson's disease (Damier et al., 1999; Panigrahi et al., 2015)).

We used two different, commonly adopted, gain functions: a $\tanh$ function and a linear variant. However, as we state in the proof of our analytical result in Appendix 2.B, we define the gain as the slope of the gain function at $\boldsymbol{x} = \mathbf{0}$. Therefore, our stability condition Eqn. (2.3) guarantees local asymptotic stability for any (at least locally) smooth function that has a nonzero derivative at $\boldsymbol{x} = \mathbf{0}$. Therefore, our stability condition Eqn. (2.3) holds for a much larger class of functions than the ones that we tested (for example, supralinear input–output functions may exist in sensory cortices (Hennequin et al., 2018; Rubin et al., 2015)). As an example of

such a (rectified) supralinear gain function, one could use $f(x_i) = g(x_i + \sqrt{r_0})^2 - r_0$ for $x_i > -\sqrt{r_0}$ and $f(x_i) = 0$ otherwise. For this function, following our derivation in Appendix 2.B, the condition for local asymptotic stability is $g < 1/(2\sqrt{r_0}\alpha(\boldsymbol{W}))$. However, we emphasise that for such an expansive nonlinear function, neuronal activity that is too far away from $\mathbf{0}$ may not return to $\mathbf{0}$ (i.e., the basin of attraction to $\mathbf{0}$ is likely small).

When simulating neuronal dynamics using stability-optimised circuits, we used the initial condition that maximises the evoked energy (see Eqn. (1.17)) of the corresponding linear neuronal dynamics. In future studies, one could investigate the effects of using different initial conditions and whether they can produce better fits for different target output activities. For example, it is plausible that low-frequency dynamics may more easily generate low-frequency muscle activity whereas it may be more difficult for a linear combination of high-frequency neuronal dynamics to produce a desired low-frequency output.

In this chapter, we only studied the effects of identically modulating the gain of all neurons in recurrent neuronal networks. In Chapter 3, we study the effects of independently modulating the gain of single neurons.

2.A Definitions and theorems

In this appendix, we give any mathematical definitions that we have used in this chapter, as well as definitions that are necessary for the proofs that we show in Appendix 2.B.

Definition 1 (Evolution operator)**.** *Consider the linear differential equation*

$$\frac{d\boldsymbol{x}(t)}{dt} = \boldsymbol{A}\boldsymbol{x}(t)\,. \tag{2.9}$$

The unique solution to Eqn. (2.9) *can be written in the form*

$$\boldsymbol{x}(t) = \boldsymbol{T}(t,s)\boldsymbol{x}(s)\,, \tag{2.10}$$

where $\boldsymbol{T}(t,s)$ *is the associated* evolution operator *of Eqn.* (2.9) *(Pereira, 2011). (The matrix* $\boldsymbol{T}(t,s)$ *is also called the* fundamental matrix *Teschl (2012).)*

Definition 2 (Uniform contraction)**.** *Let* $\boldsymbol{T}(t,s)$ *be the evolution operator associated with Eqn.* (2.9)*. The operator* $\boldsymbol{T}(t,s)$ *is said to be a* uniform contraction *if*

$$\|\boldsymbol{T}(t,s)\| \leq Ce^{-\eta(t-s)}\,,$$

where C *and* η *are positive constants (Pereira, 2011).*

Definition 3 (Global asymptotic stability)**.** *If the largest real part in the spectrum of* $\boldsymbol{A}$ *in the linear system Eqn.* (2.9) *is less than* 0 *(i.e.,* $\alpha(\boldsymbol{A}) < 0$*; see Eqn.* (1.11)*), then the origin is* globally asymptotically stable *and* $\boldsymbol{x} \to \boldsymbol{0}$ *as* $t \to \infty$*.*

Remark 1. *The trivial solution of Eqn.* (2.9) *is globally asymptotically stable if and only if the evolution operator is a uniform contraction (Pereira, 2011, Thm. 4).*

Theorem 1. *Assume that the matrix* $\boldsymbol{T}(t,s)$ *of Eqn.* (2.10) *has a uniform contraction. Consider the perturbed equation*

$$\frac{d\boldsymbol{y}(t)}{dt} = \boldsymbol{A}\boldsymbol{y}(t) + \boldsymbol{R}(\boldsymbol{y}(t))\,,$$

and assume that

$$\|\boldsymbol{R}(\boldsymbol{y})\| \leq M\|\boldsymbol{y}\|^{1+c}$$

for some $M, c > 0$*. The origin is then locally asymptotically stable.*

We can use Theorem 1 to prove that if the linearisation of a nonlinear system is asymptotically stable about the origin, then the linearised system describes the local behaviour of the nonlinear system near the origin (Pereira, 2011).

Finally, we state the following result from dynamical-systems theory (Teschl, 2012) that plays a central role in our analysis.

Theorem 2. *Equation.* (2.9) *is globally asymptotically stable if and only if* $\alpha(\boldsymbol{A}) < 0$*. Moreover, there is a positive constant* κ *for every* $a < -\alpha(\boldsymbol{A})$ *such that*

$$\|e^{t\boldsymbol{A}}\| \leq \kappa e^{-ta}\,, \quad t \geq 0\,.$$

2.B Main analytical result from Chapter 2 and proof

In this section, we present our theorem that guarantees that the trivial solution of Eqn. (2.1) is locally asymptotically stable.

Theorem 3 (Local stability of neuronal activity in recurrent neuronal networks)**.** *If the gain function f in Eqn. (2.1) is a differentiable (possibly nonlinear) function (with $f(\mathbf{0}) = \mathbf{0}$ and $f'(\mathbf{0}) = g \neq 0$), then if $\alpha(\boldsymbol{W}) < g^{-1}$, as $t \to \infty$, we have*

$$\|\boldsymbol{x}(t)\| \leq \frac{C\|\boldsymbol{h}\|}{\eta}\,, \tag{2.11}$$

for sufficiently small $\|\boldsymbol{h}\|$, where $\|\boldsymbol{h}\| = \sup_t \|\boldsymbol{h}(t)\|$, and where $C > 0$, $\eta < 1 - g\alpha(\boldsymbol{W})$.

Theorem 3 guarantees that the trivial solution to Eqn. (2.1) is locally asymptotically stable if the external input is sufficiently small. We now state a useful corollary of this result.

Corollary 1 (Linear gain function)**.** *If there is no external input — i.e., $\boldsymbol{h} = \mathbf{0}$ and the gain function $f(\boldsymbol{x})$ is such that $f(\boldsymbol{x}) = g\boldsymbol{x}$ for $g \in \mathbb{R}$ — then the trivial solution to Eqn. (2.1) is globally asymptotically stable if and only if $\alpha(\boldsymbol{W}) < g^{-1}$. Moreover, there exists $K > 0$ such that*

$$\|\boldsymbol{x}(t)\| \leq Ke^{-\eta t/\tau}\,, \quad t \geq 0\,, \tag{2.12}$$

where $\eta < 1 - g\alpha(\boldsymbol{W})$.

Proof of Theorem 3. For some t such that $\|\boldsymbol{x}(t)\|$ is sufficiently small, we can rewrite Eqn. (2.1) by expanding the gain function $f(\boldsymbol{x})$ in a Taylor series and rearranging terms to obtain

$$\begin{aligned} \tau\frac{d\boldsymbol{x}}{dt} &= \boldsymbol{W}(f(\mathbf{0}) + f'(\mathbf{0})\boldsymbol{x}(t) + \mathcal{O}(\boldsymbol{x}(t))^2) - \boldsymbol{x}(t) + \boldsymbol{h}(t) \\ &= [\boldsymbol{W}f'(\mathbf{0}) - \boldsymbol{I}]\boldsymbol{x}(t) + \boldsymbol{h}(t) + \boldsymbol{R}(\boldsymbol{x})\,, \end{aligned} \tag{2.13}$$

where $\|\boldsymbol{R}(\boldsymbol{x})\| = \mathcal{O}(\boldsymbol{x}^2)$. We now truncate our Taylor expansion to first order to obtain

$$\tau \frac{d\boldsymbol{x}}{dt} = [\boldsymbol{W}g - \boldsymbol{I}]\boldsymbol{x}(t) + \boldsymbol{h}(t)\,, \tag{2.14}$$

where $g = f'(\boldsymbol{0})$.

For the homogeneous part of Eqn. (2.14),

$$\tau \frac{d\boldsymbol{x}}{dt} = [\boldsymbol{W}g - \boldsymbol{I}]\boldsymbol{x}(t)\,, \tag{2.15}$$

we can construct a solution in terms of an evolution operator $\boldsymbol{T}(t,s)$, as stated in Definition 1. Specifically, we can write

$$\boldsymbol{x}(t) = \boldsymbol{T}(t,s)\boldsymbol{x}(s)\,. \tag{2.16}$$

From integrating Eqn. (2.15), we find that $\boldsymbol{T}(t,s) = e^{-\frac{(t-s)}{\tau}(\boldsymbol{W}g-\boldsymbol{I})}$. However, for our purposes, we simply need to obtain a bound on $\boldsymbol{T}(t,s)$. For $\boldsymbol{T}(t,s)$ to be a uniform contraction, which implies that the origin is locally asymptotically stable, we must have $\alpha(\boldsymbol{W}g - \boldsymbol{I}) < 0$ because of Theorem 2. This then implies that $\alpha(\boldsymbol{W}g) < 1$ (see Lemma 1 in Appendix A where we discuss how subtraction of a diagonal matrix (e.g., $\boldsymbol{I}$) affects the spectrum of the resulting matrix). Finally, because $\alpha(\boldsymbol{W}g) = g\alpha(\boldsymbol{W})$, as g is a scalar, we obtain the inequality $g < 1/\alpha(\boldsymbol{W})$ guaranteeing local asymptotic stability. This yields the bound

$$\|\boldsymbol{T}(t,s)\| \leq Ce^{-\eta(t-s)/\tau}, \tag{2.17}$$

where $C > 0$ and $\eta < 1 - g\alpha(\boldsymbol{W})$.

Note that we truncated our Taylor expansion up to first order in Eqn. (2.14). We therefore have to check that the higher-order terms do not affect the stability of the trivial solution. From Theorem 1, we know that because $\|\boldsymbol{R}(\boldsymbol{x})\| = \mathcal{O}(\boldsymbol{x}^2)$ in Eqn. (2.13), so the stability of the trivial solution is not affected by these higher-order terms and the origin is asymptotically stable. It remains to study the inhomogeneous Eqn. (2.14) due to external input $\boldsymbol{h}(t)$.

Using variation of parameters (Teschl, 2012, Section 3.4) (which is also called variation of constants), in Eqn. (2.14), we obtain

$$\boldsymbol{x}(t) = \boldsymbol{T}(t,0)\boldsymbol{x}_0 + \frac{1}{\tau}\int_0^t \boldsymbol{T}(t,u)\boldsymbol{h}(u)du\,, \tag{2.18}$$

where $\boldsymbol{T}(t,s)$ is the evolution operator corresponding to the homogeneous solution (2.16). We note that for a tonic input $\boldsymbol{h}(t) = \boldsymbol{h}$, and using $\boldsymbol{T}(t,s) = e^{-\frac{(t-u)}{\tau}(\boldsymbol{W}g-\boldsymbol{I})}$ (see above), for long times ($t \to \infty$), we obtain $\boldsymbol{x}(t) \to (\boldsymbol{W}g - \boldsymbol{I})^{-1}\boldsymbol{h}$.

For a time-dependent input $\boldsymbol{h}(t)$, using the triangle inequality and our bounds on the evolution operator in Eqn. (2.17), for sufficiently small $\|\boldsymbol{h}\|$, where we define $\|\boldsymbol{h}\| = \sup\{\|\boldsymbol{h}(u)\| : u \leq t\}$, it follows that

$$\begin{aligned}
\|\boldsymbol{x}(t)\| &\leq \|\boldsymbol{T}(t,0)\|\|\boldsymbol{x}_0\| + \frac{1}{\tau}\int_0^t \|\boldsymbol{T}(t,u)\|\|\boldsymbol{h}(u)\|du \\
&\leq Ce^{-\eta t/\tau}\|\boldsymbol{x}_0\| + \frac{1}{\tau}\int_0^t Ce^{-\eta(t-u)/\tau}\|\boldsymbol{h}\|du \\
&= Ce^{-\eta t/\tau}\|\boldsymbol{x}_0\| + C\|\boldsymbol{h}\|\Big[\frac{1-e^{-\eta t/\tau}}{\eta}\Big]\,.
\end{aligned}$$

Letting $t \to \infty$ and using $\eta < 1 - g\alpha(\boldsymbol{W})$, we obtain

$$\|\boldsymbol{x}(t)\| \leq \mu\|\boldsymbol{h}\|\,,$$

where $\mu = C/\eta + o(1)$. This concludes the proof of Theorem 3. □

We now briefly explain how Corollary 1 is a direct result of Theorem 3.

Proof of Corollary 1. For a linear gain function $f(\boldsymbol{x}) = g\boldsymbol{x}$, in the absence of external input (i.e., $\boldsymbol{h} = \boldsymbol{0}$), we obtain the following homogeneous linear differential equation:

$$\tau\frac{d\boldsymbol{x}}{dt} = [\boldsymbol{W}g - \boldsymbol{I}]\boldsymbol{x}(t)\,. \tag{2.19}$$

Note from Eqn. (2.15) that $g < 1/\alpha(\boldsymbol{W})$ is necessary for the origin to be asymptotically stable. Furthermore, we can bound the associated evolution operator $\boldsymbol{T}(t,0)$ of the solution to Eqn. (2.19),

$$\boldsymbol{x}(t) = \boldsymbol{T}(t,0)\boldsymbol{x}_0\,,$$

as

$$\|\boldsymbol{T}(t,0)\| \leq Ce^{-\eta t/\tau}\,,$$

where $C > 0$ and $\eta < 1 - g\alpha(\boldsymbol{W})$. We then have

$$\|\boldsymbol{x}(t)\| \leq C'e^{-\eta t/\tau}, \quad t \geq 0\,,$$

where $C' = C\|\boldsymbol{x}_0\|$. □

2.C Supplementary figures

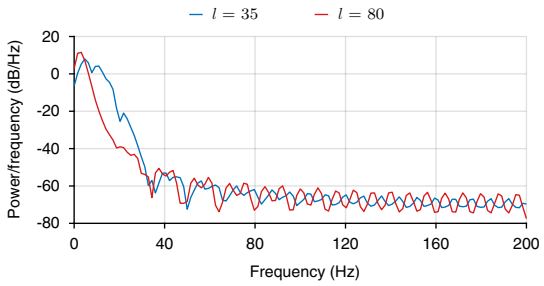

Figure 2.7: We plot the power spectrum of the target muscle activity (for both $l = 35$ ms and $l = 80$ ms) that we show in Fig. 2.5 and which we use in Fig. 2.6. Lower values of l in Eqn. (2.7) correspond to more 'variable' muscle activity, which is also reflected in its power spectrum.

CHAPTER 3

Single-neuron gain modulation

In this chapter, we study the effects of independently modulating the gain of individual neurons — which we call *single-neuron gain modulation* — in recurrent neuronal networks. In contrast to identically changing the gain of all neurons (which we explored in Chapter 2), we find that independently modulating each neuron's gain enables a much greater variety of neuronal activities. We show that by changing the gain of even only one neuron in a recurrent neuronal network, we can substantially reduce errors between a network output and a target. We also introduce a reward-based learning rule that reduces errors in network outputs by iteratively changing each neuron's gain. Finally, we demonstrate the effectiveness of our learning rule in various different models. The results that we present from Section 3.4 to Section 3.7 form part of an article that has been accepted at Nature Neuroscience: *Jake P. Stroud, Mason A. Porter, Guillaume Hennequin, and Tim P. Vogels, 'Motor primitives in space and time via targeted gain modulation in cortical networks'*.

3.1 Introduction

There have been many studies that examine how synaptic modifications in neuronal networks can reduce network output errors (Hoerzer et al., 2014; Legenstein et al., 2010; Miconi, 2017; Sussillo and Abbott, 2009; Sussillo et al., 2015). Such approaches have received much attention because synaptic changes are thought to underlie learning for many types of tasks, including motor tasks (which is the main topic of study in this thesis) (Hosp et al., 2011; Kida and Mitsushima, 2018; Molina-Luna et al., 2009; Peters et al., 2014; Rosenbaum, 2009; Sanes and Donoghue, 2000). However, some recent experimental evidence suggests that there are few synaptic changes in both premotor and primary motor cortices in macaque monkeys on the time scale of a single experimental session when learning new motor tasks (Perich et al., 2017; Sadtler et al., 2014). Additionally, it can be difficult for motor cortex to generate certain new patterns of neural activity, at least on the time scale of hours (Golub et al., 2018; Sadtler et al., 2014). This is thought to be the case because pre-existing connectivity in motor cortex constrains the possible patterns of neural activities that can be generated (Sadtler et al., 2014). Therefore, based on these results, there may be few synaptic changes in motor cortices on short time scales when learning new movements.

Other experimental studies suggest that movement-specific neural activity in motor cortex is generated from movement-specific preparatory states that set the initial condition of the neuronal activity (Churchland et al., 2010, 2012; Shenoy et al., 2013). This suggests the possibility that when learning new movements, a new initial condition is created to generate the required neuronal activity while the network connectivity remains fixed. However, changing only the initial condition of the neuronal activity in cortical motor circuits may limit the variety of possible movements that can be generated (for example, the time scale of decay of neuronal activity is unaffected by changes in the initial condition).

Alternatively, changes can also occur in the input–output gain of neurons during motor learning both in motor cortex (Kida and Mitsushima, 2018) and in downstream spinal motoneurons (Vestergaard and Berg, 2015; Wei et al., 2014). Gain modulation is also a possible mechanism to generate new patterns of neural activity while circuit connectivity remains fixed (Salinas and Sejnowski, 2001; Salinas and Thier, 2000; Swinehart et al., 2004; Zhang and Abbott, 2000). However, there have been relatively few theoretical studies of gain modulation in neuronal networks. One such study demonstrated that supervised training of neuronal gains in the input layer of a feedforward network can enable learning of a variety of target outputs (Swinehart et al., 2004). In another study, it was shown that independently modulating the gain of neurons in a recurrent neuronal network can allow the output of a network to change substantially (Zhang and Abbott, 2000). However, it is unclear what the effects of single-neuron gain modulation may be in biologically-motivated recurrent neuronal-network models.

In this chapter, we study the possibility of using single-neuron gain modulation in recurrent neuronal networks with fixed connectivity to generate desired target outputs. In Section 3.3, we investigate the effectiveness of changing the gain of only one neuron in a recurrent neuronal network at reducing network output errors. In Section 3.4, we introduce a reward-based learning rule for independently changing the gain of each neuron in a recurrent neuronal network so that network output errors are reduced. In Sections 3.5–3.7, we demonstrate the effectiveness of the learning rule in a variety of models.

3.2 Methods

We use the same model as the one that we discussed in Sections 1.2.1.3 and 1.2.4. For convenience, we reproduce our full model description below.

3.2.1 Neuronal dynamics

We study recurrent neuronal networks of $N = 2M$ neurons (of which M are excitatory and M are inhibitory) for which the neuronal activity $\boldsymbol{x}(t) = (x_1(t), \ldots, x_N(t))^\mathsf{T}$ evolves according to the dynamical system

$$\tau \frac{d\boldsymbol{x}(t)}{dt} = -\boldsymbol{x}(t) + \boldsymbol{W} f(\boldsymbol{x}(t); \boldsymbol{g}) \,, \tag{3.1}$$

from some initial condition $\boldsymbol{x}(0) = \boldsymbol{x}_0$. In Eqn. (3.1), $f(\boldsymbol{x}; \boldsymbol{g})$ denotes the element-wise application of the static scalar gain function f to the neuronal activity vector $\boldsymbol{x}$. In keeping with Hennequin et al. (2014), we set the single-neuron time constant to be $\tau = 200$ ms so that neuronal-activity transients have a similar duration to those observed experimentally during reaching tasks (Churchland et al., 2012). The gain function f, which governs the transformation of neuronal activity $\boldsymbol{x}$ into firing rates relative to a baseline rate r_0, is

$$f(x_i; g_i) = \begin{cases} r_0 \tanh(g_i x_i / r_0) \,, & \text{if } x_i < 0 \,, \\ (r_{\max} - r_0) \tanh(g_i x_i / (r_{\max} - r_0)) \,, & \text{if } x_i \geq 0 \,, \end{cases} \tag{3.2}$$

where the gain value g_i is the slope of the function f at $x_i = 0$. Therefore, g_i controls the input–output sensitivity of neuron i (Rajan et al., 2010). We use a baseline rate of $r_0 = 20$ Hz and a maximum firing rate of $r_{\max} = 100$ Hz so that the resulting neuronal firing rates are consistent with observations (Churchland et al., 2012; Kao et al., 2015; Lara et al., 2018) (but see Section 3.7 where we use a different baseline firing rate). As we mentioned in Section 1.2.1, $f(\boldsymbol{x}; \boldsymbol{g})$ describes the neuronal firing rates relative to the baseline rate r_0. (For a visualisation of f, see the curves in Fig. 1.3, where we plot f for two different values of the gain g_i.)

Unless we state otherwise, we generate the synaptic weight matrix $\boldsymbol{W}$ in Eqn. (3.1) in line with Hennequin et al. (2014). That is, we use stability-optimised circuits (see Section 1.2.4). We use the initial condition $\boldsymbol{x}(t = 0) = \boldsymbol{x}_0$ that maximises the evoked energy $\varepsilon(\boldsymbol{x}_0)$ in Eqn. (C.20). (See Appendix C.2 for a discussion of

how we obtain such an initial condition.) By choosing such a weight matrix and initial condition, the neuronal dynamics governed by Eqn. (3.1) display complex multiphasic activity transients; these are similar to those observed in motor cortex during movement execution (Hennequin et al., 2014) (see Sections 1.2.4 and 1.3).

3.2.2 Creating target network outputs reminiscent of muscle activity

To create target network outputs that resemble electromyogram (EMG) recordings of muscle activity, we draw muscle activities of duration $T = 500$ ms from a Gaussian process with a covariance function $K \in [0, T] \times [0, T] \to \mathbb{R}_{\geq 0}$ that consists of a product of a squared-exponential kernel (to enforce temporal smoothness) and a non-stationary kernel that produces a temporal envelope similar to that of real EMG data during reaching (Churchland et al., 2012). Specifically,

$$K(t, t') = e^{-\frac{(t-t')^2}{2\ell^2}} \times E(t/\sigma) \times E(t'/\sigma)\,, \tag{3.3}$$

with $E(t) = te^{(-t^2/4)}$. The parameter σ controls the duration of the generated muscle activity and ℓ controls the autocorrelation of the muscle activity. Based on test simulations, we find that $\sigma = 110$ ms and $\ell = 50$ ms generates realistic muscle activity, in line with experimental observations, that lasts approximately 500 ms (Churchland et al., 2012; Russo et al., 2018; Sussillo et al., 2015) (see Section 2.6 for a discussion of this). We also multiply the resulting muscle activity by a constant to ensure that it has the same order of magnitude as the neuronal activity, and we use a sampling rate of 400 Hz (i.e., target muscle activity is a vector of length 200).

In other words, target muscle activity that we generate corresponds to $\boldsymbol{Lu}$ where $\boldsymbol{LL}^\intercal = \boldsymbol{K}$, $\boldsymbol{K} \in \mathbb{R}^{200\times 200}$ is the covariance function, $\boldsymbol{u} \in \mathbb{R}^{200}$, and $\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ (i.e., elements of the vector $\boldsymbol{u}$ are drawn independently and identically from a Gaussian distribution with mean 0 standard deviation 1).

3.2.3 Network output

Throughout this thesis, we generate network outputs as weighted linear combinations of excitatory neuronal firing rates. A similar approach for generating network outputs has been used many times previously (Hennequin et al., 2014; Hoerzer et al., 2014; Russo et al., 2018; Sussillo and Abbott, 2009; Sussillo et al., 2015; Wang et al., 2018).

We compute the network output activity $z(t)$ at time t as

$$z(t) = \boldsymbol{m}^{\mathsf{T}} f(\boldsymbol{x}^E(t); \boldsymbol{g}^E) + b\,, \tag{3.4}$$

where $\boldsymbol{m}\,, \boldsymbol{x}^E(t)\,, \boldsymbol{g}^E \in \mathbb{R}^M$, the quantity M is the number of excitatory neurons, $\boldsymbol{x}^E(t)$ is the excitatory neuronal activity, and f is the gain function (see Eqn. (3.2)). We fit the readout weights $\boldsymbol{m}$ and the offset b to an initial target output (see the right-hand side of Fig. 3.1a; also see each subsequent section for further details) using least-squares regression with all gains set to 1. To ameliorate any issues of overfitting, we use 100 noisy trials, in which we add Gaussian white noise to the initial condition $\boldsymbol{x}_0$ for each trial with a signal-to-noise ratio of 30 dB (Hennequin et al., 2014).

3.3 Effects of changing the gain of one neuron in a recurrent neuronal network

In this section, we investigate the effects of changing the gain of one neuron in a 200-neuron network. Starting with all gains set to 1, we fit the readout weights so that the network output generates a target output (see Fig. 3.1a and Section 3.2.3). We find that doubling the gain of one neuron from 1 to 2 causes an increase in the amplitude and frequency of its firing-rate activity, but the other neurons' firing rates appear to change only by a comparatively small amount (see Fig. 3.1b). The observation of an increase in the frequency and amplitude of the firing rate of the

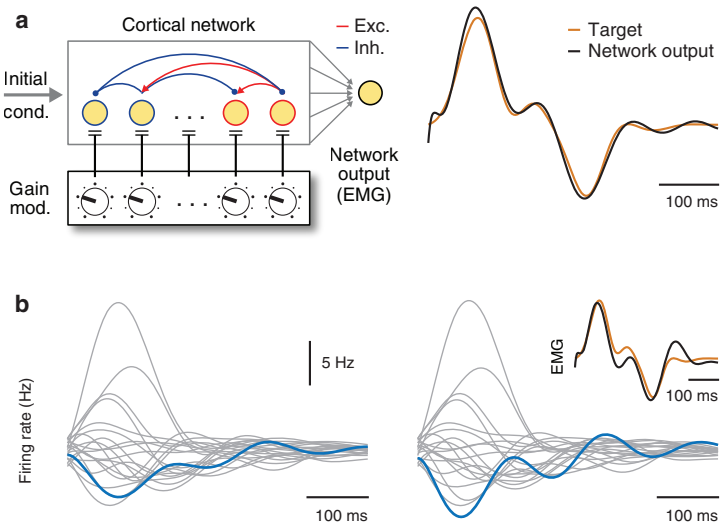

Figure 3.1: Effects of modulating the gain of only one neuron in a recurrent neuronal network. **a**, *Illustration of the our model setup. On the left, we show that we can control the gain of individual neurons in a recurrent neuronal network. On the right, we show the network output in black and the target output in orange (see Section 3.3). The network output acts as a proxy for muscle electromyogram (EMG) activity (see Section 3.2.2).* **b**, *We plot the firing rates of* 20 *neurons in grey, with (left) all gains set to* 1 *and (right) after setting the gain of one neuron to* 2*. The blue curves show the firing rate of the neuron that we gain modulated. In the inset on the far right, the black curve shows the network output after setting the gain of the afforementioned neuron to* 2*. In orange, we also show the same target that we showed in panel (a).*

gain-modulated neuron is in line with the effects that we found previously in Section 2.5. We also note that the network output is altered after modulating the gain of the aforementioned neuron. (See the inset in the top right of Fig. 3.1b.)

We now ask the following question: Can we substantially reduce the error between the network output and a new target output by modulating the gain of only one neuron? (See Eqn. (D.1) for how we calculate errors.) For the same weight matrix and readout weights that we used in Fig. 3.1, we generate a new target (which we call target A) and we separately change the gain of each neuron, one at a time, while keeping the gain of each other neuron fixed at 1. We find that for one

specific neuron, increasing its gain to approximately 5.5 maximally (across all neurons and tested gains) reduces the error between the network output and target A (see the blue curve in Fig. 3.2a). By setting this neuron's gain to approximately 5.5, the network output is altered substantially so that it is closer to target A. (Compare the black and blue curves Fig. 3.2c.)

We also generate another target, which we call target B, and we find that another specific neuron and gain maximally (across all neurons and tested gains) reduces the error between the network output and target B (see the red curve in Fig. 3.2d). In fact, setting this neuron's gain to 0 (i.e., effectively removing this neuron from the network) maximally reduces the error between the network output and target B. (See the minimum of the red curve in Fig. 3.2d.) Interestingly, the two neurons that maximally reduce the output error for targets A and B, respectively, also substantially reduce the error for targets B and A, respectively (see Figs. 3.2a,b,d,e).

However, we find that the majority of neurons do not substantially reduce the error between a network output and a target (see Figs. 3.2b,e). To illustrate this, we plot the error versus neuronal gain for a neuron chosen uniformly at random (see the yellow curves in Figs. 3.2a,d). We find that this neuron does not reduce the error substantially for any tested gain value and for either target.

3.4 A reward-based learning rule for single-neuron input–output gains

In this section, we ask the following question: Can we accurately generate a novel target output by independently modulating the gain of all neurons simultaneously? After our initial investigations into single-neuron gain modulation in Section 3.3, we now use a biologically-inspired, reward-based learning rule that iteratively changes each neuron's gain so that the error between the network output and a target is

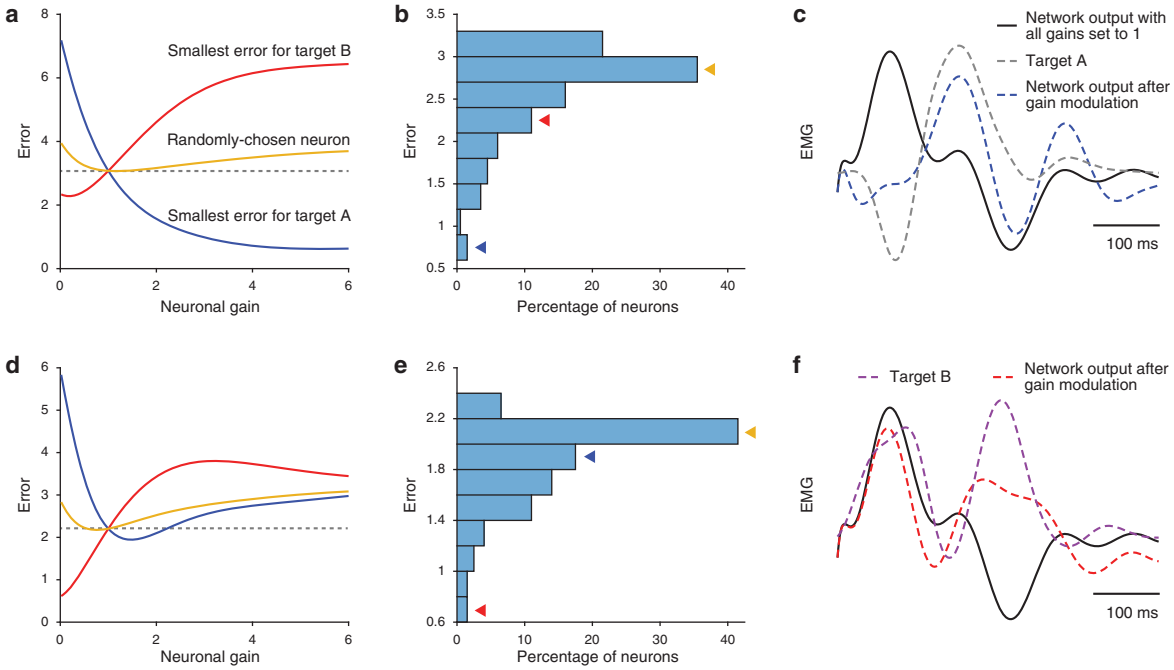

Figure 3.2: Reducing network output errors by modulating the gain of only one neuron in a recurrent neuronal network. **a**, *We plot the error between the network output and target* A *(see the main text) versus neuronal gain for* 3 *neurons. The blue curve shows the error for the neuron that maximally (across all neurons and tested gains) reduces the error for target* A. *The red curve shows the error for the neuron that maximally (across all neurons and tested gains) reduces the error for target* B *(see the main text). The yellow curve shows the error for a neuron chosen uniformly at random. The grey dashed line identifies the error with all gains set to* 1. **b**, *Histogram of the minimum errors for all neurons for target* A. *We indicate the minimum error for each of the* 3 *neurons that we showed in panel (a) with the coloured arrowheads.* **c**, *We show the network output with all gains set to* 1 *(black curve), target* A *(grey dashed curve), and the network output after setting the gain of a neuron so that the error between the network output and target* A *is reduced maximally (blue dashed curve).* **d**–**e**, *The same as panels (a)–(b), respectively, except that we show the errors between the network output and target* B. **f**, *We show the initial network output with all gains set to* 1 *(black curve), target* B *(purple dashed curve), and the network output after setting the gain of a neuron so that the error between the network output and target* B *is reduced maximally (red dashed curve).*

reduced (on average) over training iterations. We adopt this approach for two primary reasons: We want an effective method for simultaneously changing multiple neurons' gains so that network output errors are reduced; and we want to use an approach that is biologically plausible.

Most theoretical studies of reward-based learning have investigated synaptic changes (Frémaux and Gerstner, 2016; Hoerzer et al., 2014; Legenstein et al., 2010; Mazzoni et al., 1991; Miconi, 2017; Williams, 1992), rather than changes in neuronal responsiveness. However, in motor control, the input–output gain of motor neurons can change (Kida and Mitsushima, 2018; Vestergaard and Berg, 2015). It has also been demonstrated experimentally that neuromodulators such as serotonin and dopamine, which are involved in reward-based learning (Boureau and Dayan, 2011; Cools et al., 2011; Doya, 2002; Hosp et al., 2011; Luft and Schwarz, 2009; Molina-Luna et al., 2009), can affect neural input–output sensitivity (Hernandez-Lopez et al., 2000; Thurley et al., 2008; Wei et al., 2014).

We devise a reward-based node-perturbation learning rule (Mazzoni et al., 1991) that includes only local information and a single binary reward signal that reflects a system's recent performance. We update the gain g_i for neuron i after each training iteration t_n (with $n = 1, 2, 3, \ldots$) according to the following learning rule:

$$g_i(t_n) = g_i(t_{n-1}) + R(t_{n-1})(g_i(t_{n-1}) - \bar{g}_i(t_{n-1})) + \xi_i(t_n)\,, \tag{3.5}$$

where

$$\begin{aligned} R(t_n) &= \operatorname{sgn}(\bar{\epsilon}(t_{n-1}) - \epsilon(t_n))\,, \\ \bar{\epsilon}(t_n) &= \alpha\bar{\epsilon}(t_{n-1}) + (1-\alpha)\epsilon(t_n)\,, \\ \bar{g}_i(t_n) &= \alpha\bar{g}_i(t_{n-1}) + (1-\alpha)g_i(t_n)\,, \end{aligned} \tag{3.6}$$

where $\epsilon(t_n)$ represents the output error at iteration t_n (see Eqn. (D.1) for how we calculate errors), sgn is the sign function, $\xi_i(t_n) \sim \mathcal{N}(0, 0.001^2)$ is a Gaussian random variable with mean 0 and standard deviation 0.001, and $\alpha = 0.3$. The initial modula-

tory signal is $R(t_0) = 0$, and the other initial conditions are $\bar{\epsilon}(t_0) = \epsilon(t_0)$ (where $\epsilon(t_0)$ is the error before training; i.e., with all gains set to 1) and $\bar{g}_i(t_0) = g_i(t_0) = 1$. One can interpret the terms $\bar{g}_i$ and $\bar{\epsilon}$ as low-pass-filtered gains and errors, respectively, over recent iterations with a history controlled by the decay rate α (Miconi, 2017). We use these parameter values in our learning rule for all of our simulations. We find that varying the standard deviation of the noise term ξ or the factor α has little effect on the learning dynamics, in line with Hoerzer et al. (2014).

Although our learning rule in Eqn. (3.5) is similar to reward-modulated 'exploratory Hebbian' (EH) synaptic plasticity rules (Hoerzer et al., 2014; Legenstein et al., 2010; Miconi, 2017), we investigate changes in neuronal gains (i.e., the responsiveness of neurons) inside a recurrent neuronal network rather than synaptic weight changes. The above notwithstanding, we expect our learning rule to perform well for a variety of learning problems. For example, it can solve credit-assignment problems, because one can formulate such a node-perturbation learning rule as reinforcement learning with a scalar reward (Saito et al., 2011).

The modulatory signal R does not provide information about the sign and magnitude of the error, and it also does not indicate the amount that each readout (if using multiple readouts) contributes to a recent change in performance. The modulatory signal R indicates only whether performance is better or worse, on average, compared with previous trials. One can view the modulatory signal as an abstract model for phasic output of dopaminergic systems in the brain (Frémaux and Gerstner, 2016; Hosp et al., 2011; Huntley et al., 1992; Molina-Luna et al., 2009).

We use the following procedure for updating neuronal gains. We update the gains for iteration t_1 according to Eqn. (3.5), and we obtain the network output from the gain pattern $\boldsymbol{g}(t_1)$. We then calculate the error $\epsilon(t_1)$ from the output, and we subsequently calculate the modulatory signal $R(t_1)$ and the quantities $\bar{\epsilon}(t_1)$ and $\bar{\boldsymbol{g}}(t_1)$ using Eqn. (3.6). We then repeat this process for all subsequent iterations.

3.5 Learning novel network outputs through gain modulation

To demonstrate the effectiveness of the learning rule in Eqn. (3.5), we use a 200-neuron stability-optimised circuit (see Section 3.2). As in Section 3.3, we fit the readout weights so that, with all gains set to 1, the network output is similar to real EMG muscle activity when executing a movement. (See the solid black curve in Fig. 3.3a and see Section 3.2.) We then train the neuronal gains using the learning rule in Eqn. (3.5) so that the model generates a new target output (see the solid orange curve in Fig. 3.3a). We find that errors between the network output and the target tend to decrease monotonically over training iterations, and a few thousand training iterations are required for the error to saturate to a small value (see Fig. 3.3b).

During training, the neuronal gains change in a noisy manner because of the noise term in Eqn. (3.5) (see Fig. 3.3c). After training, the distribution of neuronal gains closely follows a Gaussian distribution (see Fig. 3.3c); and the same initial condition for the neuronal activity can produce either of two distinct outputs, depending on the applied gain pattern (see Fig. 3.3d).

Although the network output is substantially altered after training, the neuronal firing rates change only slightly (see Fig. 3.4a). Independent training sessions on the same target produce nonidentical but correlated gain patterns (see Fig. 3.4b). The learned gain patterns are also similarly robust to noisy initial conditions compared to the case of all gains set to 1 (see Fig. 3.4c), except for large signal-to-noise ratios (i.e., approximately larger than 25 dB). See Appendix 3.A for further details.

We also compare the performance of training through gain modulation with 3 alternative training approaches. We train either the neuronal gains, the initial condition $\boldsymbol{x}_0$, the recurrent weight matrix $\boldsymbol{W}$ in Eqn. (3.1), or a rank-1 perturbation of the recurrent weight matrix using back-propagation. (See Appendix 3.A for further details.) We find empirically for this task, that training the neuronal gains provides

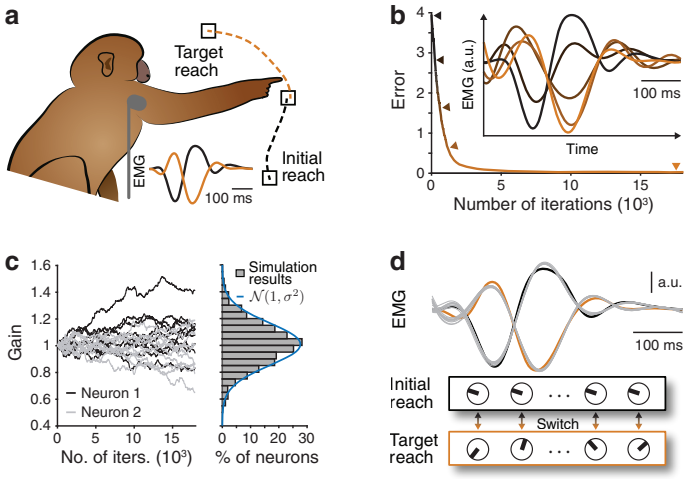

Figure 3.3: Learning new network outputs through neuron-specific gain modulation. **a**, *We show the network output with all gains set to* 1 *(solid black curve) and the target network output (solid orange curve). (See the main text.) These network output activities act as proxies for electromyogram (EMG) muscle activity when executing a movement, such as reaching.* **b**, *The mean (over* 10 *independent training sessions) error in network output decreases during training with neuron-specific modulation. In the inset, we show five snapshots of network output (indicated by arrowheads) as learning progresses.* **c**, *(Left) Neuronal gain changes during training for* 2 *example neurons (grey and black) and* 10 *training sessions. (Right) Histogram of gain values after training. The blue curve is a Gaussian fit with a mean of* 1 *and a standard deviation of* $\sigma \approx 0.157$. **d**, *Network outputs (grey curves) with all gains set to* 1 *and the new learned gain pattern (which we compare to both targets, given in black and orange) for* 10 *noisy initial conditions with noise added to the initial condition* x_0 *with a signal-to-noise ratio of* 30 *dB.*

a similar learning performance (i.e., error reduction) as training the initial condition or the recurrent weight matrix (see Fig. 3.4d). However, training a rank-1 perturbation of the weight matrix performs much worse than the 3 aforementioned training approaches.

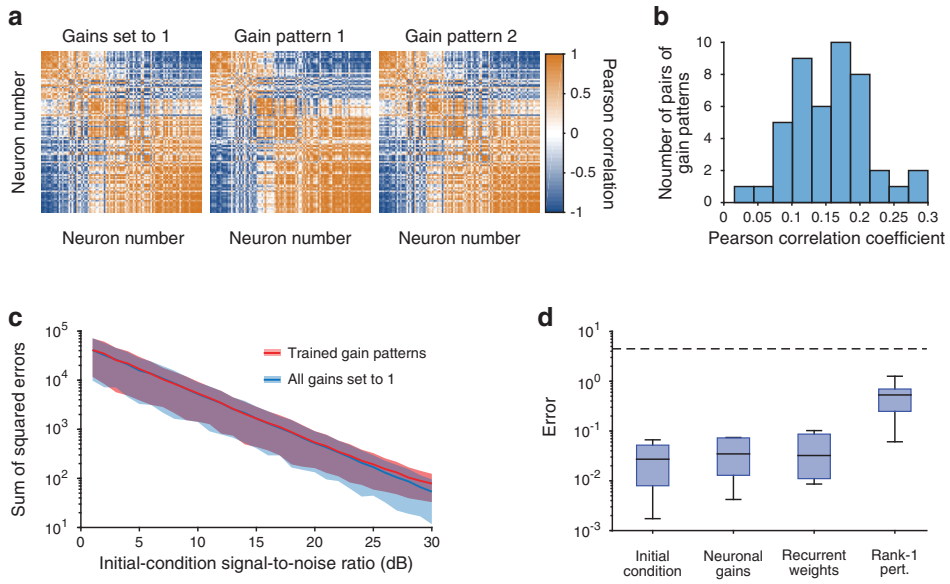

Figure 3.4: **a**, *Correlation matrices of the firing rates for all pairs of neurons with (left) all gains set to* 1 *and (centre and right) two (of the ten) trained gain patterns for the task in Fig. 3.3. The order of neurons is the same in all three matrices. As a result of training, there is not a substantial reorganisation in Pearson correlations between pairs of neurons.* **b**, *Histogram of the Pearson correlation coefficients between the* 45 *possible pairs of the* 10 *trained gain patterns that we obtained for the task that we showed in Fig. 3.3.* **c**, *Error between the network output with white Gaussian noise added to the initial condition* $\boldsymbol{x}_0$ *and the network output without noise added to* $\boldsymbol{x}_0$ *for various strengths of the signal-to-noise ratio. We plot results with all gains set to* 1 *(blue) and the* 10 *trained gain patterns (red) for the task in Fig. 3.3. Shading indicates* 1 *standard deviation. (See Appendix 3.A for further details.)* **d**, *Box plot of the errors after training independently on* 10 *different target movements using back-propagation when training either the initial condition, the neuronal gains, the recurrent weight matrix, or a rank-*1 *perturbation of the recurrent weight matrix (see Appendix 3.A). The dashed black line is the mean error before training. We use Tukey style for the whiskers.*

3.6 Learning through gain modulation in different models

We have shown that gain modulation can provide an effective mechanism of generating new outputs when we use a particular model (which we described in Sections 3.2 and 3.4). We now demonstrate that learning through gain modulation is also possible in other, commonly-used variants of our model (Hennequin et al., 2014; Sussillo et al., 2015). We train neuronal gains on the same task as the one that we showed in Fig. 3.3 using 3 alternative models. We fit the readout weights so that prior to any training (i.e., with all gains set to 1), the network output is the same in each model (see the solid black curve in Fig. 3.3a).

Motor circuits that drive movements also engage in periods of movement preparation (see Fig. 1.7) (Churchland et al., 2010; Li et al., 2015; Shenoy et al., 2013), suggesting a role for gain modulation in shaping circuit dynamics both during movement planning and during movement execution. We simulate preparatory activity using a ramping input to the system (3.1) (Hennequin et al., 2014), such that gain modulation now directly affects the neuronal activity at the time $t = 0$ (i.e., at movement onset). We use the same ramping input function as the one that was used in Hennequin et al. (2014). It is $\exp(t/\tau_{\text{on}})$ for $t < 0$ and $\exp(-t/\tau_{\text{off}})$ after movement onset (i.e., $t \geq 0$), with an onset time of $\tau_{\text{on}} = 400$ ms and an offset time of $\tau_{\text{off}} = 2$ ms. We find that learning performance (i.e., error reduction) for the task that we showed in Fig. 3.3 is slightly poorer if we do employ a ramping input than if we do not. (Compare the red and blue curves in Fig. 3.5a.) This occurs because gain modulation during the preparatory phase changes the neuronal activity at movement onset, allowing it to leave the null space of the readout weights (which are fixed) and thus elicit premature muscle activity at movement onset.

We also construct a 'chaotic' variant of our model (Sussillo and Abbott, 2009). These networks are chaotic in the sense that the neuronal dynamics in Eqn. (3.1) have a positive maximum Lyapunov exponent (see Section 1.2.1.3) (Sompolinsky

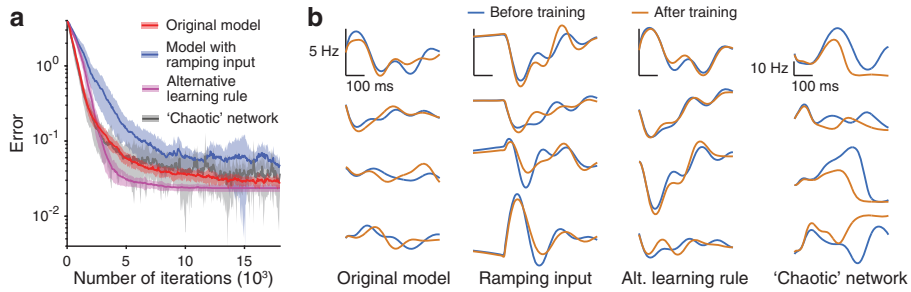

Figure 3.5: Learning through neuron-specific gain modulation in different models. **a**, *Mean error during training for our original model from Fig. 3.3 (red); the model with a biologically motivated ramping input (blue); the model when using the alternative learning rule (3.8), in which learning automatically stops at a sufficiently small error (purple); and when using a 'chaotic' recurrent network model (grey). Shading indicates one standard deviation.* **b**, *The firing rates of* 4 *example neurons before (i.e., with all gains set to* 1*) and after training the neuronal gains in (left) our original model, (centre left) our model with a ramping input, (centre right) our model with the alternative learning rule, and (right) the model when using a 'chaotic' network.*

et al., 1988). We use a synaptic weight matrix $\boldsymbol{W}$ (as described in Section 1.2.4) prior to optimisation, but now we use parameter values of $\gamma = 1$ and $\rho = 1.5$ (see Section 1.2.4). We also set $\tau = 20$ ms, and we choose the initial condition $\boldsymbol{x}_0$ from a uniform distribution on the interval $[-10, 10]$. We use only the first 0.5 s of neuronal activity, and we train the neuronal gains in this chaotic variant of our model on the same task that we described in the previous paragraph. We achieve similar learning performance compared to our original model from Fig. 3.3 (compare the red and grey curves in Fig. 3.5a), even though the neuronal dynamics are very different (compare the far left and far right panels in Fig. 3.5b).

Finally, we also use an alternative learning rule to train the neuronal gains; in this rule, learning slows down as the decrease in error slows down. (See Eqns. (3.8) and (3.9) in Appendix 3.B.) We find that the error decreases at a faster rate than that in the original learning rule in Eqns. (3.5) and (3.6) (see the purple curve in Fig. 3.5a.) We speculate that this occurs because the standard deviation

of the noise perturbation term in the alternative learning rule becomes smaller over training iterations as the error decreases.

Notably, in all of these scenarios, changes in neuronal responsiveness alone — for example, via inputs from neuromodulatory afferents — can cause dramatic changes in network outputs, thereby providing an efficient mechanism for rapid switching between movements, without requiring any changes in either synaptic architecture or the initial condition.

3.7 Investigating the effects of more strongly nonlinear neuronal dynamics

We initially choose the baseline firing rate ($r_0 = 20$ Hz in Eqn. (3.1)) to be consistent with experimentally-measured firing rates in motor cortex (Churchland et al., 2012; Kao et al., 2015; Lara et al., 2018). Thus, most of the time, neurons operate within the linear part of their nonlinear gain function f; that is, the neuronal dynamics are similar to the case of using the linear gain function $f(x_i; g_i) = g_i x_i$ (see Figs. 3.6a,c). To test if our results hold for scenarios with more strongly nonlinear dynamics, we reduce the baseline firing rate to $r_0 = 5$ Hz. This increases the neuronal activity near the lower-saturation regime (i.e., towards the left part of the curves in Fig. 1.3) of the gain function (see Figs. 3.6b,c).

With $r_0 = 5$ Hz, we also train the neuronal gains on the same task that we showed in Fig. 3.3. As expected from the larger range of possible network outputs (and improved learning performance) in nonlinear recurrent neuronal-network models than in linear ones (Hoerzer et al., 2014; Miconi, 2017; Sussillo and Abbott, 2009), we observe better learning performance for $r_0 = 5$ Hz than for $r_0 = 20$ Hz (compare the black and blue curves in Fig. 3.6d), and we obtain a very similar distribution of gain values after training (see Fig. 3.6e).

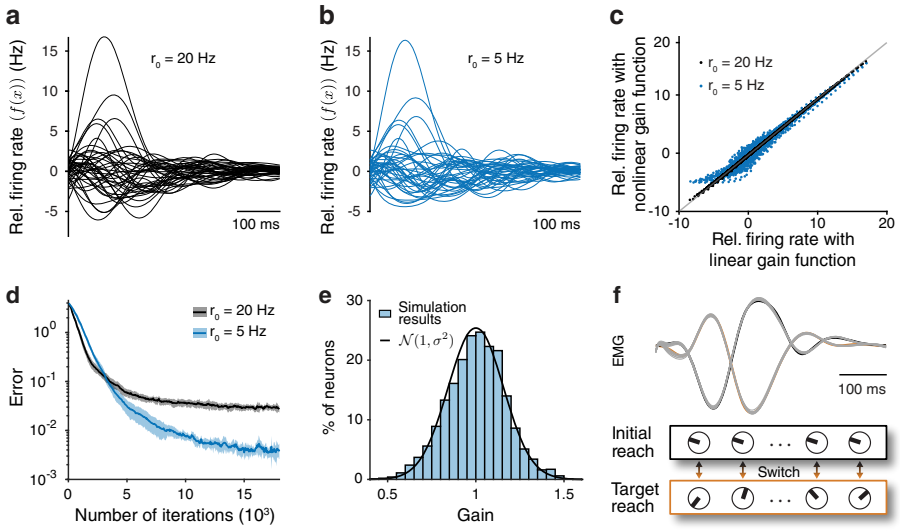

Figure 3.6: Examining effects of more strongly nonlinear neuronal dynamics by using a baseline rate of $r_0 = 5$ Hz. **a**, *Relative firing rate of 20 excitatory and 20 inhibitory neurons in a 200-neuron network with $r_0 = 20$ Hz in Eqn. (3.2).* **b**, *Relative firing rate of the same neurons as those in panel (a), but with $r_0 = 5$ Hz.* **c**, *We plot the relative firing rates of all neurons over time (although, in our visualisation, we only show every tenth data point) when using the nonlinear gain function f (see Eqn. (3.2)) with (black) $r_0 = 20$ Hz and (blue) $r_0 = 5$ Hz versus the relative firing rates that result from using the linear gain function $f(x_i; g_i) = g_i x_i$ in Eqn. (3.1). We set each neuronal gain to 1, and we plot the identity line in grey.* **d**, *Mean error during training with $r_0 = 20$ Hz (black) and with $r_0 = 5$ Hz (blue) for the same task that we showed in Fig. 3.3. Shading indicates one standard deviation.* **e**, *Histogram of gain values after training with $r_0 = 5$ Hz. The black curve is a Gaussian distribution with a mean of 1 and a standard deviation of $\sigma \approx 0.157$ (i.e., the distribution that we obtained with $r_0 = 20$ Hz in Fig. 3.3c).* **f**, *Network outputs (grey curves) with all gains set to 1 and the new learned gain pattern with $r_0 = 5$ Hz for 10 noisy initial conditions with noise added to the initial condition $\boldsymbol{x}_0$ with a signal-to-noise ratio of 30 dB. We show both targets in black and orange.*

3.8 Conclusions and discussion

In this chapter, we studied the effects of modulating the gain of individual neurons in recurrent neuronal networks. We found that increasing the gain of only one neuron substantially affects the frequency and amplitude of that neuron's firing-rate activity, whereas the other neurons' firing rates are changed by a comparatively small amount. These changes in the firing rate of the gain-modulated neuron are consistent with the changes that we observed in Section 2.5, where we modulated the gain of all neurons in a recurrent neuronal network.

We also found that modulating the gain of only one neuron in recurrent neuronal networks can yield substantial reductions in network output errors. Depending on the target output, particular neurons and gain values are most effective at reducing network output errors. In future work, it would be interesting to understand why such neurons seem to be most effective at reducing network output errors. The answer will likely involve studying the relationships between the network connectivity, the readout weights, and the firing rates of the neurons. For example, neurons that substantially reduce network output errors may have strong connections to the readout unit and the time course of the neuron's firing rate may correlate with network output errors over the duration of the movement.

After studying the effects of manually tuning the gain of neurons (see Section 3.3), we introduced a reward-based learning rule for iteratively changing the neuronal gains in recurrent neuronal networks so that network output errors, on average, are reduced. We were able to generate desired network outputs by training only neuronal gains, and our approach achieves a similar performance to training either the initial condition of the neuronal activity or training the recurrent weight matrix. It would be interesting to understand if these different training approaches achieve a different learning performance when training on other, perhaps more complex tasks (e.g., if one uses more readout units).

We showed that learning through gain modulation is effective in a variety of different models, for which the neuronal dynamics can differ substantially. We also showed that one can improve learning performance in a model in which the neuronal firing rates are more strongly nonlinear. To create more strongly nonlinear neuronal dynamics, we reduced the baseline firing rate to $r_0 = 5$ Hz. Alternatively, one can scale the initial condition $\boldsymbol{x}_0$ by a factor greater than 1 to create more strongly nonlinear firing-rate activities (Hennequin et al., 2014). (In our simulations, following Hennequin et al. (2014), we scaled $\boldsymbol{x}_0$ so that $\|\boldsymbol{x}_0\|_2 = 1.5\sqrt{N}$, where N is the number of neurons (see Appendix C.2); and we set the baseline firing rate to be consistent with experimental recordings (Churchland et al., 2012).) If one has access to relevant data sets, a more comprehensive approach may be to fit both the baseline firing rate r_0 and the factor by which one scales $\boldsymbol{x}_0$ so that the resulting neuronal firing rates most closely resemble experimental recordings.

We adopted a particular, commonly used functional form for the gain function (see Eqn. (3.2)). However, it will be important to also test whether our key results from this chapter hold if we use a strictly positive gain function (see Section 1.2.1.4) where gain changes also affect the baseline firing rate of neurons, for example, one could use the following gain function:

$$f(x_i) = \begin{cases} g_i r_0 \tanh(x_i/r_0) + r_0 \,, & \text{if } x_i < 0 \,, \\ g_i (r_{\max} - r_0) \tanh(x_i/(r_{\max} - r_0)) + r_0 \,, & \text{if } x_i \geq 0 \,. \end{cases} \tag{3.7}$$

From preliminary simulations, we find that learning through gain changes is still effective when using such a gain function, however, due to the altered effective baseline firing rates after gain modulation, network outputs do not automatically return to baseline at the end of the movement. Therefore, one must choose the readout weights in such a way that the altered baseline firing rates do not affect baseline network output activity.

In line with previous research (Churchland et al., 2012; Russo et al., 2018; Sussillo et al., 2015), we trained networks to generate specific target outputs (which

we suggest act as a proxy for muscle activity during movement execution). This is a simplification of actual motor learning, as there are many different possible muscle activations that can lead to a 'successful' movement. For some motor tasks, it is probably more biologically plausible to train a network to increase the success of the desired movement defined by the position of an end effector while also minimising the total amount of muscle activity (e.g., see Kambara et al. (2013); Miconi (2017)). Nevertheless, our learning rule is biologically realistic, in that it uses only local information and a single scalar signal (which is the total sum of squared errors) per trial. It thus does not carry detailed information about the exact way in which an output trajectory deviates from a desired trajectory. We thus expect that our main results will still be relevant for more biologically realistic models of motor learning (e.g., using a biophysically realistic model of a human arm (Miconi, 2017)).

3.A Supplementary simulation details for Fig. 3.4

For Fig. 3.4c, we generate 100 network outputs for each of the 10 trained gain patterns that we obtained from the task that we showed in Fig. 3.3 using 100 different instances of white Gaussian noise added to the initial condition $\boldsymbol{x}_0$ with a signal-to-noise ratio of s dB (where we consider values of s between 1 and 30 dB in increments of 1). We then calculate the square of the Euclidean 2-norm between each network output and the network output that we obtain when we do not add noise to the initial condition. We call these squared errors $\boldsymbol{e}_1$. (This vector has $1,000$ entries, with one entry for each network output.) We also generate $1,000$ outputs with all gains set to 1 using $1,000$ different instances of white Gaussian noise added to the network initial condition $\boldsymbol{x}_0$ with a signal-to-noise ratio of s dB. (We again consider values of s between 1 and 30 dB in increments of 1.) We then calculate the square of the Euclidean 2-norm between each of these network outputs and the network output that we obtain with all gains set to 1 and no noise added to the initial condition. We call these squared errors $\boldsymbol{e}_2$. For each signal-to-noise

ratio s, we plot in Fig. 3.4c the mean and standard deviation of $\boldsymbol{e}_1$ (i.e., the squared error corresponding to the trained gain patterns) in red and $\boldsymbol{e}_2$ (i.e., the squared error corresponding to all gains set to 1) in blue. We obtain very similar errors for both the trained and untrained (i.e., all gains set to 1) gain patterns, except for large (i.e., approximately larger than 25 dB) signal-to-noise ratios.

For Fig. 3.4d, we generate 10 different target muscle activities (see Section 3.2.2) and, independently for each movement, we train either the neuronal gains, the recurrent synaptic weight matrix, the initial condition, or a rank-1 perturbation of the recurrent synaptic weight matrix using a gradient-descent training procedure (with gradients that we obtain from back-propagation (Rumelhart et al., 1986)). We use the same 200-neuron network and readout weights that we used in Fig. 3.3. In other words, before any training, the network output is the black curve that we showed in Fig. 3.3d. The cost function for the training procedure is the squared Euclidean 2-norm between the actual network output and the target output scaled by the total sum of squares of the target output (i.e., Eqn. (D.1) in Appendix D). We train in a similar way to when we use our reward-based learning rule, that is, we provide the initial condition $\boldsymbol{x}_0$ at time $t = 0$, we then run the neuronal dynamics until $t = 0.5$ s and we then update parameters by summing (over all times) partial derivatives of the cost function with respect to the neuronal gains. We run the gradient-descent training procedure until the difference between the cost function at successive training iterations is below 10^{-5} (i.e., until the cost saturates to a small value). For the rank-1 perturbation, we independently train vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^{200\times 1}$ to reduce the error between the network output, which we obtain from the neuronal dynamics in Eqn. (3.1) with $\boldsymbol{W}$ replaced by $\boldsymbol{W} + \boldsymbol{u}\boldsymbol{v}^\mathsf{T}$, and the target output. We plot the errors for 10 different target outputs for each of our 4 different training approaches in Fig. 3.4d.

3.B Alternative learning rule

One can also adapt our learning rule so that learning ceases when the modulatory signal $R(t_n)$ saturates at a sufficiently small value. A way to achieve this is by instead placing the noise term ξ_i inside the brackets in Eqn. (3.5), so that the modulatory signal R multiplies ξ_i, together with changing the sgn function in Eqn. (3.6) to the $\tanh$ function. This yields the following learning rule:

$$g_i(t_n) = g_i(t_{n-1}) + R(t_{n-1})(g_i(t_{n-1}) - \bar{g}_i(t_{n-1}) + \xi_i(t_n))\,, \tag{3.8}$$

where

$$\begin{aligned} R(t_n) &= \tanh(\eta(\bar{\epsilon}(t_{n-1}) - \epsilon(t_n)))\,, \\ \bar{\epsilon}(t_n) &= \alpha\bar{\epsilon}(t_{n-1}) + (1-\alpha)\epsilon(t_n)\,, \\ \bar{g}_i(t_n) &= \alpha\bar{g}_i(t_{n-1}) + (1-\alpha)g_i(t_n)\,, \end{aligned} \tag{3.9}$$

and $\eta = 50,000$ controls the slope of the $\tanh$ function at 0 (i.e., when the low-pass-filtered error $\bar{\epsilon}(t_n)$ matches the current error $\epsilon(t_n)$). Learning now stops when $\bar{\epsilon}(t_{n-1}) = \epsilon(t_n)$; see the purple curve in Fig. 3.5a. We achieve a qualitatively similar learning performance by using Eqns. (3.8) and (3.9) instead of Eqns. (3.5) and (3.6), respectively. Compare the purple and red curves in Fig. 3.5a.

CHAPTER 4

Coarse, group-based learning of neuronal gains

Thus far in this thesis, we have studied both the effects of identically modulating the gain of all neurons identically (see Chapter 2) and independently modulating the gain of each neuron (see Chapter 3) in recurrent neuronal networks. Mechanisms for realistically changing neuronal gains (such as neuromodulators (Hernandez-Lopez et al., 2000; Thurley et al., 2008; Wei et al., 2014)), will likely affect multiple neurons in a similar manner, as opposed to each neuron independently. In line with the existence of diffuse (i.e., not neuron-specific) neuromodulatory projections to M1 (Hosp et al., 2011; Huntley et al., 1992; Molina-Luna et al., 2009), in this short chapter, we investigate the effects of modulating the gain of groups of neurons in recurrent neuronal networks. That is, we identically modulate neurons within pre-defined groups. We find that such coarse-grained modulation gives similar performance to neuron-specific control for as few as 20 groups in a network of 200 neurons when groups are chosen uniformly at random. For a given number of groups, one can improve performance if, instead of grouping neurons randomly, we use a specialised grouping for particular target movements based on previous training sessions.

Notably, even with random groupings, we show that network size (i.e., the number of neurons) hardly affects learning performance when we use a single readout unit. We find that performance depends much more on the number of groups than on the number of neurons per group. However, when the task involves two or more readout units, larger networks do learn better, and achieving a good performance necessitates using a larger number of independently modulated groups. Finally, smaller networks typically learn faster, but they ultimately exhibit poorer performance, demonstrating that there is a trade-off between network size, number of groups, and task complexity (i.e., the number of readout units). The results that we present in this chapter form part of an article that has been accepted at Nature Neuroscience: *Jake P. Stroud, Mason A. Porter, Guillaume Hennequin, and Tim P. Vogels, 'Motor primitives in space and time via targeted gain modulation in cortical networks'*.

4.1 Methods

For the simulations in this chapter, we use the same model setup that we used in Chapter 3. Specifically, Eqn. (3.1) governs the neuronal dynamics, we use the gain function in Eqn. (3.2) (with $r_0 = 20$ Hz), and we generate the synaptic weight matrix $\boldsymbol{W}$ in line with Hennequin et al. (2014). That is, we use stability-optimised circuits (see Section 1.2.4). We use the learning rule we presented in Section 3.4 to train neuronal gains so that networks generate desired target outputs. For further modelling details, see Section 3.2.

4.1.1 Generating groups for group-based gain modulation

We create groups of neurons in recurrent neuronal networks so that neurons in the same group always have the same gain and we independently modulate the gain of each group (see Fig. 4.1a). We can generate n (modulatory) groups in a network of N neurons with $1 \leq n \leq N$. Thus, if we use n groups, we have n free

parameters (which are the gain values for each group), that we can modify. Our generation mechanism for 'random groups' is as follows. For each of the n groups, we choose N/n neurons uniformly at random without replacement. If n does not divide N, we assign the remaining neurons to groups uniformly at random.

We also create 'specialised groups' for a particular target movement. We obtain specialised groups by applying k-means clustering (where k is the desired number of groups) to 10 gain patterns that we obtain from 10 prior independent training sessions (using neuron-specific modulation; see Section 3.5) on the same target and which correspond to the minimum error for each training session. We thus apply k-means clustering to a matrix of size $N \times 10$, where row i contains the gain values for neuron i from the 10 independent training sessions to the same target. Applying k-means clustering generates groups so that neurons in the same group tend to have similar gain values following training using neuron-specific modulation.

4.2 Learning one target movement

In Fig. 4.1b, we show the performance of group-based modulation for the same task that we showed in Fig. 3.3 (where we used neuron-specific modulation; see the red curve in Fig. 4.1b). We find that with only a few groups (e.g., 2), performance is poor regardless of whether one uses random or specialised groups. Using more independently modulated groups improves performance so that 20 random groups yields a similar performance as neuron-specific modulation (see the overlap of the black and red shading in the left panel of Fig. 4.1b). We also find that 10 specialised groups substantially outperforms both random groups and neuron-specific modulation.

Notice that for a given number of training iterations and when using specialised groupings, 10 groups tends to outperform 20 groups. This unintuitive behaviour is likely due to a combination of both our learning rule and the groups that we use. For example, our learning rule provides a single binary modulatory signal (R

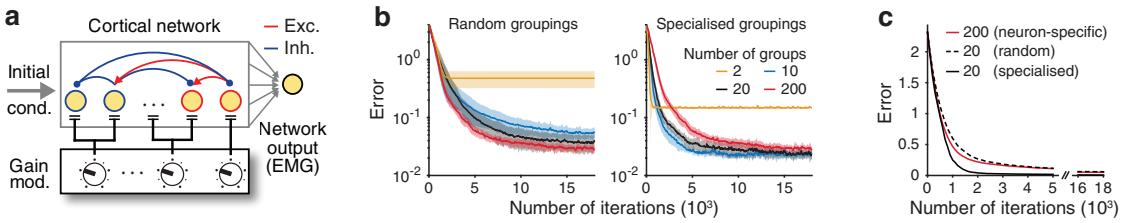

Figure 4.1: Learning one target movement with group-based gain modulation. **a**, *Schematic of our model for group-based gain modulation. We identically modulate neurons within a group (see Section 4.1.1). We use a* 200*-neuron network for these simulations.* **b**, *Mean error over* 10 *training sessions (where shading indicates one standard deviation) using (left) random and (right) specialised groupings for* 2*,* 10*,* 20*, and* 200 *(i.e., neuron-specific) groups. The target output is the same as in Fig. 3.3.* **c**, *Mean error during training for* 20 *random,* 20 *specialised, and* 200 *(i.e., neuron-specific) groups when training idependently on* 5 *different target movements.*

in Eqn. (3.5)) indicating whether performance is better or worse, on average, compared with previous trials. Therefore, when increasing the number of independently modulated groups (e.g., 20 versus 10 groups), a change in a single gain value has a smaller contribution (on average) to a change in performance. For example, if we have 1 group (i.e., we modulate every neuron identically), we know that a change in output error due to a change in this one gain value is purely attributable to this single gain value. Therefore, with fewer groups, one may reach a local minimum in fewer training iterations, however the error at the local minimum may be large (e.g., see the yellow curves in Fig. 4.1b). Therefore, there is a trade-off between the number of groups, the number of training iterations, and the desired final error after training. These effects are in line with the known reduction in the speed of learning when increasing the number of free parameters if using node-perturbation learning rules such as ours (Werfel et al., 2005).

We also note that the method by which we obtain specialised groupings uses gain patterns that correspond to the minimum error over all training iterations when using neuron-specific modulation. Neurons that have similar gain values at the

minimum error do not necessarily also have correlated gain changes over the entire course of training. On the contrary, the gain values may have changed in rather different ways during training. However, when we train with specialised groupings, the gains within a group are tied to the same value during training and we can obtain better performance than neuron-specific modulation (see the right panel of Fig. 4.1b). Therefore, if neurons yield similar gain values after training when we use neuron-specific modulation, our results imply that one can improve performance if these neurons are constrained to have the same gain value during training.

In Fig. 4.1c, we show mean results when training independently on 5 different target outputs (note that the vertical axis scale is logarithmic in Fig. 4.1b and linear in Fig. 4.1c). (We use different specialised groups for each of the 5 targets.) We find that 20 random groups perform similarly to neuron-specific modulation and 20 specialised groups substantially outperform both neuron-specific modulation and 20 random groups. Importantly, in all of the simulations that we show in Fig. 4.1, for a given number of groups, specialised groupings always outperform random groupings over the range of training iterations that we used. Therefore, because we generate specialised groups by applying k-means clustering to 10 gain patterns that correspond to the minimum error on 10 independent training sessions using neuron-specific modulation, the improvement in performance from using such groupings implies that there are some important shared characteristics among such a set of 10 gain patterns.

4.3 Learning multiple movements using a fixed grouping

It is likely unrealistic that the brain can reorganise its modulatory synaptic inputs onto motor cortex in order to execute different movements. Therefore, a given modulatory grouping should perform well for multiple movements. In this section, we examine the possibility of using a fixed grouping when learning multiple different movements.

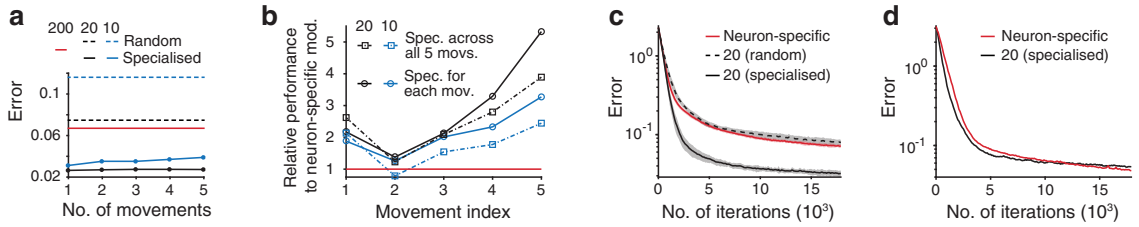

Figure 4.2: Learning multiple target movements using a fixed grouping. **a**, *Mean minimum errors after training when we use the same grouping for learning different numbers of movements. (See Appendix 4.A.1 for further details.)* **b**, *Relative improvement in performance compared with neuron-specific modulation for each of* 5 *movements when using specialised groups shared across all (squares) or for each (circles) of the* 5 *movements using either* 10 *(blue) or* 20 *(black) groups. A value of* 2 *implies that the error is* 2 *times smaller after training compared to neuron-specific modulation. We indicate the performance of neuron-specific modulation using the red line.* **c**, *Mean error over* 10 *training sessions (where shading indicates one standard deviation) when learning* 5 *movements using either* 20 *random groups, neuron-specific modulation, or* 20 *specialised groups shared across all* 5 *movements.* **d**, *Mean error over* 10 *training sessions when learning* 10 *novel movements using the specialised grouping (with* 20 *groups) shared across the* 5 *previously trained movements from panel (c).*

We first note that using random groups or neuron-specific modulation do not (on average) yield different performances when learning multiple different movements. However, when we create specialised groupings for learning multiple movements (see Appendix 4.A.1 for further details), performance can degrade as you increase the number of movements (see the blue points in Fig. 4.2a). However, we find that a specialised grouping consisting of 20 groups can achieve a consistently good performance when learning any number of up to 5 different movements (see the black points in Fig. 4.2a).

For the 5 target movements that we used in Fig. 4.2a, we plot the improvement in performance (see Fig. 4.2b) compared with neuron-specific modulation (see the red line) for each target movement when using specialised groupings either shared

across all (squares) 5 movements or for each (circles) of the 5 movements. We find that target movement index 5 (see the far right data points in Fig. 4.2b), yields the best improvement in performance compared with the other 4 movements for each grouping scenario. Additionally, using the specialised grouping for only target movement index 5, yields an error over five times smaller than neuron-specific modulation when we use 20 groups (see the black circle for movement index 5 in Fig. 4.2b). For movement index 2, performance is hardly improvement compared with neuron-specific modulation, even when we use 20 specialised groups for only movement index 2. In general, we find that using a specialised grouping of 10 groups shared across all 5 movements (see the blue squares) produces the smallest increase in performance compared with the other scenarios, no matter which movement you learn. In contrast, using a specialised grouping of 20 groups that is specific to each movement (see the black circles) tends to produce the largest increase in performance, no matter which movement you learn.

In Fig. 4.2c, we plot the mean error during training when using a specialised grouping of 20 groups shared across all 5 movements (e.g., the minimum of the solid black curve in Fig. 4.2c corresponds to the black data point on the far right in Fig. 4.2a). We also use this specialised grouping to learn 10 hitherto-untrained movements (see Fig. 4.2d), and we achieve a similar performance to neuron-specific modulation.

These results suggest that one does not need to independently modulate each neuron's gain to accurately learn target movements. Using 20 random groups in a network of 200 neurons (i.e., 10 % of the number of free parameters compared with using neuron-specific modulation), yields a similar learning performance to neuron-specific modulation. Moreover, we can find (albeit crudely; see Section 4.1.1), specialised groups that substantially outperform neuron-specific modulation when learning multiple different movements. It would be interesting to understand whether one can find even more effective 'specialised groupings' for learn-

ing target movements and whether the brain employs mechanisms for constructing such modulatory groups.

4.4 Effects of network size when learning with random groups

We now examine how network size (i.e., the number of neurons), affects learning performance when using group-based modulation. We only use random groups for all subsequent simulations in this chapter. For our simulations in this section, we generate 5 different target outputs and run 10 independent training sessions for each target. We consider various different numbers of random groups for networks with $N = 100$, $N = 200$, and $N = 400$ neurons. We fit the readout weights so that each scenario generates the same network output when all gains are set to 1.

We observe that increasing the number of independently modulated groups yields smaller errors (on average over multiple independent training sessions and over different target movements) after training (see Fig. 4.3a). This is also typically accompanied by a smaller variance in errors. In line with our observations in Fig. 4.1, we find that with an increasing number of modulatory groups, more training iterations are required to achieve the typically smaller error associated with a larger number of modulatory groups.

In Fig. 4.3b, we compare the learning performance for the 5 different targets and the 3 networks. The relative sizes of the errors for the 5 movements after training appear to be unrelated to the errors before training for any number of groups. That is, different movements appear to exhibit different learning performance regardless of the error before training. For example, the movement coloured in grey in Fig. 4.3b has the smallest error both before and after training when using up to 3 modulatory groups. However, increasing the number of modulatory groups above 3 yields little improvement in performance for this movement compared with the other movements. In Fig. 4.3c, we show that all 3 networks yield a similar learning performance for any number of random groups up to and including neuron-specific

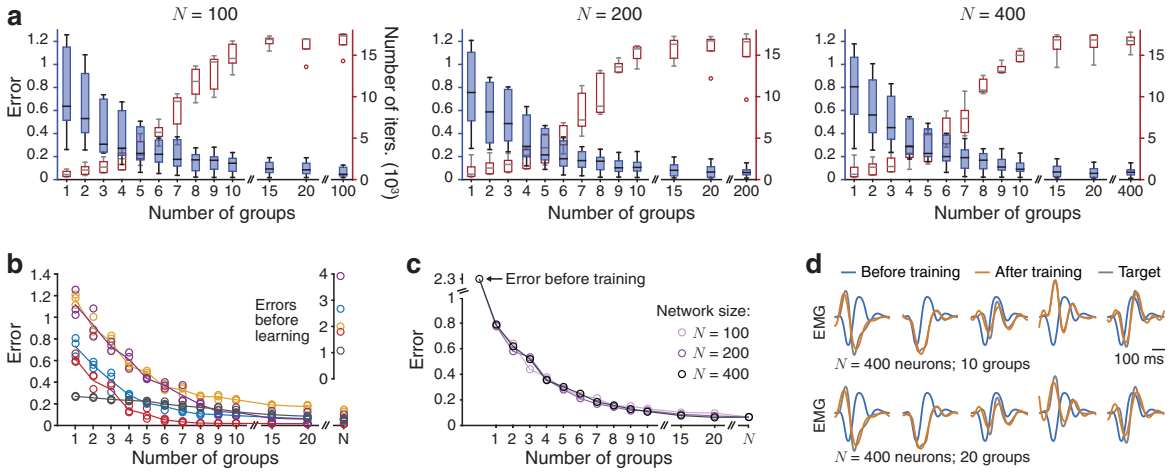

Figure 4.3: Effects of network size when learning with group-based gain modulation. **a**, *Box plots (in blue) of the minimum error after training on* 5 *different target movements for different numbers of random groups with networks of* 100, 200, *and* 400 *neurons. We also include box plots (in red) for the minimum number of training iterations required before the error is within* 1 *% of the minimum error. We use Tukey style for the whiskers in the box plots.* **b**, *The curves give the mean error over* 10 *training sessions and across the* 3 *networks for each of the* 5 *target movements. The circles represent the mean error for each network, and the different colours indicate each of the* 5 *different targets (e.g., for a given number of groups, there are three circles in each of five different colours). (The* N *on the horizontal axis indicates neuron-specific modulation.)* **c**, *Mean minimum errors across all* 5 *targets for each network versus the number of modulatory groups. (We use the same data to generate the plots in panels (a)–(c))* **d**, *Outputs for all* 5 *targets from the trial that produces the median error for the* 400*-neuron network for the cases of* 10 *and* 20 *groups.*

modulation. Performance depends much more on the number of groups than on the number of neurons per group. We also find that there is only a small improvement in performance for all 3 networks when using more than approximately 10 modulatory groups. That is, regardless of network size, the error saturates to a similar value when we use a large number of modulatory groups.

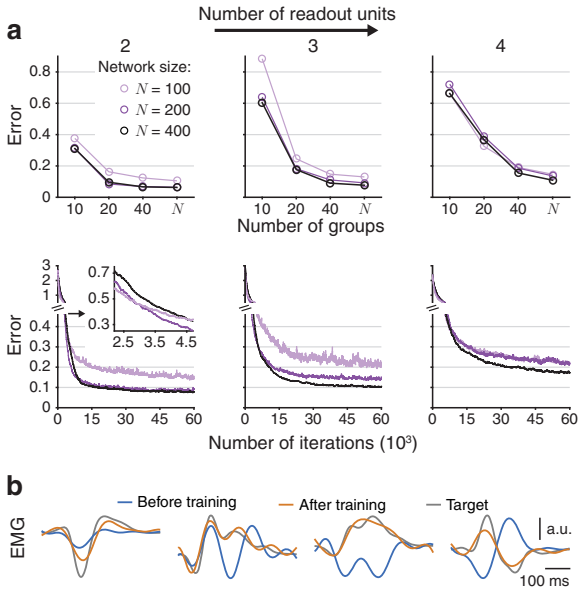

Figure 4.4: Effects of increasing the number of readout units. **a**, *(Top) Mean minimum error as a function of the number of random groups when learning each of (left)* 2*, (centre)* 3*, and (right)* 4 *readouts for the same networks that we used in Fig. 4.3 (i.e., networks of* 100*,* 200*, and* 400 *neurons). (The* N *on the horizontal axis indicates neuron-specific modulation.) (Bottom) The corresponding mean errors during training for the case of* 40 *groups. The inset is a magnification of the initial training period for the case of* 2 *readout units.* **b**, *Outputs producing the median error for the case of* 4 *readout units using* 40 *groups in the* 400*-neuron network. (See Appendix 4.A.2 for further details.)*

4.5 Increasing task complexity

So far in this chapter, we have used only 1 readout unit. It could be the case that the results that we have obtained so far do not hold when we increase the number of readout units (i.e., increasing the task complexity). In this section, we examine the effects of increasing the number of readout units.

We find that when the task involves two or more readout units, larger networks do learn better, and achieving a good performance necessitates using a larger number of independently modulated groups (see Fig. 4.4 and Appendix 4.A.2). For example, when we use 4 readout units, good performance requires approxi-

mately 40 or more modulatory groups (see the top right panel of Fig. 4.4a and also Fig. 4.4b). As we increase the number of readout units, we observe an increase in the relative improvement in performance from using more independently modulated groups (see the top row in Fig. 4.4a). We also find that smaller networks typically learn faster (see the inset in the bottom left panel of Fig. 4.4a), but they ultimately exhibit poorer performance (see the light purple curves in Fig. 4.4a). Therefore, there is a trade-off between network size, number of groups, and task complexity (i.e., the number of readout units).

4.6 Conclusions and discussion

If neuronal gains do change during learning, realistic mechanisms that can cause such changes will likely only be able to affect neuronal gains in a coarse (i.e., non neuron-specific) manner. For example, synaptic inputs from other brain regions could affect neuronal gains (Chance et al., 2002), and the experimentally observed diffuse neuromodulatory projections onto primary motor cortex (Hosp et al., 2011; Huntley et al., 1992; Molina-Luna et al., 2009) can change neuronal gains (Thurley et al., 2008). Thus, it may be the case that proximal neurons will be affected in similar ways. Therefore, it is important to understand the effectiveness of modulating groups of neurons rather than each neuron independently.

We found that 20 modulatory groups chosen uniformly at random (see Section 4.1.1) yields a similar performance to neuron-specific control in a 200-neuron network when we use 1 readout unit. We also found that performance can be improved substantially if, instead of grouping neurons uniformly at random, we choose groups based on previous training sessions by using 'specialised groupings' (see Section 4.1.1). Importantly, there exist specialised groupings that perform similarly across multiple different movements. Such specialised groupings acquired from learning one set of movements can also perform well on novel movements.

Our results from Figs. 4.1 and 4.2 were based on using a 200-neuron network. However, networks in the brain contain orders of magnitude more neurons. Therefore, it is important to understand how our results are affected when we change the network size. Interestingly, we found that network size hardly affects learning performance for a single readout unit and good performance is improved only slightly if one uses more than approximately 10 random groups. However, when the task involves more than one readout unit, more independently modulated groups are required to achieve good performance and larger networks do yield better performance.

To create specialised groupings, we applied k-means clustering to gain patterns that correspond to the minimum error when training using neuron-specific modulation (see Section 4.1.1). Although this is rather crude, we found that specialised groups substantially improve performance compared with neuron-specific modulation. It may be interesting to obtain groups using alternative approaches. For example, one could obtain groups by applying k-means clustering to gain patterns obtained over the entire course of training using neuron-specific modulation. Therefore, neurons placed in the same group may tend to have similar gain values over the entire course of training. We also tested the effects of creating groups by applying k-means clustering to either the neuronal firing rates or to the readout weights, but we found no improvement in performance compared with using random groups (not shown).

Another interesting area of future research may be to investigate the difference between the time series of gain values obtained during the course of training when using specialised groups compared with neuron-specific modulation. For example, the gain pattern corresponding to the minimum error during training when using neuron-specific modulation is a local minimum in the neuronal gain 'error landscape' (or 'loss landscape') (Li et al., 2018; Rakitianskaia et al., 2016). We showed that when we constrain training by using specialised groups corresponding to this

gain pattern, performance can be improved. But an important questions is: Do we still find a similar local minimum when using the specialised groupings? We know the same local minimum likely still exists, but it is unclear whether one can reach it when we use the specialised grouping. Therefore, it would be interesting to investigate how the error landscape changes when using grouped modulation.

In the remaining chapters, we use group-based gain modulation for all of our simulations and we use a 400-neuron network with 40 random modulatory groups.

4.A Supplementary simulation details

4.A.1 Details for Fig. 4.2

We generate 5 different target outputs and run 10 independent training sessions for each target. For the random groupings (see Section 4.1.1), we use different independently-generated random groups for each simulation. For the specialised groups (see Section 4.1.1), for a given number of groups, we use the same grouping in all simulations. We plot the results of using 10 or 20 groups with either random or specialised groups in Fig. 4.2. We use $18,000$ training iterations for all simulations.

We now explain how we determine specialised groups that are shared by multiple movements (i.e., we use the same grouping for learning multiple movements). We apply k-means clustering (where k is the desired number of groups) across all of the gain patterns that we obtain using neuron-specific modulation for each of the movements. That is, we apply k-means clustering to a matrix of size $N \times (10 \cdot q)$, where N is the number of neurons and q is the number of movements (and, equivalently, the number of gain patterns).

4.A.2 Details for Fig. 4.4

When we use multiple readout units, we generate 10 different 'initial' and target outputs for each readout unit. For example, for 2 readout units, we generate 10 different initial and target outputs for each of units 1 and 2. We fit readout weights so that with all gains set to 1, the network generates the initial output for each readout unit. We run independent training sessions for these 10 sets of target outputs and calculate mean errors across the 10 training sessions. For a given number of readout units, we use the same sets of initial and target outputs for all 3 network sizes and each number of random modulatory groups. We thus fit readout weights so that each scenario generates the same output with all gains set to 1. (The readout weights remain fixed throughout training.) We use $60,000$ (instead of $18,000$) training iterations to ensure error saturation.

CHAPTER 5

Learned gain patterns can provide motor primitives for novel movements

In Chapters 3 and 4, we trained networks to generate target outputs by iteratively changing neuronal gains using our learning rule in Eqn. (3.5). However, in addition to slowly learning movements relatively slowly over time, we know that the brain can also rapidly generate new movements to a reasonable degree of accuracy (at least if the movement corresponds to neural activity that is relatively similar to neural activity associated with previously learned movements (Golub et al., 2018; Sadtler et al., 2014)). As an example, consider an inexperienced darts player who has spent several hours practising throwing at two different numbers on the dart board, say the numbers 10 and 20. If the player now decides to throw for the number 8 (which they have not attempted yet), we know that the brain can use information gained from throwing at the numbers 10 and 20 to update so-called 'internal models' (Wolpert and Flanagan, 2001; Wolpert et al., 1998) in the motor system to try and throw to the number 8.

It has been suggested that one mechanism by which the brain generates move-

ments is by combining previously acquired movement building blocks (or 'motor primitives') (Giszter, 2015; Thoroughman and Shadmehr, 2000). Utilising such an approach for movement control can allow motor systems to rapidly generate movements, and there is considerable evidence that brains use motor primitives at various different levels of the motor system (for example, evidence for motor primitives has been found in muscles, spinal motoneurons, and cortical motor circuits (Bizzi and Cheung, 2013; Flash and Hochner, 2005; Giszter, 2015)).

In principle, in our model, it is possible to independently learn numerous gain patterns that correspond to many different movements, supporting the possibility of a repertoire (which we call a 'library') of modulation states that a network can use, in combination, to produce a large variety of outputs. In this short chapter, we examine whether a learned library of gain patterns can act as motor primitives so that one can 'intuit' new gain patterns for new desired movements as combinations of previously acquired gain patterns.

We find that we can accurately generate desired target movements by linearly combining gain patterns from a previously learned library. This may seem unintuitive, but we show that one can understand this result mathematically by calculating power-series expansions of the solution of the linearised neuronal dynamics. We also find that increasing the number of elements in the library reduces errors in network outputs. Importantly, when we use more strongly nonlinear neuronal dynamics by reducing the baseline firing rate to $r_0 = 5$ Hz, it is still possible to learn new movements by using combinations of existing gain patterns. The results that we present in this chapter form part of an article that has been accepted at Nature Neuroscience: *Jake P. Stroud, Mason A. Porter, Guillaume Hennequin, and Tim P. Vogels, 'Motor primitives in space and time via targeted gain modulation in cortical networks'*.

5.1 Methods

For the simulations in this chapter, we use the same setup that we used in Chapters 3 and 4. Specifically, Eqn. (3.1) governs the neuronal dynamics, we use the gain function in Eqn. (3.2) (with $r_0 = 20$ Hz unless we state otherwise), and we generate the synaptic weight matrix $\boldsymbol{W}$ in line with Hennequin et al. (2014). That is, we use stability-optimised circuits (see Section 1.2.4). We use a 400-neuron network with 40 random modulatory groups (see Section 4.1.1 for how we generate such groups) for all of our simulations in this chapter. For further modelling details, see Section 3.2.

5.1.1 Creating libraries of learned movements

To create libraries of learned movements in our model, we train a 400-neuron network with 40 random groups (see Section 4.1.1) on each of 100 different target movements independently using our learning rule Eqn. (3.5) (see Section 3.2.2 for a description of how we create target movements). In other words, this generates 100 different gain patterns, with one for each movement. For library sizes of $l \in \{1, 2, \ldots, 50\}$, we choose 100 samples of l movements (from the learned gain patterns and their associated outputs) uniformly at random without replacement for each l. We then fit the set of l movements in each of the 100 sample libraries using least-squares regression for each of 100 hitherto-untrained novel target movements (see Fig. 5.1 for an illustration). We constrain the fitting coefficients c_j from the least-squares regression by requiring that $c_j \geq 0$ for all j and $\sum_{j=1}^{l} c_j = 1$. That is, we consider convex combinations of the coefficients c_j. We calculate the fit error (i.e., the error between the fit and the target), the output error (i.e., the error between the output and the target), and the error between the fit and the output for each of the 100 novel target movements, each of the 100 library samples, and each l (see Appendix D for how we calculate errors). See Appendix 5.A

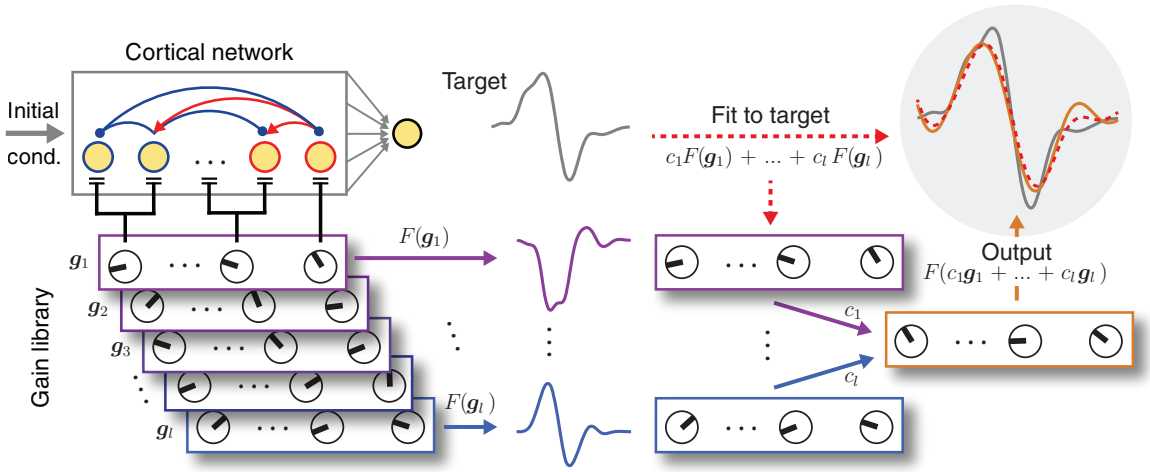

Figure 5.1: Schematic of a learned library of gain patterns ($\boldsymbol{g}_1, \ldots, \boldsymbol{g}_l$, which we colour from purple to blue) and a combination $c_1F(\boldsymbol{g}_1) + \ldots + c_lF(\boldsymbol{g}_l)$ of their outputs (which we denote by F) that we fit (red dashed curve) to a novel target (grey curve). (Upper right) The output $F(c_1\boldsymbol{g}_1 + \ldots + c_l\boldsymbol{g}_l)$ (which we show in orange) of the same combination of corresponding gain patterns also closely resembles the target. (See the main text for further details.)

for further simulation details.

5.2 Learned gain patterns can be combined to generate new desired movements

In Fig. 5.2a, we plot the distribution of gains over all 100 trained gain patterns. We find that the distribution resembles a normal distribution (blue curve) with a similar mean and variance as those that we found in Fig. 3.3c. This implies that when learning different movements, but where the movements are taken from the same generating process (see Section 3.2.2), the distribution of neuronal gains is similar for the different movements (also see Fig. 5.4 in Appendix 5.B). Interestingly, when we create gain patterns by sampling uniformly at random from the distribution that we show in Fig. 5.2a, we find that the network outputs that result from these gain patterns are substantially more homogeneous than the original trained movements and likely would not constitute a good library for movement generation (compare

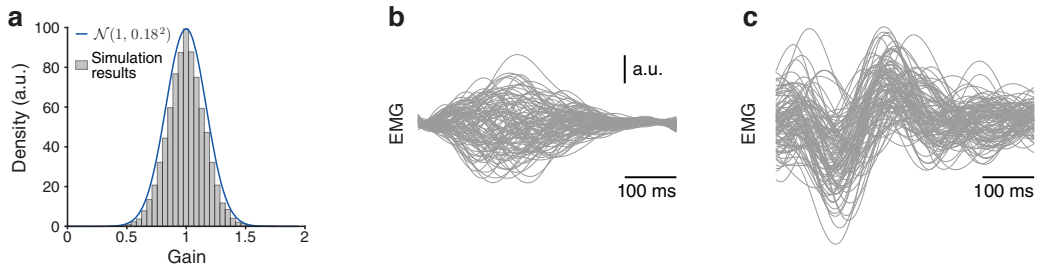

Figure 5.2: **a**, *The resulting distribution of gains from training independently on each of* 100 *target movements (see Section 5.1.1). As a comparison, in Fig. 3.3 we obtained a standard deviation of* $\sigma \approx 0.157$. **b**, *Each output from the* 100 *trained gain patterns.* **c**, *Outputs of* 100 *randomly-generated gain patterns that are creating by sampling uniformly at random from the distribution in panel (a).*

panels (b) and (c) in Fig. 5.2). Therefore, this implies that the exact association between the gain values and the respective groups of neurons they control is important for gain patterns to act as a library for movement generation, not merely the distribution of the gain values.

Following the method that we describe in Section 5.1 and Appendix 5.A, we find that we can accurately generate new movements from convex combinations of previously learned gain patterns (see Fig. 5.3). Performance is limited by the accuracy with which one can construct target movements as linear combinations of existing primitives. (See Fig. 5.3a and the correlations between output errors and fit errors in Fig. 5.3b.) Increasing the number of elements in the movement library reduces the error between a target movement and its fit, which is also reflected in a progressively better match between the target and the network output (see Figs. 5.3b,c). Importantly, the difference between network outputs and fits remains small for all tested numbers of library elements (see the blue curve in Fig. 5.3c).

In Fig. 5.3, we show results for library sizes up to $l = 20$. However, we find that there is only a small change in the errors between $l = 20$ and $l = 50$ numbers of library elements (see Fig. 5.5a in Appendix 5.B). Interestingly, for more than approximately 20 library elements, the error between the output and the target

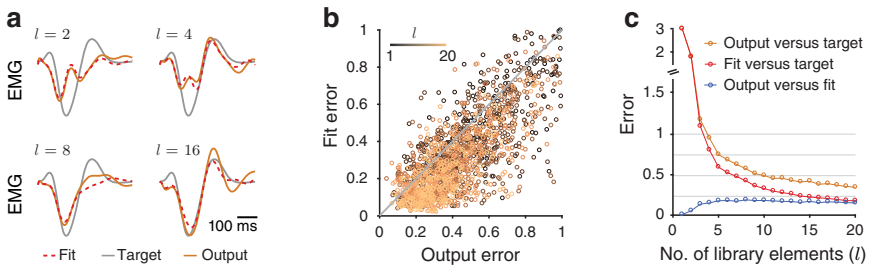

Figure 5.3: **a**, *Example target, fit, and output (grey, red dashed, and orange curves, respectively) producing the* 50*th-smallest output error over* 100 *randomly-generated combinations (see Section 5.1.1 for a description of the generation process) of* l *library elements using* $l = 2$, $l = 4$, $l = 8$, *and* $l = 16$. **b**, *Fit error versus the output error for* 100 *randomly-generated combinations of* l *library elements for* $l = 1, \ldots, 20$. *Each point represents the* 50*th-smallest error between the output and the fit across* 100 *novel target movements. We show the identity line in grey.* **c**, *Median errors of the* 100 *randomly-generated combinations of* l *library elements versus the number of library elements. We use a* 400*-neuron network with* 40 *random modulatory groups for these simulations (see Section 5.1). (See Appendix 5.A for further details.)*

(i.e., the orange curve in Fig. 5.5a in Appendix 5.B), appears to be composed in approximately equal proportion of fit errors and errors between the output and the fit.

We also find that the distribution of errors for each library size resemble Gaussian distributions (see Fig. 5.5b in Appendix 5.B). Therefore, our using the median error (or 50th-smallest error) in Fig. 5.3, appears to be a reasonable description of the data. We also note that although it is difficult to discern the exact errors for each number l of library elements in Fig. 5.3b, we find a strong positive correlation between fit and output errors for all tested numbers of library elements (see Fig. 5.5f in Appendix 5.B). Additionally, we obtain qualitatively similar results if we average over the 100 randomly-generated combinations of l library elements compared with if we instead average over the 100 novel target movements (compare Fig. 5.5e in Appendix 5.B with Fig. 5.3b; also see Appendix 5.A).

We also examined the possibility of using gain patterns as motor primitives with more strongly nonlinear dynamics by setting the baseline firing rate to $r_0 = 5$ Hz (see Section 3.7 for a discussion of the effects of using a baseline rate of $r_0 = 5$ Hz). Importantly, it is still possible to learn new movements by using combinations of existing gain patterns (see Fig. 5.6 in Appendix 5.B). As before, performance is limited by the accuracy with which one can construct target movements as linear combinations of existing primitives. (See the correlations between network output errors and fit errors in Fig. 5.6b in Appendix 5.B.) We obtain very similar results to those we obtained for the case of $r_0 = 20$ Hz. Errors in network output decrease on average with increasing numbers of gain patterns in the movement library (see the orange curve in Fig. 5.6c in Appendix 5.B), and the difference between the network output and corresponding fit remains small for all tested numbers of library elements (see the blue curve in Fig. 5.6c in Appendix 5.B). However, reducing r_0 to sufficiently small values (that are below 5 Hz) does eventually lead to a deterioration in the relationship between gain patterns and their corresponding outputs.

5.3 Analysis of linear combinations of gain patterns and their associated neuronal dynamics

So far in this chapter, we have illustrated that there is a consistent mapping between learned gain patterns and their outputs. Specifically, we illustrated that for a library of l gain patterns $(\boldsymbol{g}_1, \ldots, \boldsymbol{g}_l)$, a convex combination $c_1 F(\boldsymbol{g}_1) + \ldots + c_l F(\boldsymbol{g}_l)$ (so $c_j \geq 0$ for all j and $\sum_{j=1}^{l} c_j = 1$) of their corresponding outputs (which we denote by F) approximates the output $F(c_1 \boldsymbol{g}_1 + \ldots + c_l \boldsymbol{g}_l)$ that we obtain by combining the gain patterns with the same coefficients (see Figs. 5.1 and 5.3). Note that the subscript index j denotes the library element j and is not a neuron index. We now provide some mathematical intuition of this approximation by studying linearised solutions of the neuronal dynamics. Because the network output is a linear combination of

the neuronal firing rates, we study convex combinations of neuronal activity $\boldsymbol{x}$ directly; this then guarantees that the same result holds for convex combinations of network outputs.

For a convex combination (i.e., a weighted mean) of l vectors or matrices $\boldsymbol{\psi}$ with weights c_j, it is convenient to use the following notation:

$$\mathcal{C}\left[\tilde{\boldsymbol{\psi}}\right] = \sum_{j=1}^{l} c_j \boldsymbol{\psi}\,, \tag{5.1}$$

where the tilde in the square brackets is a reminder that we are summing over the index of the associated library terms. When we employ matrix multiplication in the following mathematical analysis, it is convenient to represent a gain pattern $\boldsymbol{g}_j \in \mathbb{R}^N$ using the matrix notation $\boldsymbol{G}_j = \mathrm{diag}(\boldsymbol{g}_j)$ (that is, the neuronal gains are elements along the diagonal of $\boldsymbol{G}_j \in \mathbb{R}^{N\times N}$, all other elements are 0, and the index j denotes library element j). Using this notation, the solution $\boldsymbol{x}_j(t) \in \mathbb{R}^N$ of the linearised dynamics of Eqn. (3.1) around $\boldsymbol{x} = \boldsymbol{0}$ is given by

$$\boldsymbol{x}_j(t) = e^{\frac{t}{\tau}(\boldsymbol{W}\boldsymbol{G}_j - \boldsymbol{I})}\boldsymbol{x}_0\,, \tag{5.2}$$

under the assumption that there are N distinct eigenvectors for the matrix $\boldsymbol{W}\boldsymbol{G}_j - \boldsymbol{I}$ and that we are away from any bifurcations. Let

$$\boldsymbol{u}(t) = e^{\frac{t}{\tau}(\boldsymbol{W}\mathcal{C}[\tilde{\boldsymbol{G}}] - \boldsymbol{I})}\boldsymbol{x}_0 \tag{5.3}$$

denote the neuronal activity that results from a convex combination $\mathcal{C}\left[\tilde{\boldsymbol{G}}\right]$ of gain patterns. We need to show that $\boldsymbol{u}(t)$ is approximately the same as the convex combination of the individual neuronal dynamics $\boldsymbol{x}_j(t)$ with the same coefficients c_j. That is, we need to show that the difference

$$\boldsymbol{\Delta}(t) = \boldsymbol{u}(t) - \mathcal{C}\left[\tilde{\boldsymbol{x}}(t)\right] \tag{5.4}$$

is small with respect to the magnitude of the neuronal activity. We first note that

$\frac{d\boldsymbol{\Delta}}{dt}\big|_{t=0} = \mathbf{0}$, which we prove as follows:

$$\begin{aligned}\frac{d}{dt}\boldsymbol{u}(t)\Big|_{t=0} &= \frac{1}{\tau}\left(\boldsymbol{W}\mathcal{C}\left[\tilde{\boldsymbol{G}}\right] - \boldsymbol{I}\right)\boldsymbol{x}_0 \\ &= \frac{1}{\tau}\mathcal{C}\left[\boldsymbol{W}\tilde{\boldsymbol{G}} - \boldsymbol{I}\right]\boldsymbol{x}_0 \\ &= \frac{d}{dt}\mathcal{C}\left[\tilde{\boldsymbol{x}}(t)\right]\Big|_{t=0},\end{aligned} \tag{5.5}$$

where we used the fact that $\sum_{j=1}^{l} c_j = 1$ to go from the first to the second line, and we note that the matrices $\boldsymbol{W}$ and $\boldsymbol{I}$ do not depend on the gain patterns.

To see whether we can also expect $\boldsymbol{\Delta}(t)$ to be small for $t > 0$, it is useful to consider the power-series expansion of the matrix exponentials on the right-hand side of Eqn. (5.4):

$$\mathcal{C}\left[\tilde{\boldsymbol{x}}(t)\right] = \mathcal{C}\left[\left(\sum_{m=0}^{\infty}\frac{(\boldsymbol{W}\tilde{\boldsymbol{G}} - \boldsymbol{I})^m}{m!}\right)^{\frac{t}{\tau}}\boldsymbol{x}_0\right], \tag{5.6}$$

$$\boldsymbol{u}(t) = \left(\sum_{m=0}^{\infty}\frac{\left(\boldsymbol{W}\mathcal{C}\left[\tilde{\boldsymbol{G}}\right] - \boldsymbol{I}\right)^m}{m!}\right)^{\frac{t}{\tau}}\boldsymbol{x}_0\,. \tag{5.7}$$

We observe in numerical simulations (not shown) that power-series expansions of this form are accurate descriptions of the associated neuronal dynamics up to second order in m. We therefore truncate to $m = 2$, and we evaluate the difference of Eqns. (5.6) and (5.7):

$$\boldsymbol{\Delta}(t) = \left(\frac{1}{2}\right)^{\frac{t}{\tau}}\left(\mathcal{C}\left[\left((\boldsymbol{W}\tilde{\boldsymbol{G}})^2 + \boldsymbol{I}\right)^{\frac{t}{\tau}}\right] - \left(\left(\boldsymbol{W}\mathcal{C}\left[\tilde{\boldsymbol{G}}\right]\right)^2 + \boldsymbol{I}\right)^{\frac{t}{\tau}}\right)\boldsymbol{x}_0\,. \tag{5.8}$$

We need to check if the right-hand side of Eqn. (5.8) is small compared to the neuronal dynamics (i.e., compared to Eqn. (5.6)). One way to check if this holds at certain times t is to substitute values of t into Eqns. (5.8) and (5.6) and calculate the ratio of the norms of these two expressions. Setting $t = \tau$ — at $t = \tau = 200$ ms, the neuronal dynamics are close having reached their maximum amplitude (see

Fig. 1.7b) — yields

$$\frac{\|\boldsymbol{\Delta}(t)\,|_{t=\tau}\,\|}{\|\mathcal{C}\left[\tilde{\boldsymbol{x}}(t)\,|_{t=\tau}\right]\|} \approx \frac{\left\|\left(\mathcal{C}\left[\left(\boldsymbol{W}\tilde{\boldsymbol{G}}\right)^2+\boldsymbol{I}\right]-\left(\boldsymbol{W}\mathcal{C}\left[\tilde{\boldsymbol{G}}\right]\right)^2-\boldsymbol{I}\right)\boldsymbol{x}_0\right\|}{\left\|\left(\mathcal{C}\left[\left(\boldsymbol{W}\tilde{\boldsymbol{G}}\right)^2+\boldsymbol{I}\right]\right)\boldsymbol{x}_0\right\|} = \frac{\left\|\left(\mathcal{C}\left[\left(\boldsymbol{W}\tilde{\boldsymbol{G}}\right)^2\right]-\left(\mathcal{C}\left[\boldsymbol{W}\tilde{\boldsymbol{G}}\right]\right)^2\right)\boldsymbol{x}_0\right\|}{\left\|\left(\mathcal{C}\left[\left(\boldsymbol{W}\tilde{\boldsymbol{G}}\right)^2\right]+\boldsymbol{I}\right)\boldsymbol{x}_0\right\|}. \tag{5.9}$$

We now study the magnitude of the numerator and the denominator of Eqn. (5.9) and show that the ratio of the former to the latter is small. Both the numerator and the denominator scale approximately in linear proportion to the norm of the product of $\boldsymbol{W}^2$ and $\boldsymbol{x}_0$. (The identity matrix in the denominator is small compared to $\boldsymbol{W}^2$.) The main difference between the numerator and denominator is their dependencies on the gain patterns $\boldsymbol{G}_j$. The numerator scales approximately proportionally to a 'weighted variance' of the gain patterns, whereas the denominator scales approximately proportionally to a weighted mean of the squared gain patterns. Because our learned gain patterns are typically narrowly distributed, with a mean of 1 and approximate standard deviation of 0.17 (see Fig. 5.4 in Appendix 5.B), this ratio is small (on the order of 10^{-3}) and actually decreases with an increasing library size. Numerically, we confirm that the normalised error in Eqn. (5.9) is indeed small, which also corroborates the results of this chapter.

Finally, although we restricted our discussion above to a linear gain function, we note that our numerical simulations suggest that Eqn. (5.4) is also small for the nonlinear gain function of Eqn. (3.2) (see Figs. 5.3, 5.5, and 5.6) that we used in all our simulations.

5.4 Conclusions and discussion

It is known that the brain can rapidly generate certain 'new' movements (Bizzi and Cheung, 2013; Giszter, 2015; Golub et al., 2018; Sadtler et al., 2014; Wolpert

et al., 1998). One way of achieving this is to combine previously learned movement building blocks (or motor primitives) (Bizzi and Cheung, 2013; Flash and Hochner, 2005; Giszter, 2015; Thoroughman and Shadmehr, 2000). In line with this perspective, in our model, we found that previously acquired gain patterns can be linearly combined in a predictable manner to generate new movements. Network output errors (on average) decrease as the number of elements in the movement library increases and errors between actual and expected network outputs remain low for all tested numbers of library elements. We provided some mathematical intuition for these results by calculating power-series expansions of the solution of the linearised neuronal dynamics. We found that if the variance of the gain patterns in the movement library is small compared with the mean of the squared gain patterns, then errors between actual and expected network outputs are expected to be small. Empirically, we found that the above-mentioned ratio of the variance to the mean of the gain patterns is satisfied in our simulations (for example, see Fig. 5.4 in Appendix 5.B). We also found that when we use more strongly nonlinear neuronal dynamics by reducing the baseline firing rate to $r_0 = 5$ Hz (see Fig. 3.6), we obtain very similar results to those obtained for the case of $r_0 = 20$ Hz.

Although the idea of using motor primitives to facilitate rapid acquisition of new movements is well established (Bizzi and Cheung, 2013; Flash and Hochner, 2005; Giszter, 2015; Thoroughman and Shadmehr, 2000), our approach proposes the first (to our knowledge) circuit-level mechanism for achieving this objective. In addition to neuromodulatory systems (Hernandez-Lopez et al., 2000; Molina-Luna et al., 2009; Thurley et al., 2008; Wei et al., 2014), the cerebellum is a natural candidate structure to coordinate such motor primitives (Thoroughman and Shadmehr, 2000; Wolpert et al., 1998), as it is known to project to M1 and to play a critical role in error-based motor learning (Spampinato et al., 2017; Thoroughman and Shadmehr, 2000).

To test whether a library of gain patterns can be used to generate novel move-

ments, we created both the movements in the library and the novel target movements using the same generating process (see Section 5.1.1). It would be interesting to understand how effective a library of gain patterns can be at generating novel movements when they are generated in a different way (for example, by changing σ in Eqn. (3.3), one can change the duration of the novel target movements; see Chapter 6). One could also create a library of gain patterns that correspond to many different movement shapes (by changing ℓ in Eqn. (3.3)) and durations (by changing σ in Eqn. (3.3)). (See Section 6.6 for a discussion related to this.) One might expect that such a comprehensive movement library may generate a large variety of novel movements, however, the accuracy by which it can generate each movement may be small.

In addition to the above discussion, it is unclear what a 'novel' movement really is. Realistically, 'novel' movements will always bear some resemblance to previously performed (or learned) movements.The use of brain-machine interfaces have allowed researchers glimpses into what types of movements are difficult to generate on short time scales (Carmena et al., 2003; Golub et al., 2018; Sadtler et al., 2014). It has been observed that macaque monkeys can rapidly generate new movements if the new movement corresponds to neural activity that is relatively similar to neural activity associated with previously learned movements (Golub et al., 2018; Sadtler et al., 2014). Therefore, it appears that the brain's ability to rapidly generate new movements requires that the new movements are relatively similar to previously learned movements, which is in line with the approach that we used in this chapter (Sadtler et al., 2014).

5.A Supplementary simulation details

In this appendix, we provide more details on the simulations that we performed in this chapter.

In Fig. 5.3a, for an example target (and for $l = 2$, $l = 4$, $l = 8$, and $l = 16$), we

plot the output and fit that produce the 50th-smallest error between the output and the target across the 100 randomly-generated libraries (see Section 5.1.1).

For each l and for each randomly-generated combination of library elements, we order the 100 novel target movements based on the error between the output and the fit, and we select the one that is the 50th smallest (i.e., close to the median error). We then extract the output and fit errors for this target and repeat this procedure for each of the 100 randomly-generated combinations of library elements and for $l = 1, \ldots, 50$. We plot these results in Figs. 5.3b and 5.5d. In Fig. 5.3, we plot results for $l \in \{1, 2, \ldots, 20\}$; in Fig. 5.5, we plot results for $l \in \{1, 2, \ldots, 50\}$. Observe that there is only a small change in the errors between $l = 20$ and $l = 50$. In 5.5b, we calculate the median error over the 100 target movements and we plot the distribution of these median errors over the 100 randomly-generated combinations of library elements for $l = 5$ and $l = 20$.

Additionally, for each l and for each of the 100 target movements, we order the 100 combinations of library elements based on the error between the output and the fit, and we select the one that is the 50th smallest. We then extract the output and fit errors for this combination and repeat this procedure for each of the 100 target movements and for $l = 1, \ldots, 50$. We plot these results in Fig. 5.5e. This indicates that we obtain qualitatively similar results if we average over the 100 target movements or if we instead average over the 100 combinations of library elements. In Figs. 5.3c and 5.5a, we first calculate the median error over the 100 target movements for each l and for each of the 100 combinations of library elements. We then plot the median of these errors over the 100 combinations of library elements for each l.

We also calculate the Pearson correlation coefficient between the output and the fit errors for each l when taking the 50th-smallest error across the 100 novel target movements (see 5.5f) or across the 100 randomly-generated samples (see 5.5g).

We also repeat these simulations using a baseline rate $r_0 = 5$ Hz and we plot the results of these simulations in Fig. 5.6. Therefore, the simulation details for Figs. 5.3a–c also apply to Figs. 5.6a–c, and the simulation details for Figs. 5.5f–g also apply to Figs. 5.6d–e.

5.B Supplementary figures

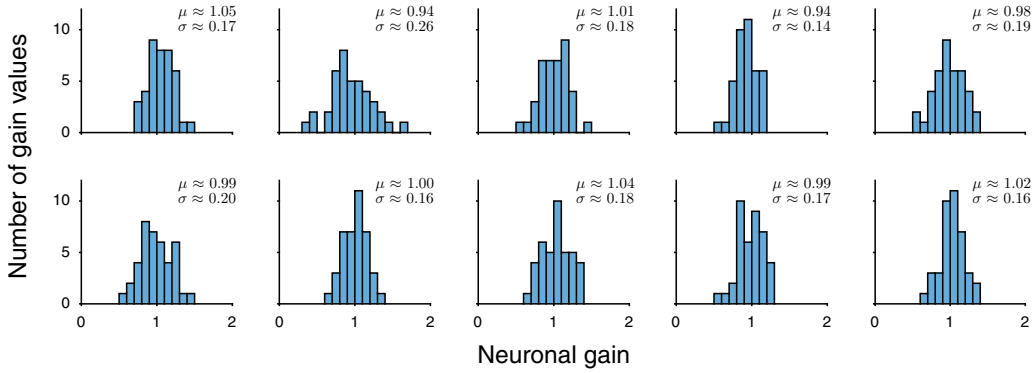

Figure 5.4: We plot the distribution of gain values for 10 *of the* 100 *trained gain patterns that we use to generate movement libraries (see Section 5.1.1 for details of how we create these libraries of gain patterns). We chose these* 10 *gain patterns uniformly at random. We show the mean* μ *and standard deviation* σ *for each of the* 10 *gain patterns. We obtain a mean (over the* 100 *gain patterns) standard deviation (*σ*) of approximately* 0.177 *and a mean mean (*μ*) of approximately* 0.996*.*

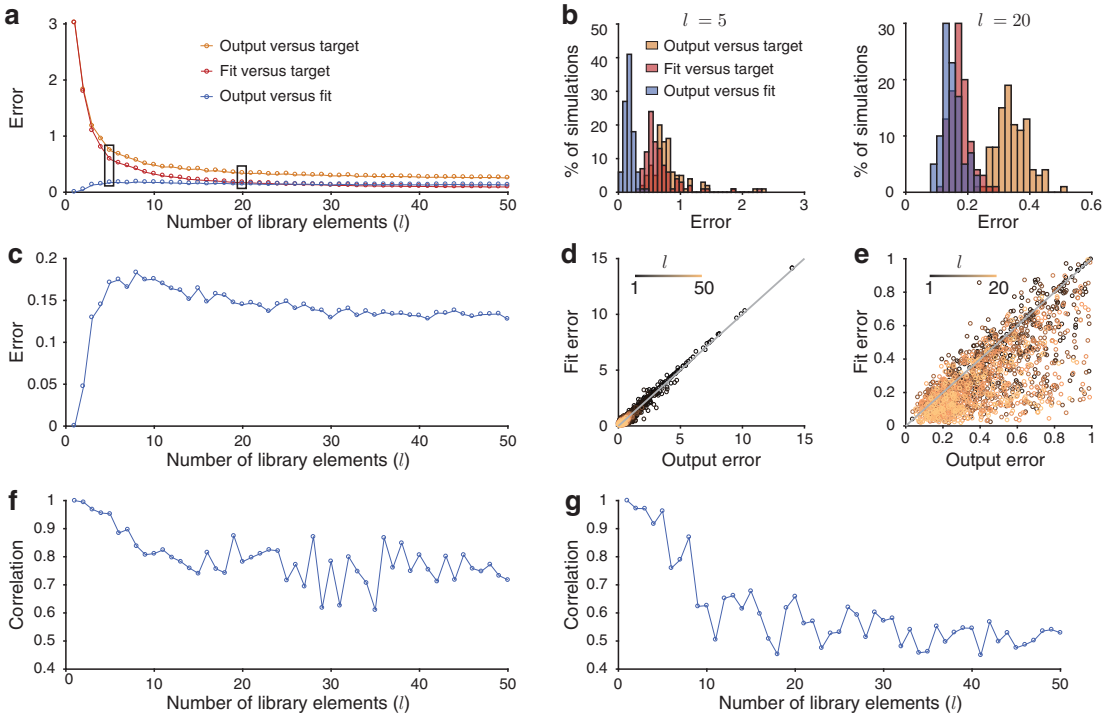

Figure 5.5: **a**, *We show the same plot as in Fig. 5.3c, but for up to* $l = 50$ *library elements.* **b**, *The distributions of errors across* 100 *different libraries for (left)* $l = 5$ *and (right)* $l = 20$ *(see the black rectangles in panel (a)). (Note the difference in horizontal-axis scales in the two plots.)* **c**, *The error between the output and the fit from panel (a) using a different vertical axis scale.* **d**, *The same plot as in Fig. 5.3b, but for* $l = 1, \ldots, 50$ *and with extended axes. Each point represents the* 50*th-smallest error between the output and the fit across* 100 *novel target movements. We show the identity line in grey.* **e**, *The same as in panel (d), but each point represents the* 50*th-smallest error between the output and the fit across the* 100 *libraries for each of the* 100 *novel target movements. We plot these data in the square* $[0, 1] \times [0, 1]$ *and for* $l = 1, \ldots, 20$. **f**, *For the data in panel (d), we plot the Pearson correlation coefficient between the output and the fit errors for each number of library elements (up to* $l = 50$*).* **g**, *For the data in panel (e), we plot the Pearson correlation coefficient between the output and the fit errors for each number of library elements (up to* $l = 50$*). (See Appendix 5.A for further details.)*

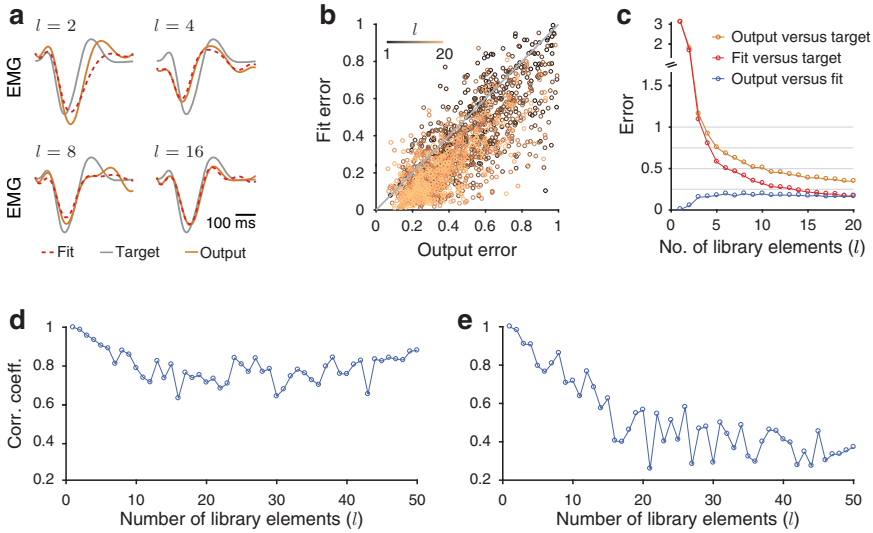

Figure 5.6: Gain patterns as motor primitives with $r_0 = 5$ Hz. **a**, *Example target (grey), fit (dashed red), and output (orange) that produces the 50th-smallest output error over 100 randomly-generated combinations of l library elements using $l = 2$, $l = 4$, $l = 8$, and $l = 16$.* **b**, *Fit error versus the output error for 100 randomly-generated combinations of l library elements for $l = 1, \ldots, 20$. Each point represents the 50th-smallest error between the output and the fit across 100 novel target movements. We show the identity line in grey.* **c**, *Median errors of the 100 randomly-generated combinations of l library elements versus the number of library elements.* **d**, *For the data in panel (b), we plot the Pearson correlation coefficient between the output and the fit errors for each number of library elements (up to $l = 50$).* **e**, *The same as panel (d), but for data corresponding to the 50th-smallest error for each novel target movement, rather than for each randomly-generated combination of library elements (up to $l = 50$) (see Appendix 5.A). Compare panels (d) and (e) of this figure with panels (f) and (g) in Fig. 5.5.*

CHAPTER 6

Gain modulation can control movement speed

In this chapter, we investigate whether changing neuronal gains in recurrent neuronal networks can control the speed of movements. Given a network that produces an initial movement lasting approximately 0.5 s, we find that we can generate the same movement shape, but lasting 5 times longer, by changing only neuronal gains. We can also find two gain patterns so that linearly interpolating between them generates the same movement shape at intermediate speeds. Thus, we can smoothly control the speed of movements by linearly changing neuronal gains. We can also obtain separate families of gain patterns that affect either only the shape or only the speed of a movement, thereby enabling efficient and independent movement control in space and time. Finally, we show that we can learn to combine an existing library of gain patterns to generate new movement shapes while preserving movement speed control. In other words, we create a motor primitives library of gain patterns for controlling movements in space and time. The results that we present in this chapter form part of an article that has been accepted at Nature Neuroscience: *Jake P. Stroud, Mason A. Porter, Guillaume Hennequin, and Tim P. Vogels, 'Motor primitives in space and time via targeted gain modulation in cortical*

networks'.

6.1 Introduction

Thus far in this thesis, we have demonstrated that simple (even coarse, group-based) gain modulation enables control of network outputs of the same, fixed duration (approximately 0.5 s; see Section 3.2.2). To control movements of different durations, motor networks must be able to slow down or speed up muscle outputs (i.e., change the duration of movements without affecting their shape). Several recent studies have investigated potential mechanisms of how recurrent neuronal networks can control the speed of network outputs (Hardy and Buonomano, 2018; Laje and Buonomano, 2013; Rajan et al., 2016; Remington et al., 2018; Wang et al., 2018). One possibility is that by scaling a tonic input to a recurrent neuronal network (which has a pre-trained architecture), one can control the speed of network outputs (Hardy et al., 2017; Remington et al., 2018; Wang et al., 2018). The results of using such an approach to control movement speed appear to align with experimental observations in medial frontal cortex; neurons in this region typically display complex heterogeneous firing-rate activity that scales approximately temporally when monkeys perform tasks that require speed-specific movements to be produced (Remington et al., 2018; Wang et al., 2018). Such studies also suggest that changes in the effective input–output gain (due to a tonic input) of neurons may be important for controlling the speed of neuronal activity (Wang et al., 2018). Furthermore, the neuromodulator dopamine, which can affect neuronal input–output responses (Hernandez-Lopez et al., 2000; Thurley et al., 2008), appears to play a fundamental role in how the brain encodes time (Soares et al., 2016). Therefore, tonic inputs (which are similar to gain changes), or neuromodulatory inputs are likely important for controlling the speed of neuronal activity.

In our model, we have already shown that increasing the gain of all neurons identically in a recurrent neuronal network, extends the duration of neuronal firing

rates (although the frequency of oscillation of neuronal activity also increases; see Fig. 2.4b). However, the frequency of oscillation of the neuronal activity is also affected. Alternatively, it is conceivable that by changing neuronal gains independently, we can change the speed of network outputs without affecting their shape.

In this chapter, we investigate whether neuronal gain modulation in recurrent neuronal networks allows control of the speed of intended movements. In Section 6.3.1, we investigate the possibility of generating the same movement shape, but lasting 5 times longer, by changing only neuronal gains using our learning rule in Eqn. (3.5). We then use back-propagation to train the neuronal gains (see Section 6.3.2) and we show that we can learn a single gain pattern that generates a slow-variant of multiple different movements associated with different initial conditions (see Section 6.3.3). In Section 6.4, we demonstrate that we can smoothly control the speed of multiple movements by simply linearly interpolating between two learned gain patterns. We then show that we can separately change movement speed while preserving movement shape (see Section 6.5), and such an approach allows one to linearly combine previously learned gain patterns to generate new movement shapes while maintaining movement speed control (see Section 6.6). (The simulations for Figs. 6.1c, 6.2, 6.3, 6.4, and 6.6b,d,f were performed in collaboration with Guillaume Hennequin.)

6.2 Methods

Our model setup in this chapter is similar to the one that we used in Chapters 3–5. For the sake of convenience, we describe our model below.

6.2.1 Neuronal dynamics

We use recurrent neuronal networks of $N = 2M$ neurons (of which M are excitatory and M are inhibitory) for which the neuronal activity $\boldsymbol{x}(t) = (x_1(t), \ldots, x_N(t))^\intercal$

evolves according to the dynamical system

$$\tau \frac{d\boldsymbol{x}(t)}{dt} = -\boldsymbol{x}(t) + \boldsymbol{W} f(\boldsymbol{x}(t); \boldsymbol{g}) \,, \tag{6.1}$$

from some initial condition $\boldsymbol{x}(0) = \boldsymbol{x}_0$. In Eqn. (6.1), $f(\boldsymbol{x}; \boldsymbol{g})$ denotes the element-wise application of the static scalar gain function f to the neuronal activity vector $\boldsymbol{x}$. In keeping with Hennequin et al. (2014), we set the single-neuron time constant to be $\tau = 200$ ms. The gain function f, which governs the transformation of neuronal activity $\boldsymbol{x}$ into firing rates relative to a baseline rate r_0, is

$$f(x_i; g_i) = \begin{cases} r_0 \tanh(g_i x_i / r_0) \,, & \text{if } x_i < 0 \,, \\ (r_{\max} - r_0) \tanh(g_i x_i / (r_{\max} - r_0)) \,, & \text{if } x_i \geq 0 \,, \end{cases} \tag{6.2}$$

where the gain value g_i is the slope of the function f at $x_i = 0$ and thus g_i controls the input–output sensitivity of neuron i (Rajan et al., 2010). We use a baseline rate of $r_0 = 20$ Hz and a maximum firing rate of $r_{\max} = 100$ Hz (see Section 3.7 for a discussion of why we choose these values). As we mentioned in Section 1.2.1, $f(\boldsymbol{x}; \boldsymbol{g})$ describes the neuronal firing rates relative to the baseline r_0.

We generate the synaptic weight matrix $\boldsymbol{W}$ in line with Hennequin et al. (2014), that is, we use stability-optimised circuits (see Section 1.2.4). Unless we state otherwise, we use the initial condition $\boldsymbol{x}(t = 0) = \boldsymbol{x}_0$ that maximises the evoked energy $\varepsilon(\boldsymbol{x}_0)$ in Eqn. (C.20) (see Appendix C.2 for a discussion of how to obtain such an initial condition). With this setup, the neuronal dynamics governed by Eqn. (6.1) display complex multiphasic activity transients that last approximately 0.5 s (i.e., similar to those observed in motor cortex when executing movements (Churchland et al., 2012; Hennequin et al., 2014)). For all of our simulations in this chapter, we use a 400-neuron network with 40 random modulatory groups (see Section 4.1.1 for a discussion of how we determine such groups).

6.2.2 Creating target muscle activity

In this chapter, we generate target muscle activities of duration $T = 2.5$ s. We draw muscle activity from a Gaussian process with a covariance function $K \in [0, T] \times [0, T] \to \mathbb{R}_{\geq 0}$ that consists of a product of a squared-exponential kernel (to enforce temporal smoothness) and a non-stationary kernel that produces a temporal envelope similar to that of real electromyogram (EMG) data during reaching (Churchland et al., 2012). Specifically,

$$K(t, t') = e^{-\frac{(t-t')^2}{2\ell^2}} \times E(t/\sigma) \times E(t'/\sigma)\,, \tag{6.3}$$

where $E(t) = te^{(-t^2/4)}$ and we set $\sigma = 550$ ms and $\ell = 250$ ms (see Section 2.6 for a discussion on how the parameters σ and ℓ affect the resulting muscle activity). We also multiply the resulting muscle activity by a scalar to ensure that it has the same order of magnitude as the neuronal activity, and we use a sampling rate of 200 Hz (i.e., each movement consists of 500 evenly-spaced points). We then construct a 'fast' (0.5 s) variant of each movement. We sample fast variants using 100 evenly-spaced points, and we then augment 400 instances of 0 values to the final 2 s of the movement to ensure that both 'fast' (0.5 s) and 'slow' (2.5 s) movement variants have the same length (see the top right of Fig. 6.1a).

In other words, target muscle activity that we generate corresponds to $\boldsymbol{Lu}$ where $\boldsymbol{LL}^\intercal = \boldsymbol{K}$, $\boldsymbol{K} \in \mathbb{R}^{500\times 500}$ is the covariance function, $\boldsymbol{u} \in \mathbb{R}^{500}$, and $\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ (i.e., elements of the vector $\boldsymbol{u}$ are drawn independently and identically from a Gaussian distribution with mean 0 standard deviation 1).

Note that we are modelling network output as a proxy for muscle-force activity. When we generate fast and slow movement variants, we scale the duration of the muscle activity without changing its amplitude (see our generation process in the paragraph immediately above). To actually generate the same movement so that it lasts 5 times longer, we also need to scale the amplitude of the muscle activity by the factor $1/5^2 = 1/25$. To demonstrate the effectiveness of learning

through gain modulation, we omit this scaling, so the tasks on which we train are more difficult ones, as the target activity without the scaling has a substantially larger amplitude throughout the movement. (However, see Fig. 6.7 in Appendix 6.B where we do scale the amplitude of muscle activity when creating fast and slow movement variants.) Alternatively, it may be possible for gain modulation of downstream motoneurons in the spinal cord to account for scaling of the amplitude of muscle activity when performing movements at different speeds (for example, see Vestergaard and Berg (2015)). Finally, if the output of cortical motor circuits always has a large amplitude, no matter the speed of the movement, then this signal is more robust to noise compared with if cortical motor circuits produce very low-amplitude outputs for slow movements. It would be interesting to measure experimentally the relative contributions to amplitude scaling from cortical motor circuits and downstream motoneurons when performing movements at different speeds.

6.2.3 Network output

As in Chapters 2–5, we compute the network output activity $z(t)$ at time t as a weighted linear combination of excitatory neuronal firing rates:

$$z(t) = \boldsymbol{m}^{\mathsf{T}} f(\boldsymbol{x}^E(t); \boldsymbol{g}^E) + b\,, \tag{6.4}$$

where $\boldsymbol{m}\,, \boldsymbol{x}^E(t)\,, \boldsymbol{g}^E \in \mathbb{R}^M$, the quantity M is the number of excitatory neurons, $\boldsymbol{x}^E(t)$ is the excitatory neuronal activity, and f is the gain function (see Eqn. (6.2)). See each subsequent section for details on how we set the readout weights $\boldsymbol{m}$ and the offset b. (For the remainder of this chapter, we will use the term 'readout weights' to denote the collection of both $\boldsymbol{m}$ and b.)

6.3 Learning slow-movement variants through gain modulation

In this section, we investigate whether we can learn gain patterns that generate the same movement shape but lasting 5 times longer.

6.3.1 Training using our learning rule

To investigate whether we can learn gain patterns that generate slow-movement variants (see Fig. 6.1a), we generate 10 different movement shapes as we described in Section 6.2.2. For each of the 10 movements, we fit readout weights using least-squares regression, such that with all gains set to 1, the network output generates the fast variant (which lasts approximately 0.5 s). To ameliorate any issues of overfitting, we use 100 noisy trials, in which we add Gaussian white noise to the initial condition $\boldsymbol{x}_0$ for each trial with a signal-to-noise ratio of 30 dB (Hennequin et al., 2014). We use a 400-neuron network with 40 random modulatory groups (see Section 4.1.1 for a discussion of how we determine such groups). For each of the 10 movements, we train the neuronal gains for each of the 40 modulatory groups using our learning rule in Eqns. (3.5) and (3.6) so that the network output generates the slow-movement variant. (The initial condition $\boldsymbol{x}_0$ and readout weights remain fixed.) We use $60,000$ training iterations, and we run 10 independent training sessions for each of the 10 different movements.

We find that it is possible to generate slow-movement variants using our learning rule (see Fig. 6.1b). Thus, through only gain changes, we can generate the same movement shape so that it lasts 5 times longer. As expected, we observe that the neuronal firing rates associated with the slow variant decay to baseline more slowly than the fast variant (see the bottom panel of Fig. 6.1b). Some movements appear to be more difficult to learn than others (see Fig. 6.6a in Appendix 6.B) and the distribution of neuronal gains after training is different to what we obtained previously when learning target movements that last 0.5 s (compare Fig. 6.6c in

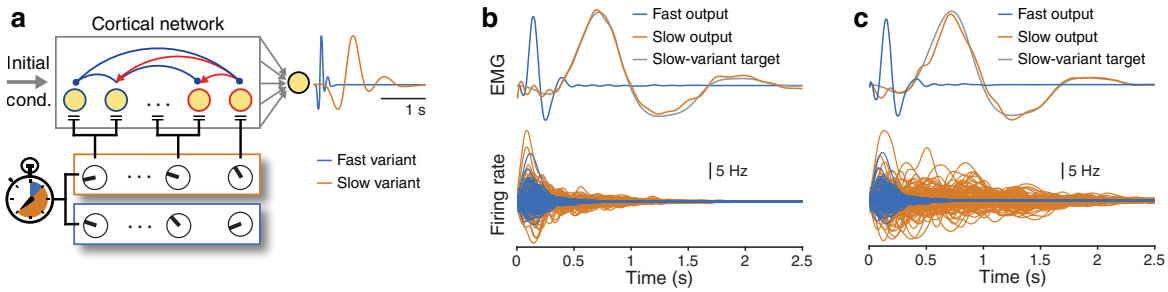

Figure 6.1: Learning slow-movement variants through gain modulation. **a**, *Schematic of gain patterns for fast (0.5 s) and slow (2.5 s) movement variants. (Here and throughout this chapter, we show the former in blue and the latter in orange.) We train a 400-neuron network using 40 random modulatory groups for all simulations (see Section 6.2).* **b**, *(Top) We train a network to extend its output from a fast to a slow-movement variant using our reward-based learning rule. (Bottom) Example firing rates of 50 excitatory and 50 inhibitory neurons for both fast and slow speed variants.* **c**, *The same as panel (b), but now we use a back-propagation algorithm to train the neuronal gains (see Section 6.3.2).*

Appendix 6.B with Fig. 5.2a). After training, the real part of the eigenvalues of the linearised version of Eqn. (6.1) around $\boldsymbol{x} = \mathbf{0}$ are all below 0 (see the left panel of Fig. 6.6e in Appendix 6.B). Therefore, the equilibrium point at $\boldsymbol{x} = \mathbf{0}$ remains stable after training although many of the gain values have changed substantially (see Fig. 6.6c in Appendix 6.B).

We also perform the same task as the one that we show in Fig. 6.1b but when we scale the amplitude of the slow-variant target movement by the factor $1/25$ (see Fig. 6.7 in Appendix 6.B). Scaling the slow-variant target movement by this factor corresponds to the same actual movement but lasting 5 times longer (see Section 6.2.2). We find that we can also accurately generate the amplitude-scaled slow-variant target (see Fig. 6.7 in Appendix 6.B). Therefore, this suggests that gain modulation can also account for the scaling of muscle activity when performing movements at different speeds. However, as we noted in Section 6.2.2, it may be possible for gain modulation of downstream motoneurons in the spinal cord to account for scaling of the amplitude of muscle activity when performing movements

at different speeds (for example, see Vestergaard and Berg (2015)).

6.3.2 Training using back-propagation

Although the slow-movement variants can be learned successfully using our learning rule Eqn. (3.5), we find that the slow-variant outputs tend to be more sensitive to noisy initial conditions than the fast variants (see the left panel of Fig. 6.8 in Appendix 6.B). Furthermore, in Fig. 6.1b, we see that the neuronal firing rates have decayed substantially towards baseline after approximately 0.75 s, even though the output activity is close to its maximum value. Therefore, a small change in the initial condition would likely substantially affect the neuronal activity for times after approximately 0.75 s.

We therefore perform the task that we showed in Fig. 6.1b (i.e., generating slow-movement variants by changing neuronal gains) using a gradient-descent training procedure with gradients that we obtain from back-propagation (Rumelhart et al., 1986). Together with learning the gain pattern for the slow variant, we jointly optimise a single set of readout weights (shared by both the fast-movement and slow-movement variants), as we discussed in Section 6.2.3, as part of the same training procedure. We still fix the gains at 1 for the fast variant. The cost function for the training procedure is equal to the squared Euclidean 2-norm between actual network outputs and the corresponding target outputs at both fast and slow speeds plus the Euclidean 2-norm of the readout weights, where the latter acts as a regulariser. We run gradient descent for 500 iterations, which is well after the cost has stopped decreasing. (Note that we require many fewer training iterations for this task when we train using back-propagation compared with using our reward-based learning rule; compare the x-axis scales in Figs. 6.6a and b.)

Using the target movement from Fig. 6.1b, we plot the output of the back-propagation training procedure in Fig. 6.1c, and we plot results of all simulations in Figs. 6.6b,d in Appendix 6.B on the same 10 target movements as those that

we used in Fig. 6.6a. Following training, the slow variants are learned successfully (see Figs. 6.1c and 6.6b in Appendix 6.B) and are less sensitive to the same noisy initial conditions (see Fig. 6.8 in Appendix 6.B). The neuronal firing rates oscillate transiently, with a substantially lower frequency than either the fast variants or the slow variants trained by our reward-based learning rule. (Compare the bottom panels of Figs. 6.1b and 6.1c.)

6.3.3 Controlling the speed of multiple movements associated with different initial conditions

In Fig. 6.1, we trained a single gain pattern to generate a slow variant of a given movement shape. However, it may be more useful if a single gain pattern can generate slow-movement variants of multiple different movement shapes associated with different initial conditions (ICs) of the neuronal activity. We test this possibility by generating a collection of m orthogonal ICs, in which each IC evokes neuronal activity of approximately equal amplitude with all gains set to 1 (see Appendix 6.A.1 for how we generate such ICs). Given m ICs, we also uniformly-at-random choose m fast target movements and their slow counterparts (see Section 6.2.2) out of a fixed set of 10 different movements. We then train a 400-neuron network (with 40 random modulatory groups; see Section 6.2) to generate the correct fast and slow target movements by optimising a single set of readout weights (shared by both fast and slow variants) and a single gain pattern that generates the slow variants. (We set the gains for each of the fast variants to 1.) We train using the same gradient-descent method with back-propagation that we described in Section 6.3.2.

We find that a single gain pattern can slow down multiple (up to approximately five) distinct movements, which result from five orthogonal ICs, by a factor of 5 (see Fig. 6.2). Errors tend to increase with an increasing number m of slow-movement variants that one wishes generate with a single gain pattern (see Figs. 6.2a,b). Consequently, based on the results that we have presented in Section 6.3, one

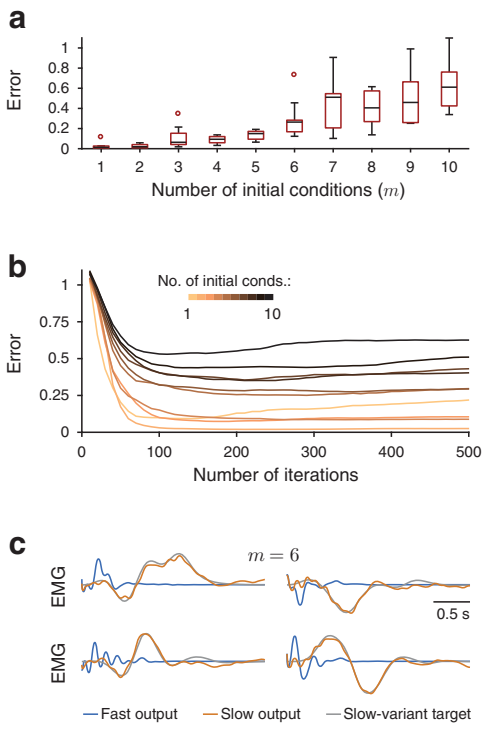

Figure 6.2: Learning multiple slow-movement variants through gain modulation. **a**, *Box plot of the slow-variant errors after training for* 10 *independent training sessions for each number* m *of initial conditions for* $m = 1, \ldots, 10$. *We use Tukey style for the whiskers. (See Appendix 6.A.1 for further simulation details.)* **b**, *Mean error over* 10 *training sessions for* $m = 1, \ldots, 10$ *initial conditions.* **c**, *For the case of* 6 *initial conditions, we plot* 4 *example outputs that correspond to the* 5*th-smallest error across the* 10 *training sessions.*

can extend the temporal scale of transient neuronal activity several-fold and for multiple movements through specific changes in neuronal gains.

6.4 Smoothly controlling the speed of movements

Following training on a slow and a fast variant of the same movement (see Section 6.3), we find that naively interpolating between the two gain patterns does not yield the same movement at intermediate speeds (see Fig. 6.3a), consistent with human subjects being unable to consistently apply learned movements at novel speeds (Collier and Wright, 1995; Hardy et al., 2017). Therefore, even when we consider 'fast' and 'slow' variants of the same movement, both our learning rule and the back-propagation training do not learn to 'slow down' the movement; instead, they learn two seemingly unrelated gain patterns.

However, it is possible to modify our back-propagation training procedure by including additional constraints on the fast and slow gain patterns (see Appendix 6.A.2)

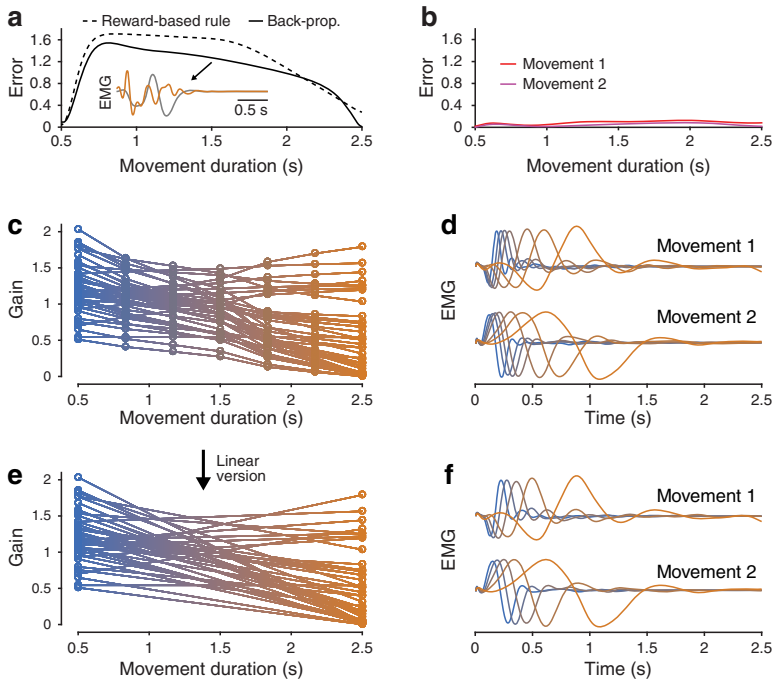

Figure 6.3: Smoothly controlling the speed of movements through gain modulation. **a**, *Interpolation between fast and slow gain patterns does not reliably generate target outputs of intermediate speeds when trained only at the fast and slow speeds. We show an example output (orange) that lasts a duration of* 1.5 *s and the associated target (grey).* **b**, *Linear interpolation between fast and slow gain patterns can successfully generate target outputs when trained at* 5 *additional intermediate speeds. We train* 1 *set of gain patterns (see panel (c)) on two target outputs associated with* 2 *different initial conditions (see Appendix 6.A.2). (We plot these results with the same axis scale as in panel (a).)* **c**, *The* 7 *optimised gain patterns for all* 40 *modulatory groups when training at* 7 *evenly-spaced speeds. (We call this collection of* 7 *trained gain patterns a 'speed manifold'.)* **d**, *We show outputs that result from the* 7 *trained gain patterns from panel (c) for both initial conditions (see Appendix 6.A.2).* **e**, *We can linearly interpolate between the fast and the slow gain patterns from panel (c). We use this interpolation to generate the outputs we show in panel (f).* **f**, *Outputs from both initial conditions for* 5 *evenly-spaced speeds between the fast and slow gain patterns from panel (e).*

so that interpolating between the two gain patterns produces progressively faster or slower outputs (see Fig. 6.3b). We successfully train the network to generate two

movements (associated with two different initial conditions) at 7 different speeds with durations that range from 0.5 s to 2.5 s (see Figs. 6.3c,d and Appendix 6.A.2 for further simulation details). Linear interpolation between the fast and slow gain patterns (see Fig. 6.3e) now generates smooth speed control of both movements at any intermediate speed (see Figs. 6.3b,f). In other words, to control movement speed, we learn a 'manifold' (Gallego et al., 2017) in neuronal gain space that interpolates between the fast and the slow gain patterns (see Fig. 6.9 in Appendix 6.B).

6.5 Joint control of movement shape and speed through gain modulation

Thus far, we have shown that gain modulation can affect either the shape or the speed of a movement. Flexible and independent control of both the shape and speed of a movement (i.e., joint control) necessitates separate representations of space and time in the gain patterns. A relatively simple possibility is to find a single, universal manifold in neuronal gain space (see Section 6.4) for speed control (we call this the 'speed manifold') and combine it with gain patterns that are associated with different movement shapes. Biologically, this may be achievable using separate modulatory systems. We achieve such separation by simultaneously training one speed manifold and 10 gain patterns for 10 different movement shapes such that movements are encoded by the product of shape-specific and speed-specific gain patterns. (See Fig. 6.4a and Appendix 6.A.3.) Following training, we can generate each of the 10 movements at the 7 trained speeds by multiplying a speed-specific gain pattern (see Fig. 6.4b) with the desired shape-specific gain pattern. Importantly, we can also accurately generate each of the 10 different movements at any intermediate speed by simply linearly interpolating between the fast and slow gain patterns (see Figs. 6.4c,d). We thereby obtain separate families of gain

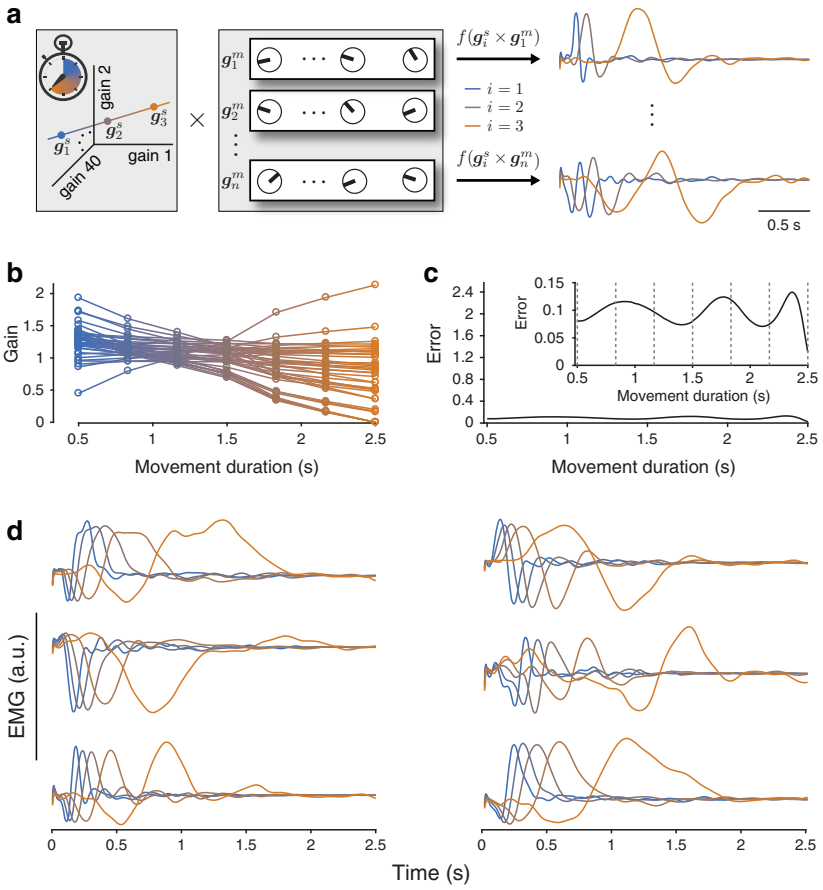

Figure 6.4: Joint control of movement shape and speed. **a**, *One can jointly learn the gain patterns* $\boldsymbol{g}_i^s$ *for (left box) movement speed and* $\boldsymbol{g}_j^m$ *for (right box) movement shape so that the product of two such gain patterns produces a desired movement at a desired speed. In the rightmost panel, we show example outputs for two movement shapes at* 3 *interpolated speeds between the fast and slow gain patterns (see Section 6.5).* **b**, *We show the* 7 *optimised gain patterns for controlling movement speed (i.e.,* $\boldsymbol{g}_i^s$ *for* $i \in \{1, \ldots, 7\}$ *from panel (a)) for all* 40 *modulatory groups when training on* 10 *different movement shapes.* **c**, *Mean error over all* 10 *movements when linearly interpolating between the fast and slow gain patterns for controlling movement speed from panel (b). We use the same vertical axis scale as in Figs. 6.3a,b. In the inset, we plot the same data using a different vertical axis scale. The vertical dashed lines identify the* 7 *movement durations that we use for training.* **d**, *Outputs at* 5 *interpolated speeds between the fast and slow gain patterns for* 6 *of the* 10 *movements. (See Appendix 6.A.3 for further details.)*

patterns for movement shape and speed that independently control movements in space and time.

6.6 Learning gain-pattern primitives to control movement shape and speed

Finally, we investigate whether we can construct new movements with arbitrary durations using previously acquired gain pattern primitives. We examine the possibility of using both the speed manifold and the 10 trained shape-specific gain patterns that we obtained previously (see Fig. 6.4) as a library of spatio-temporal motor primitives. We test this library using 100 novel target movement shapes using a similar approach to the one that we used in Chapter 5. For each target movement, we learn the coefficients for linearly combining the 10 shape-specific gain pattern primitives to obtain each new movement at both the fast and slow speeds while keeping the speed manifold fixed (see Fig. 6.5a and Appendix 6.A.4).

We find that it is possible to accurately generate the new movements at fast and slow speeds using the above spatio-temporal library of gain patterns (see Fig. 6.10 in Appendix 6.B), and we are able to produce the new movements with similar accuracies as those at the fast and slow speeds at any intermediate speed by linearly interpolating between the fast and slow gain patterns from the unaltered speed manifold. (See Fig. 6.5b and the black and red curves in Fig. 6.5c.) The mean error of approximately 0.5 across all movement durations (see the black curve in Fig. 6.5c) is similar to the error that we obtained previously from a movement library that consists of 10 gain patterns (see Fig. 5.3c). We can substantially outperform both the (uniformly-at-random) permuted gain patterns from their associated targets (see Appendix 6.A.4) and using least-squares fitting (which we used previously in Chapter 5) to combine gain patterns. (See the grey and black dashed curves in Fig. 6.5c.)

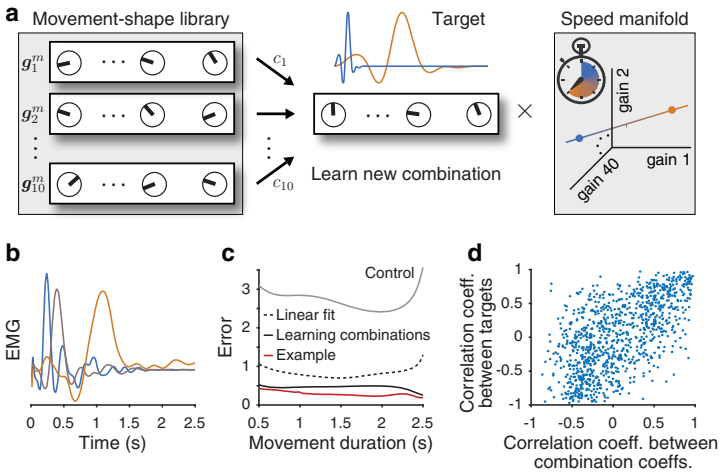

Figure 6.5: Learning gain-pattern primitives to control movement shape and speed. **a**, *We are able to learn to combine (left) previously acquired gain patterns for movement shapes to generate (centre) a new target movement at both fast and slow speeds simultaneously using (right) a fixed manifold in neuronal gain space for controlling movement speed (see Appendix 6.A.4).* **b**, *We plot the output, at* 3 *different speeds, that produces the* 50*th-smallest error (across all* 100 *target movements) between the output and the target when summing errors at both fast and slow speeds.* **c**, *Mean network output error across all* 100 *target movements for all durations when learning to combine gain patterns (black solid curve). We plot the error for the output from panel (b) in red. As a control, we plot the mean error over all target movements when dissociating the learned gain patterns from their target movement by permuting (uniformly at random) the target movements (see the grey curve). We also plot the mean error over all target movements when combining gain patterns using a least-squares fit of the* 10 *learned movement shapes to the target (black dashed curve) (see Appendix 6.A.4). (To generate outputs of a specific duration, we linearly interpolate between the fast and slow gain patterns.)* **d**, *We plot the Pearson correlation coefficient between each pair of target movements versus the Pearson correlation coefficient between the corresponding pair of learned combination coefficients* $c_1, \ldots, c_{10}$.

Consistent with the idea of rapidly generating movements using motor primitives, we generate correlated target shapes by using correlated combinations of gain patterns (see Fig. 6.5d). (Note that we are unlikely to observe correlation

values close to -1 between pairs of combination coefficients because the coefficients $c_1, \ldots, c_{10}$ are likely to sum to approximately 1 (in fact the mean sum of the coefficients is 0.91).) We obtain a similar relationship between coefficient correlations and target correlations as those that we obtained in Fig. 5.3 (see Fig. 6.11 in Appendix 6.B). Therefore, one can use previously learned gain patterns for controlling movement shapes to generate new movements while maintaining independent control of movement speed.

6.7 Conclusions and discussion

Animals can generate a large variety of movements over many different time scales. However, it remains unclear how the brain may achieve such movement control. When performing motor tasks, especially ones in which motor timing is an important factor, neural activity in relevant cortical areas evolves dynamically over time and can change subtly depending on timing-specific task contexts (Crowe et al., 2014; Remington et al., 2018; Russo et al., 2018; Wang et al., 2018). It has been suggested that certain types of inputs (e.g., tonic (i.e., static) or neuromodulatory) to cortical neurons are likely mechanisms by which one can change neural activity so as to produce relevant, timing-sensitive behaviour (Remington et al., 2018; Soares et al., 2016; Wang et al., 2018). Such inputs can affect neural input–output gain sensitivity in the cortical circuit that receives the input (Chance et al., 2002; Thurley et al., 2008). It has also been suggested that effective changes in neural input–output nonlinearities may be a potential mechanism for controlling cortical activity in motor-timing tasks (Wang et al., 2018). Motivated by such observations and our results from Chapter 2, in this chapter, we investigated whether changes in neuronal input–output gains alone in recurrent neuronal networks enables control of the speed of network outputs (i.e., movements).

We found that one can learn gain patterns that generate the same movement shape, but lasting 5 times longer. Moreover, we can find a single gain pattern that

generates a slow-movement variant, lasting 2.5 s, of multiple different movements which are associated with different initial conditions of the neuronal activity. We can also find two gain patterns so that linearly interpolating between them generates smooth speed control of multiple movements. We call such a set of gain patterns for controlling movement speed a 'speed manifold'.

We also investigated whether one can learn gain patterns that affect either only the shape or only speed of an intended movement. We found that we can learn multiple gain patterns associated with different movement shapes together with a single speed manifold where the product of a shape-specific gain pattern with a speed-specific gain pattern on the speed manifold generates the desired movement at the desired speed. Finally, we showed that we can learn to combine previously learned shape-specific gain patterns to generate new movement shapes at fast and slow speeds. Moreover, a previously trained speed manifold can generate the new movements at any intermediate speed.

There are few models of how neural networks can control the speed of network outputs (Hardy et al., 2017; Hass and Durstewitz, 2016; Laje and Buonomano, 2013; Remington et al., 2018; Wang et al., 2018). Most studies focus on training recurrent connections in networks that exhibit chaotic neuronal dynamics (Hardy et al., 2017; Laje and Buonomano, 2013; Remington et al., 2018; Wang et al., 2018) (although one can train the network in such a way so that the neuronal dynamics decay to baseline over time and are no longer chaotic (Hardy et al., 2017)). For example, in Hardy et al. (2017), the authors trained the full weight matrix of a recurrent neuronal network so that two different tonic inputs generate the same neuronal activity but which last different durations; one that lasts 2 s and one that lasts 8 s. In contrast, we used networks with a fixed architecture that automatically exhibit rich transient neuronal activity that resembles experimentally observed cortical recordings (see Section 1.2.4) (Churchland et al., 2012; Hennequin et al., 2014).

Our model is the first to demonstrate that the speed of network outputs can be controlled by changing only neuronal gains and without requiring retraining of the synaptic weight matrix. Also, in contrast to simply changing the single-neuron time constant τ (which uniformly scales the duration, but does not affect the shape of each neuron's activity) or changing a tonic input (which simply changes the equilibrium point of neuronal activity in a stable (at least approximately) linear system (see Section 2.3)), training through gain modulation enables interactions between the shape and duration of network outputs (see Section 6.5).

It would be interesting to understand the differences between training the weight matrix together with changing tonic inputs to control the speed of network outputs, compared with training neuronal gains. Indeed, there may be some level of similarity because a tonic input can be an effective mechanism for mimicking neuronal gain changes (Chance et al., 2002). It would also be important to know whether a tonic input can control the speed of many different movements which are associated with different initial conditions of the neuronal activity. Alternatively, one could investigate whether simply changing the initial condition of the neuronal activity can control the speed of network outputs (although, this approach has been demonstrated to be incompatible with a dataset consisting of recordings of medial frontal cortex of monkeys trained to measure and produce time intervals; see Fig. 8 in Remington et al. (2018)).

The results that we presented in this chapter may also relate to experimental observations that dopamine controls movement vigour (Niv et al., 2007; Panigrahi et al., 2015). For example, to smoothly control movement speed, we find that slow movement variants tend to correspond to gain patterns with a smaller mean gain compared with the fast variants (e.g., see Figs. 6.3c and 6.4b where we find that the mean gain decreases from approximately 1.08 to 0.53 and from approximately 1.26 to 0.73, respectively). This decrease in gain for slow movements is in accordance with observations that dopamine affects both neuronal gain (Hernandez-

Lopez et al., 2000; Thurley et al., 2008) and that decreases in dopamine are associated with slowness of movement in Parkinson's disease (Kaasinen et al., 2014; Marsden, 1989). This also relates to our results from Chapter 2 where we found that increases in global gain generate neural activity transients that have a longer duration but also an increased frequency of oscillation. Thus, to generate slow movement variants through gain modulation, there is a trade-off between extending neural activity transients and reducing their frequency of oscillation. Our results suggest that decreases in mean gain are most suitable for generating slow movement variants (see Figs. 6.3c and 6.4b).

Although there are several plausible approaches for how neuronal networks may generate dynamics at different speeds (Hardy et al., 2017; Remington et al., 2018; Wang et al., 2018), we need more studies linking neural recordings with computational models to uncover mechanisms that the brain uses for such flexible sensorimotor computations.

6.A Supplementary simulation details

6.A.1 Details for Fig. 6.2

In these simulations, we train a single gain pattern that is shared by m different movements, which each last 2.5 s and where each movement corresponds to a different initial condition (IC). To generate a collection of m such ICs, in which each IC evokes neuronal activity of approximately equal amplitude with all gains set to 1, we randomly rotate the top m eigenvectors of the observability Gramian of the matrix $\boldsymbol{W} - \boldsymbol{I}$ (Hennequin et al., 2014). Specifically, we do this by creating a matrix of m columns — one for each of these m eigenvectors — and right-multiplying this matrix by a random $m \times m$ orthogonal matrix (which we obtain via a QR decomposition of a random matrix with elements drawn from a normal distribution with mean 1 and standard deviation 1).

Given m ICs, we uniformly-at-random choose m fast target movements and their slow counterparts (see Section 6.2.2) out of a fixed set of 10 different movements. We then train a recurrent neuronal network to generate the correct fast and slow target movements by optimising a single set of readout weights (shared by both fast and slow variants) and a single gain pattern that generates the slow variants (where we set the gains for each of the fast variants to 1). We train using the same gradient-descent method with back-propagation that we described in Section 6.3.2. We plot the results as a function of the number m of movement–IC pairs (see Fig. 6.2) for 10 independent draws of the ICs that we just described above.

6.A.2 Details for Figs. 6.3 and 6.6

For each of the 10 trained movements we used in Figs. 6.6a,b, we extract the mean minimum error across all simulations for the outputs that we obtain both from our learning rule (see Fig. 6.6a) and from training via back-propagation (see Fig. 6.6b). We then linearly interpolate between the learned gain patterns for the fast and slow outputs, and we calculate the error (see Appendix D) between the output and the target movement at the interpolated speed. We calculate these errors for many interpolated movement durations between 0.5 s and 2.5 s, and we plot the mean errors for both our learning rule and the back-propagation training in Fig. 6.3a. We also show an example output that lasts 1.5 s.

To demonstrate that gain modulation can provide effective smooth control of movement speed for multiple initial conditions of the neuronal activity, we train networks to generate a pair of target movements in response to a corresponding pair of orthogonal initial conditions (see Appendix 6.A.1 for a description of how we generate such initial conditions) at fast and slow speeds and also at each of 5 intermediate, evenly-spaced speeds in between these extremes. To do this, we parametrise the gain pattern of speed index s (with $s \in \{1, \ldots, 7\}$) as a convex combination of a gain pattern $\boldsymbol{g}_{s=1}$ for fast movements and a gain pattern $\boldsymbol{g}_{s=7}$ for

slow movements, with interpolation coefficients of λ_s (with $\boldsymbol{g}_s = \lambda_s \boldsymbol{g}_{s=1} + (1-\lambda_s)\boldsymbol{g}_{s=7}$, $\lambda_1 = 1$, and $\lambda_7 = 0$). We optimise (using back-propagation, as discussed in Section 6.3.2) over $\boldsymbol{g}_{s=1}$, $\boldsymbol{g}_{s=7}$, the 5 interpolation coefficients λ_s (with $s \in \{2, \ldots, 6\}$), and a single set of readout weights. For a given speed s, we use the gain pattern $\boldsymbol{g}_s$ for both movements.

We plot the 7 learned gain patterns in Fig. 6.3c, and we plot their corresponding outputs for both initial conditions in Fig. 6.3d. We show the linear version of the speed manifold (i.e., interpolating between the fast and slow gain patterns) in Fig. 6.3e. For both initial conditions, we plot outputs at 5 evenly-spaced speeds by linearly interpolating between the fast ($\boldsymbol{g}_{s=1}$) and slow ($\boldsymbol{g}_{s=7}$) gain patterns in Fig. 6.3f.

6.A.3 Details for Fig. 6.4

We simultaneously train gain patterns for controlling different movements (i.e., different movement shapes) and their speed. We train a recurrent neuronal network (using back-propagation, as we discussed in Appendix 6.A.2) to generate each of 10 different movement shapes at 7 different, evenly-spaced speeds (ranging from the fast variant to the slow variant) using a single fixed initial condition $\boldsymbol{x}_0$. To jointly learn gain patterns that control movement shape and speed, we parametrise each gain pattern as the element-wise product of a gain pattern that encodes shape (which we use at each speed for a given shape) and a gain pattern that encodes speed (which we use at each shape for a given speed). We again parametrise (see Appendix 6.A.2) the gain pattern that encodes the speed index s (with $s \in \{1, \ldots, 7\}$) as a convex combination of two common endpoints, $\boldsymbol{g}_{s=1}$ (which we use for the fast-movement variants) and $\boldsymbol{g}_{s=7}$ (which we use for the slow-movement variants). We thus optimise over 10 gain patterns for movement shape, 2 gain patterns each for fast and slow movement speeds, 5 speed-interpolation coefficients (see above), and a single set of readout weights.

In Fig. 6.4b, we plot the gain patterns that we obtain for controlling movement speed at each of the 7 trained speeds. In Fig. 6.4c, we show the mean error between the network output and the target over the 10 target movements when generating gain patterns for movement speed by linearly interpolating between the trained fast ($\boldsymbol{g}_{s=1}$) and slow ($\boldsymbol{g}_{s=7}$) gain patterns. In Fig. 6.4d, we plot the outputs of 6 of the 10 gain patterns for movement shape at each of 5 interpolated speeds between the fast and the slow gain patterns. In the rightmost panel of Fig. 6.4a, we plot 2 example movement shapes at 3 interpolated speeds.

6.A.4 Details for Figs. 6.5 and 6.10

For these figures, we use the 10 trained gain patterns for movement shapes, as well as the speed manifold from Fig. 6.4 (see Appendix 6.A.3). Using our learning rule from Eqns. (3.5) and (3.6), we train 10 coefficients $c_1, \ldots, c_{10}$ (with one for each shape-specific gain pattern; see Fig. 6.5a) to construct a new gain pattern that, together with the speed manifold, generates a new target movement at the fast and slow speeds. Specifically, we replace the gains g_i (for $i \in \{1, \ldots, N\}$) with the coefficients c_i (for $i \in \{1, \ldots, 10\}$) in Eqns. (3.5) and (3.6). We use the mean of the errors at the fast and slow speeds. To generate the network output at the fast and slow speeds, respectively, we calculate the element-wise product between the newly-constructed gain pattern and the fast and slow gain pattern, respectively, on the speed manifold. We independently train, using $10,000$ training iterations, the coefficients $c_1, \ldots, c_{10}$ on each of the 100 target movements that we used for Fig. 5.3. In Fig. 6.10, we plot histograms of the errors over the 100 target movements after training for both the fast and slow speeds. We plot the mean error (see the black curve) over all 100 target movements at interpolated speeds in Fig. 6.5c. For the output that produces the 50th-smallest summed errors from fast and slow speeds, we plot the error in red in Fig. 6.5c. As a control, we calculate the mean error between the network output and the target over the 100 target

movements when choosing one of the 100 newly-learned gain patterns uniformly at random without replacement. (See the grey curve in Fig. 6.5c.)

Additionally, instead of learning to combine gain patterns using the method that we described in the previous paragraph, we determine coefficients $c_1, \ldots, c_{10}$ using a least-squares regression by fitting the 10 learned movements to each of the 100 target movements at the fast and slow speeds simultaneously and requiring that $c_j \geq 0$ for all j and $\sum_{j=1}^{10} c_j = 1$ (i.e., we use the same method that we described in Section 5.1.1). (See the black dashed curve in Fig. 6.5c.)

Finally, we plot the Pearson correlation coefficient between pairs of target movements versus the Pearson correlation coefficient between corresponding pairs of learned coefficients $c_1, \ldots, c_{10}$ in Fig. 6.5d. In our visualisation, we plot only $1,000$ of the $4,950$ data points. (We choose these points uniformly at random.) In Fig. 6.11, we also compare these correlations with correlations that we obtain from Fig. 5.3. For the case of 10 library elements, we choose one of the 100 randomly-generated combinations of 10 library elements uniformly at random and calculate correlations between pairs of fitted coefficients and corresponding pairs of targets. We again plot only $1,000$ of the $4,950$ data points. (We choose these points uniformly at random.)

6.B Supplementary figures

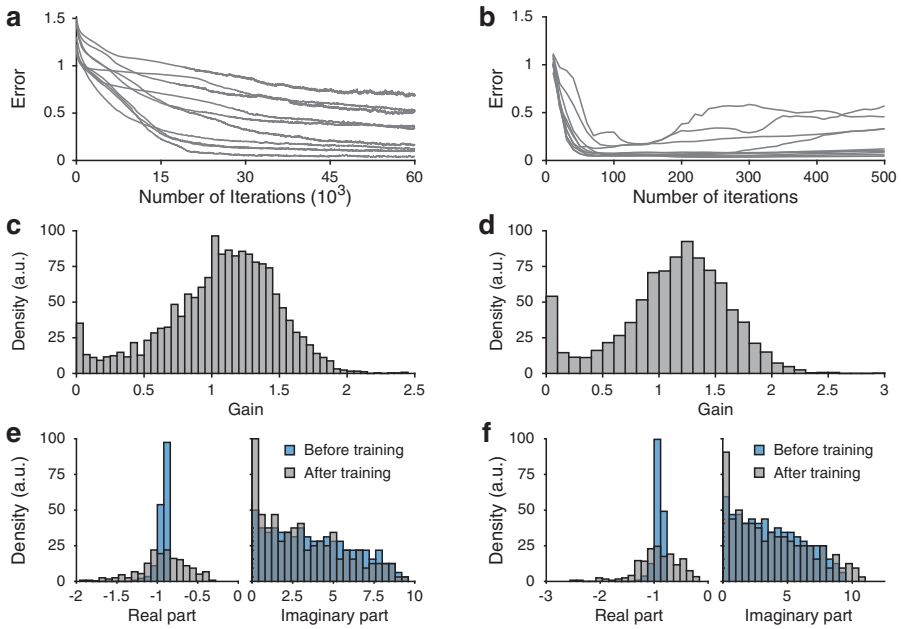

Figure 6.6: Additional results for controlling movement speeds through gain modulation. **a**, *Mean error over* 10 *training sessions for each of* 10 *different movements when learning gain patterns for slow-movement variants using our reward-based learning rule (see Section 6.3.1).* **b**, *Mean error over* 10 *training sessions for the same* 10 *movements when instead learning gain patterns for slow-movement variants using a back-propagation algorithm (see Section 6.3.2).* **c**, *Distribution of gains for the slow-movement variants across all training sessions using our reward-based learning rule.* **d**, *Distribution of gains for the slow-movement variants across all training sessions when using back-propagation.* **e**, *Histograms of the real and imaginary parts of the eigenvalues of the linearisation of Eqn.* (6.1) *around* $\boldsymbol{x} = \mathbf{0}$ *before and after training using our reward-based rule for the example in Fig. 6.1b.* **f**, *Histograms of the real and imaginary parts of the eigenvalues of the linearisation of Eqn.* (6.1) *around* $\boldsymbol{x} = \mathbf{0}$ *before and after training using the back-propagation algorithm for the example in Fig. 6.1c. (For each simulation, we train a* 400*-neuron network using* 40 *random modulatory groups (see Section 6.3).)*

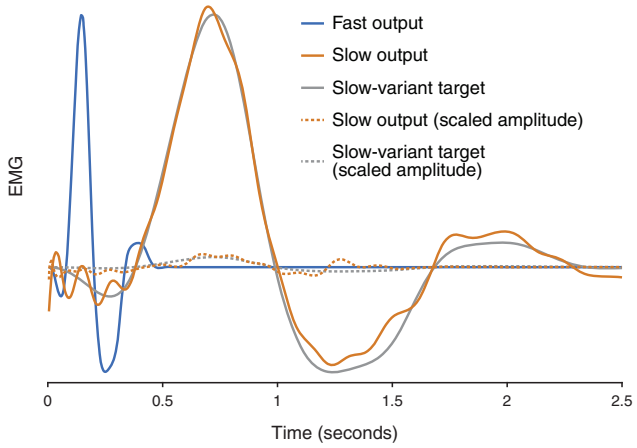

Figure 6.7: Learning slow-movement variants when scaling both the amplitude and duration of target movements. We can accurately perform the same task as the one that we showed in Fig. 6.1b when we also scale the amplitude of the slow-variant target movement by the factor $1/25$ (see the dashed curves). Scaling the slow-variant target movement by this factor corresponds to the same actual movement but lasting 5 times longer (see Section 6.2.2). We also reproduce the results from the top panel of Fig. 6.1b (solid curves) for comparison.

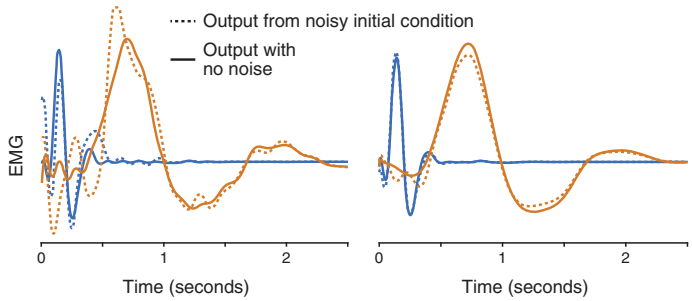

Figure 6.8: On the left and right, respectively, we show the same outputs that we plotted in Figs. 6.1b and 6.1c, but we now add white Gaussian noise (with a signal-to-noise ratio of 4 dB) to the initial condition of the neuronal activity. We observe that the outputs from the back-propagation training procedure (right-hand panel) are less sensitive than the outputs from the learning rule (left hand panel) to noisy initial conditions.

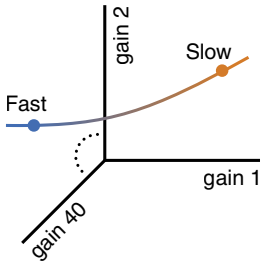

Figure 6.9: We show an illustration of a manifold in neuronal gain space for controlling movement speed when we have 40 *modulatory groups (see Section 6.4).*

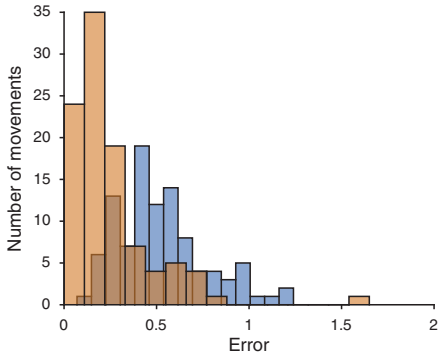

Figure 6.10: We plot histograms of the errors over the 100 *target movements at both fast (blue) and slow (orange) speeds when learning to combine gain patterns using a library of spatio-temporal gain pattern primitives (see Section 6.6).*

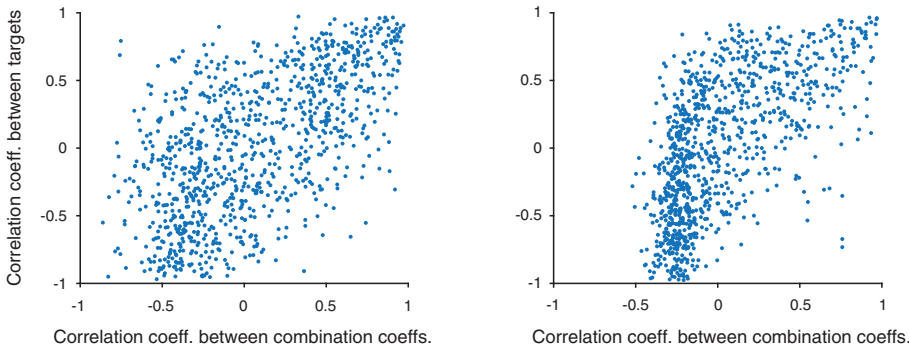

Figure 6.11: On the left, we replot Fig. 6.5d. On the right, we show the same correlations except using data from Fig. 5.3. We show correlations for the case of 10 *library elements from Fig. 5.3 and we choose one of the* 100 *randomly-generated combinations of* 10 *library elements uniformly at random.*

CHAPTER 7

Conclusions and future work

In this thesis, we have shown that modulating the input–output gain of neurons in recurrent neuronal networks is an effective mechanism for controlling the shape and speed of network outputs. We primarily used recurrent neuronal-network models that exhibit neuronal activities reminiscent of monkey primary motor cortex (Churchland et al., 2012; Hennequin et al., 2014). Therefore, such models may inform our understanding of how motor cortex controls and generates movements (Hennequin et al., 2014; Shenoy et al., 2013; Sussillo et al., 2015). Experimental recordings of motor cortex suggest that the population activity is specific to the movement being performed (Churchland et al., 2010, 2012). This population activity can arise through several possible mechanisms. Distinct neuronal activity can emerge from a fixed population-level dynamical system with different movement-specific preparatory states (Churchland et al., 2010). Alternatively, one can change the underlying dynamical system through modification of the effective connectivity (Friston, 2011) even when a preparatory state is the same across movements. Such changes in effective connectivity can arise either through a feedback loop (e.g., a low-rank addition to the synaptic weight matrix (Sussillo and Abbott, 2009))

or through patterns of movement-specific gains, as we explored in this thesis. We found that movement-specific gain patterns provide a similar performance to training a different initial condition for each desired output (with a fixed duration) and that both of these approaches outperform a rank-1 perturbation of the synaptic weight matrix (see Fig. 3.4). Gain modulation thus provides a complementary method of controlling neuronal dynamics for flexible and independent manipulation of output shape. Additionally, we showed that gain modulation provides a compelling mechanism for extending the duration of activity transients without needing to carefully construct movement-specific network architectures. Although our results demonstrate the effectiveness of learning through gain modulation in a variety of tasks, more research is required to understand how effective gain changes can be in other, possibly more complex tasks (e.g., using more readout units and training on working memory tasks (Hoerzer et al., 2014; Sussillo and Abbott, 2009)).

We propose that gain modulation may occur via neuromodulators (Hernandez-Lopez et al., 2000; Thurley et al., 2008; Wei et al., 2014), but it may also arise from a tonic (i.e., static) input that shifts each neuron's resting activity within the dynamic range of its input–output function (Chance et al., 2002) (for example, through inputs from the cerebellum). Although this is an effective way of mimicking gain changes in recurrent neuronal-network models with strongly nonlinear single-neuron dynamics (Sussillo and Barak, 2013; Wang et al., 2018), we were unable to produce desired target outputs by training a tonic input. It is worth noting that a tonic input also modifies baseline neuronal activity, thereby altering the output muscle activities away from rest. Alternatively, a quadratic gain function (Hennequin et al., 2018; Murphy et al., 2016) may enable a tonic input to reliably imitate gain changes because, for example, a monotonically increasing input to a neuron will cause a monotonic increase in the neuron's effective gain.

In our model, in which the recurrent connectivity remains fixed, synaptic modifications may take place upstream of the motor circuit (e.g., in the input synapses to

the presumed neuromodulatory neurons (Martins and Froemke, 2015)). Changes in neuronal gains can also work in concert with plasticity in cortical circuits, thereby allowing changes in the modulatory state of a network to be transferred into circuit connectivity (Swinehart and Abbott, 2005), consistent with known interactions between neuromodulation and plasticity (Frémaux and Gerstner, 2016; Gu, 2002; Luft and Schwarz, 2009; Marder, 2012; Martins and Froemke, 2015). However, future work will be needed to understand the potential interactions between gain modulation and synaptic plasticity in recurrent neuronal-network models. Consequently, understanding the neural basis of motor learning may necessitate recording from a potentially broader set of brain areas than those circuits whose activity correlates directly with movement dynamics.

Our results build on a growing literature of taking a dynamical-systems approach to studying temporally-structured cortical activity. This perspective has been effective for investigations of several cortical regions (Breakspear, 2017; Churchland et al., 2010, 2012; Kao et al., 2015; Mante et al., 2013; Shenoy et al., 2013; Wang et al., 2018). In line with this approach, our results may also be applicable to other recurrent cortical circuits that exhibit rich temporal dynamics (e.g., decision-making dynamics in prefrontal cortex (Mante et al., 2013), temporally-structured memories, etc.).

In this thesis, we used firing-rate models to describe neuronal activity. However, real neurons communicate by sending action potentials (or 'spikes') to one another. Although we discussed some of our primary reasons for using firing-rate models in Section 1.1 (e.g., the greater analytical tractability offered by firing-rate models and because motor cortices appear to operate as dynamical systems, therefore firing-rate models seem a natural choice for modelling such circuits), it is important to understand the effectiveness of learning through gain modulation when using more biologically realistic spiking models (Gerstner and Kistler, 2002). Recently, there have been several studies investigating the relationships between spiking

and rate models when training either model on particular tasks (DePasquale et al., 2016, 2018; Gilra and Gerstner, 2017; Hennequin et al., 2014; Nicola and Clopath, 2017; Schaffer et al., 2013). Some of these studies can obtain qualitatively similar dynamics, regardless of whether one uses a spiking or a rate model (DePasquale et al., 2016; Hennequin et al., 2014; Schaffer et al., 2013). Additionally, spiking models can now be trained in a qualitatively similar manner to training approaches conventionally used for firing-rate models (DePasquale et al., 2018; Gilra and Gerstner, 2017; Nicola and Clopath, 2017). Given several of these recent observations, we thus expect that training through gain modulation in spiking models may also provide an effective mechanism for controlling the shape and speed of network outputs.

In summary, our results support the view that knowing only the structure of neuronal networks is not sufficient to explain their dynamics (Bargmann, 2012; Bassett and Sporns, 2017). We extend current understanding of the effects of neuromodulation (Bargmann, 2012; Kida and Mitsushima, 2018; Marder, 2012; Thurley et al., 2008) and show that it is possible to control a recurrent neuronal network's computations without changing its connectivity. We found that modulating only neuronal responsiveness enables flexible control of cortical activity. We were also able to combine previously learned modulation states to generate new desired activity patterns, and we demonstrated that employing gain modulation allows one to smoothly and accurately control the duration of network outputs. Our results thus suggest the possibility that gain modulation is a central part of learning in cortical circuits.

APPENDIX A

Eigenvalues of a matrix following addition of a diagonal matrix

In this appendix, we provide a lemma and a proof that show how the eigenvalues of a matrix are affected following the addition of a diagonal matrix in which all diagonal entries are identical.

Lemma 1. *Let μ be an eigenvalue of the square matrix $\boldsymbol{A}$ and let $\boldsymbol{A}' = \boldsymbol{A} + c\boldsymbol{I}$, with $c \in \mathbb{R}$. The quantity $\mu + c$ is then an eigenvalue of $\boldsymbol{A}'$. Moreover, the spectrum of $\boldsymbol{A}'$ is the spectrum of $\boldsymbol{A}$ shifted c units in the positive direction.*

Proof. Assume that the matrix $\boldsymbol{A}$ has eigenvector $\boldsymbol{v}$ with eigenvalue μ, so $\boldsymbol{A}\boldsymbol{v} = \mu\boldsymbol{v}$. Let $\boldsymbol{A}' = \boldsymbol{A} + c\boldsymbol{I}$, with $c \in \mathbb{R}$. It follows that

$$\begin{aligned} \boldsymbol{A}'\boldsymbol{v} &= (\boldsymbol{A} + c\boldsymbol{I})\boldsymbol{v} \\ &= \boldsymbol{A}\boldsymbol{v} + c\boldsymbol{v} \\ &= \mu\boldsymbol{v} + c\boldsymbol{v} \\ &= (\mu + c)\boldsymbol{v}\,. \end{aligned}$$

Thus, every eigenvalue μ of $\boldsymbol{A}$ has an associated eigenvalue $\mu + c$ of $\boldsymbol{A}'$. □

APPENDIX B

Solution of linear neuronal dynamics using eigenvectors and eigenvalues

In this appendix, we show how to construct the solution of a linear recurrent neuronal network using the eigenvalues and eigenvectors of the associated system. Let's first write down the linear dynamical system from Eqn. (1.9); it has $N = 2M$ neurons (of which M are excitatory and M are inhibitory) and the neuronal dynamics are given by

$$\tau \frac{d\boldsymbol{x}(t)}{dt} = -\boldsymbol{x}(t) + \boldsymbol{W}\boldsymbol{x}(t)\,, \tag{B.1}$$

with $\boldsymbol{x}(0) = \boldsymbol{x}_0$. We let $\boldsymbol{A} = \frac{1}{\tau}(\boldsymbol{W} - \boldsymbol{I})$. We can then find the eigenvectors $\boldsymbol{v}_i$ and associated eigenvalues λ_i of the matrix $\boldsymbol{A}$ by solving

$$\boldsymbol{A}\boldsymbol{v}_i = \lambda \boldsymbol{v}_i\,. \tag{B.2}$$

The entries of $\boldsymbol{W}$ are always real, so the spectrum of $\boldsymbol{A}$ has only real and/or complex-conjugate pairs of eigenvalues (Strang, 2016). If there are a nonzero number $k \in \mathbb{N}$ of purely real eigenvalues, we label these eigenvalues as λ_i for $i = 1, \ldots, k$. The remaining $N - k$ eigenvalues are complex-conjugate pairs.

Therefore, we label these complex-conjugate eigenvalues as $\lambda_i = \bar{\lambda}_{i+(N-k)/2}$ for $i = k+1, \ldots, (N+k)/2$. We can then write the solution of Eqn. (B.1) as:

$$\boldsymbol{x}(t) = \sum_{i=1}^{N} c_i \boldsymbol{u}_i(t)\,, \tag{B.3}$$

where

$$\boldsymbol{u}_i(t) = \begin{cases} e^{\lambda_i t}\boldsymbol{v}_i\,, & i = 1, \ldots, k\,, \\ e^{\mathrm{Re}(\lambda_i)t}\big(\,\mathrm{Re}(\boldsymbol{v}_i)\cos(\mathrm{Im}(\lambda_i)t) - \mathrm{Im}(\boldsymbol{v}_i)\sin(\mathrm{Im}(\lambda_i)t)\big)\,, & i = k+1, \ldots, (N+k)/2\,, \\ e^{\mathrm{Re}(\lambda_i)t}\big(\,\mathrm{Im}(\boldsymbol{v}_i)\cos(\mathrm{Im}(\lambda_i)t) + \mathrm{Re}(\boldsymbol{v}_i)\sin(\mathrm{Im}(\lambda_i)t)\big)\,, & i = (N+k)/2+1, \ldots, N\,. \end{cases} \tag{B.4}$$

The coefficients c_i can be determined using the (real-valued) initial condition $\boldsymbol{x}_0$ at $t = 0$. We use several steps to get to the purely real solutions in Eqn. (B.4) (e.g., we use Euler's formula); these steps are outlined in several textbooks on dynamical systems, such as in (Teschl, 2012, Section 3.2).

APPENDIX C

Algorithmic procedure for generating stability-optimised circuits and finding preferred initial conditions

In Section 1.2.4, we briefly described the procedure for constructing stability-optimised circuits (Hennequin et al., 2014). In this appendix, we provide a detailed description of the method that we use to create stability-optimised circuits and how to find so called 'preferred' initial conditions that give rise to rich transient neuronal dynamics. This method was first presented in Hennequin et al. (2014).

C.1 Creating stability-optimised circuits

To create a stability-optimised circuit, one starts with a weight matrix $\boldsymbol{W}$ of $N = 2M$ neurons (of which M are excitatory and M are inhibitory), where we label the excitatory population as E and the inhibitory population as I. In block form, the weight matrix is

$$\boldsymbol{W} = \begin{pmatrix} \boldsymbol{W}_{EE} & \boldsymbol{W}_{EI} \\ \boldsymbol{W}_{IE} & \boldsymbol{W}_{II} \end{pmatrix} . \tag{C.1}$$

The probability p of connection between any two neurons is set to 0.1. Nonzero elements of $\boldsymbol{W}$ are set to $w_0/\sqrt{N}$ for excitatory connections and to $-\gamma w_0/\sqrt{N}$ for inhibitory connections, where $w_0^2 = 2\rho^2/(p(1-p)(1+\gamma^2))$. With $\gamma = 1$, this construction results in $\boldsymbol{W}$ having an approximately circular spectrum of radius ρ (Hennequin et al., 2014) (also see Section 1.2.2 and Girko (1983)). We set $\rho = 10$ and the inhibition/excitation ratio to $\gamma = 3$ (Hennequin et al., 2014).

After the construction of the initial $\boldsymbol{W}$, all of the excitatory connections remain fixed. An optimisation algorithm (see steps 1–8 below) is then used to change the inhibitory connections so that the spectral abscissa (i.e., the largest real part in the spectrum) of $\boldsymbol{W}$ is reduced below 1, which implies that the linear neuronal dynamics governed by

$$\tau \frac{d\boldsymbol{x}}{dt} = \boldsymbol{A}\boldsymbol{x}\,, \tag{C.2}$$

where $\boldsymbol{A} = \boldsymbol{W} - \boldsymbol{I}$ and $\boldsymbol{I}$ is the identity matrix, exhibit a stable equilibrium point at $\boldsymbol{0}$. An effective way to reduce the spectral abscissa of $\boldsymbol{W}$ is to actually minimise the 'smoothed spectral abscissa' $\tilde{\alpha}(\boldsymbol{A})$, which is an upper bound of the spectral abscissa $\alpha(\boldsymbol{A})$ that — among other advantages — can be minimised using an algorithmic procedure (Vanbiervliet et al., 2009).

Inhibitory weights are iteratively updated according to the negative matrix derivative $\frac{\partial\tilde{\alpha}(\boldsymbol{A})}{\partial\boldsymbol{A}}$. This matrix derivative is defined based on the *observability* and *controllability* Gramian matrices $\boldsymbol{Q}$ and $\boldsymbol{P}$, respectively, of the system in Eqn. (C.2). The symmetric positive definite matrices $\boldsymbol{Q}$ and $\boldsymbol{P}$ play a fundamental role in linear–quadratic control theory (Dorato et al., 1998).

If $\alpha(\boldsymbol{A}) < 0$, the observability and controllability Gramians $\boldsymbol{Q}$ and $\boldsymbol{P}$, respectively, are given by

$$\boldsymbol{Q} \equiv \frac{2}{\tau}\int_0^\infty e^{\frac{t}{\tau}\boldsymbol{A}^\intercal} e^{\frac{t}{\tau}\boldsymbol{A}} dt \tag{C.3}$$

$$\boldsymbol{P} \equiv \frac{2}{\tau}\int_0^\infty e^{\frac{t}{\tau}\boldsymbol{A}} e^{\frac{t}{\tau}\boldsymbol{A}^\intercal} dt\,. \tag{C.4}$$

Although Eqns. (C.3) and (C.4) are integrals involving quadratic terms, one can

compute $\boldsymbol{Q}$ and $\boldsymbol{P}$ by solving a set of linear equations. This can be seen by noting that

$$\boldsymbol{A}^\intercal \boldsymbol{Q} = \frac{2}{\tau}\int_0^\infty \boldsymbol{A}^\intercal e^{\frac{t}{\tau}\boldsymbol{A}^\intercal} e^{\frac{t}{\tau}\boldsymbol{A}} dt\,, \tag{C.5}$$

$$\boldsymbol{Q}\boldsymbol{A} = \frac{2}{\tau}\int_0^\infty e^{\frac{t}{\tau}\boldsymbol{A}^\intercal} e^{\frac{t}{\tau}\boldsymbol{A}} \boldsymbol{A} dt\,. \tag{C.6}$$

Therefore,

$$\boldsymbol{A}^\intercal \boldsymbol{Q} + \boldsymbol{Q}\boldsymbol{A} = \frac{2}{\tau}\int_0^\infty \boldsymbol{A}^\intercal e^{\frac{t}{\tau}\boldsymbol{A}^\intercal} e^{\frac{t}{\tau}\boldsymbol{A}} + e^{\frac{t}{\tau}\boldsymbol{A}^\intercal} e^{\frac{t}{\tau}\boldsymbol{A}} \boldsymbol{A} dt \tag{C.7}$$

$$= 2\int_0^\infty \frac{d}{dt}(e^{\frac{t}{\tau}\boldsymbol{A}^\intercal} e^{\frac{t}{\tau}\boldsymbol{A}}) dt \tag{C.8}$$

$$= 2e^{\frac{t}{\tau}\boldsymbol{A}^\intercal} e^{\frac{t}{\tau}\boldsymbol{A}}\Big|_0^\infty \tag{C.9}$$

$$= -2\boldsymbol{I}\,. \tag{C.10}$$

Using a similar approach, one can obtain a linear equation for the controllability Gramian $\boldsymbol{P}$. We then have

$$\boldsymbol{A}^\intercal \boldsymbol{Q} + \boldsymbol{Q}\boldsymbol{A} = -2\boldsymbol{I}\,, \tag{C.11}$$

$$\boldsymbol{A}\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A}^\intercal = -2\boldsymbol{I}\,. \tag{C.12}$$

Following Hennequin et al. (2014), we define the 'evoked energy' $\varepsilon(\boldsymbol{x}_0)$ from some initial condition $\boldsymbol{x}_0$ as $\varepsilon(\boldsymbol{x}_0) = \frac{2}{\tau}\int_0^\infty \|\boldsymbol{x}(t)\|_2^2 dt$, and it can be shown that $\varepsilon(\boldsymbol{x}_0) = \boldsymbol{x}_0^\intercal \boldsymbol{Q} \boldsymbol{x}_0$ (Hennequin et al., 2014).

The matrix $\boldsymbol{Q}$ is symmetric positive definite, so all of its eigenvalues are real and positive, thus the maximum eigenvalue is smaller than the sum of all eigenvalues. Additionally, the trace of $\boldsymbol{Q}$ (which we denote by $\mathrm{Tr}[\boldsymbol{Q}]$) is equal to the sum of its eigenvalues simply because $\boldsymbol{Q}$ is a square matrix. Therefore, for the maximum eigenvalue $\lambda_{\max}$ of $\boldsymbol{Q}$ and its associated eigenvector $\boldsymbol{v}$ with $\|\boldsymbol{v}\|_2 = 1$, the evoked energy due to the initial condition $\boldsymbol{v}$ is given by

$$\varepsilon(\boldsymbol{v}) = \boldsymbol{v}^\intercal \boldsymbol{Q} \boldsymbol{v} = \boldsymbol{v}^\intercal \lambda_{\max} \boldsymbol{v} = \lambda_{\max} \boldsymbol{v}^\intercal \boldsymbol{v} = \lambda_{\max} < \mathrm{Tr}[\boldsymbol{Q}]\,. \tag{C.13}$$

Thus, if $\mathrm{Tr}[\boldsymbol{Q}]$ is finite — which implies that $\mathrm{Tr}[\boldsymbol{Q}] < 1/\epsilon$ for some $\epsilon > 0$ — then the evoked energy $\varepsilon(\boldsymbol{v})$ is less than $1/\epsilon$ for the aforementioned ϵ, implying that the

dynamics given by Eqn. (C.2) are asymptotically stable. Note that if ϵ is small, then the neuronal dynamics are close to instability, because $\mathrm{Tr}[\boldsymbol{Q}]$ is large.

For a weight matrix that is not (yet) linearly asymptotically stable, one can ask what is the minimum s such that the spectral abscissa of the matrix $\boldsymbol{W} - s\boldsymbol{I}$ is less than 0. This is equivalent to asking what is the minimum s such that $\mathrm{Tr}[\boldsymbol{Q}(s)] < 1/\epsilon$ for $\epsilon > 0$, where

$$\boldsymbol{Q}(s) \equiv \frac{2}{\tau} \int_0^\infty e^{\frac{t}{\tau}(\boldsymbol{W}-s\boldsymbol{I})^\intercal} e^{\frac{t}{\tau}(\boldsymbol{W}-s\boldsymbol{I})} dt \,. \tag{C.14}$$

From Eqn. (C.11), $\boldsymbol{Q}(s)$ is the solution to the following linear equation:

$$(\boldsymbol{W} - s\boldsymbol{I})^\intercal \boldsymbol{Q}(s) + \boldsymbol{Q}(s)(\boldsymbol{W} - s\boldsymbol{I}) \;=\; -2\boldsymbol{I} \,. \tag{C.15}$$

Analogously for $\boldsymbol{P}(s)$ we also obtain:

$$(\boldsymbol{W} - s\boldsymbol{I})\boldsymbol{P}(s) + \boldsymbol{P}(s)(\boldsymbol{W} - s\boldsymbol{I})^\intercal \;=\; -2\boldsymbol{I} \,. \tag{C.16}$$

It is known that $\mathrm{Tr}[\boldsymbol{Q}(s)]$ is a monotonic decreasing function of s (Vanbiervliet et al., 2009). This makes sense intuitively, because the larger s, the spectrum of $\boldsymbol{W} - s\boldsymbol{I}$ becomes more negative. Thus, the dynamics are 'more' stable, implying that the evoked energy, which is upper-bounded by $\mathrm{Tr}[\boldsymbol{Q}(s)]$, is smaller. Mathematically, the unique s that satisfies $\mathrm{Tr}[\boldsymbol{Q}(s)] = 1/\epsilon$ is called the *ϵ-smoothed spectral abscissa* of $\boldsymbol{W}$ (Vanbiervliet et al., 2009). We denote the ϵ-smoothed spectral abscissa of $\boldsymbol{W}$ by $\tilde{\alpha}_\epsilon(\boldsymbol{W})$.

The quantity $\tilde{\alpha}_\epsilon(\boldsymbol{W})$ is larger than or equal to $\alpha(\boldsymbol{W})$, so if we aim to minimise $\alpha(\boldsymbol{W})$, we can instead seek to minimise $\tilde{\alpha}_\epsilon(\boldsymbol{W})$. Importantly, we can compute the derivative of $\tilde{\alpha}_\epsilon(\boldsymbol{W})$ with respect to $\boldsymbol{W}$ (Vanbiervliet et al., 2009). The gradient of $\tilde{\alpha}_\epsilon(\boldsymbol{W})$ with respect to $\boldsymbol{W}$ is

$$\frac{\partial \tilde{\alpha}_\epsilon(\boldsymbol{W})}{\partial \boldsymbol{W}} = \frac{\boldsymbol{Q}(\tilde{\alpha}_\epsilon)\boldsymbol{P}(\tilde{\alpha}_\epsilon)}{\mathrm{Tr}[\boldsymbol{Q}(\tilde{\alpha}_\epsilon)\boldsymbol{P}(\tilde{\alpha}_\epsilon)]} \,. \tag{C.17}$$

We now detail the iterative algorithmic steps we use to reduce the spectral abscissa of $\boldsymbol{W}$:

1. Compute $\alpha(\boldsymbol{W})$ for the current weight matrix $\boldsymbol{W}$.

2. Set $s = \max(\alpha(\boldsymbol{W}) \times 1.5, \alpha(\boldsymbol{W}) + 0.2)$. This guarantees that $\tilde{\alpha}_\epsilon(\boldsymbol{W}) > \alpha(\boldsymbol{W})$. The values 1.5 and 0.2 are used because, from test simulations using all the parameter values that we specified above, these values have been found to be good choices (see the supplementary information in Hennequin et al. (2014) for further discussion).

3. Solve Eqns. (C.15) and (C.16) using a Lyapunov equation solver (e.g., LYAP in MATLAB) to obtain $\boldsymbol{Q}(s)$ and $\boldsymbol{P}(s)$.

4. Calculate the gradient with which to change the weights to reduce the smoothed spectral abscissa of $\boldsymbol{W}$:

$$\frac{\partial \tilde{\alpha}_\epsilon(\boldsymbol{W})}{\partial \boldsymbol{W}} = \frac{\boldsymbol{Q}(s)\boldsymbol{P}(s)}{\mathrm{Tr}[\boldsymbol{Q}(s)\boldsymbol{P}(s)]} \,. \tag{C.18}$$

5. For every inhibitory presynaptic neuron j and any postsynaptic neuron i, we change the inhibitory weight W_{ij} in the following way

$$W_{ij} \leftarrow W_{ij} - \eta \frac{\partial \tilde{\alpha}_\epsilon(\boldsymbol{W})}{\partial \boldsymbol{W}} \,, \tag{C.19}$$

   where η is a learning rate. We find that setting $\eta = 5$ normally allows efficient and smooth gradient descent.

6. Set any positive inhibitory weights to 0.

7. Enforce inhibition to be a mean of $\gamma = 3$ times stronger than excitation. This is achieved by rescaling the inhibitory 'blocks' $\boldsymbol{W}_{EI}$ and $\boldsymbol{W}_{II}$ by $-\gamma\overline{\boldsymbol{W}}_{EE}/\overline{\boldsymbol{W}}_{EI}$ and $-\gamma\overline{\boldsymbol{W}}_{IE}/\overline{\boldsymbol{W}}_{II}$, respectively, where

$$\boldsymbol{W} = \begin{pmatrix} \boldsymbol{W}_{EE} & \boldsymbol{W}_{EI} \\ \boldsymbol{W}_{IE} & \boldsymbol{W}_{II} \end{pmatrix} \,,$$

   and $\overline{\boldsymbol{W}}_{XY}$ denotes the mean over all elements in the matrix $\boldsymbol{W}_{XY}$. This step is not necessary for stability optimisation, but see the supplementary information of Hennequin et al. (2014) for a discussion.

8. Restrict the density of inhibitory connections to be less than or equal to 0.4. This is achieved by setting the 60 % smallest-magnitude inhibitory connection strengths to 0. (One also removes any self-loops — $\boldsymbol{W}_{ii} = 0$ for all i — because one can trivially shift the spectrum of $\boldsymbol{W}$ in the negative direction by addition of a diagonal matrix with only negative elements.) This approach of maintaining sparse inhibitory connectivity is different from that used in Hennequin et al. (2014). We used this approach for simplicity. This constraint is not required for stability optimisation, but adds to the biological realism of the resulting weight matrix while still allowing the spectral abscissa to decrease to a low value.

We repeat steps 1–8 until $\alpha(\boldsymbol{W}) < 0.15$. This is normally enough iterations for convergence of the spectral abscissa.

C.2 Finding preferred initial conditions

We wish to find initial conditions that produce large deviations in neuronal activity away from baseline (Hennequin et al., 2014). We define the so-called 'optimal' initial condition, as the (unit-norm) initial condition $\boldsymbol{a}_1$ that maximises the evoked energy $\varepsilon(\boldsymbol{a}_1)$ under the linear dynamics Eqn. (C.2), where

$$\varepsilon(\boldsymbol{a}_1) = \frac{2}{\tau}\int_0^\infty \|\boldsymbol{x}(t)\|_2^2 dt\,. \tag{C.20}$$

(Note that for a weight matrix $\boldsymbol{W}$ with $\alpha(\boldsymbol{W}) < 1$, the envelope of the linear dynamics (1.9) decays exponentially over time, thus ε is also finite.)

From the previous section, we know that Eqn. (C.20) can be re-written as

$$\varepsilon(\boldsymbol{a}) = \boldsymbol{a}^\mathsf{T}\boldsymbol{Q}\boldsymbol{a} \tag{C.21}$$

where $\boldsymbol{Q}$ is the observability Gramian matrix defined by Eqn. (C.3). Recall that $\boldsymbol{Q}$ is a symmetric, positive-definite matrix. As we already showed in Eqn. (C.13), the

eigenvector corresponding to the maximum eigenvalue of $\boldsymbol{Q}$ produces the maximum evoked energy ε under the linear dynamics Eqn. (C.2). In fact, the basis of $\boldsymbol{Q}$, when placed in decreasing order according to their associated eigenvalues, defines a collection $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_N$ of N orthogonal initial conditions that each maximise the evoked energy $\varepsilon(\boldsymbol{a}_k)$ within the subspace orthogonal to all previous preferred initial conditions $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{k-1}$. We also note that for the case of linear neuronal dynamics, $\varepsilon(\boldsymbol{a}_k) = \varepsilon(\boldsymbol{a}_{-k})$. Finally, $\boldsymbol{Q}$ can be easily found by solving Eqn. (C.11) using a Lyapunov equation solver such as LYAP in MATLAB.

Unless we state otherwise, we choose the eigenvector associated with the largest eigenvalue of $\boldsymbol{Q}$ (note that all of its eigenvalues are real and positive) as the initial condition $\boldsymbol{x}_0$ for the neuronal activity. Following Hennequin et al. (2014), we also scale $\boldsymbol{x}_0$ so that $\|\boldsymbol{x}_0\|_2 = 1.5\sqrt{N}$.

APPENDIX D

Measuring error in network output

We compute the error ϵ between the network output $\boldsymbol{z} \in \mathbb{R}^T$ and a target $\boldsymbol{y} \in \mathbb{R}^T$ by calculating

$$\epsilon = 1 - R^2 = \frac{\sum_{t=1}^{T}(z(t) - y(t))^2}{\sum_{t=1}^{T}(y(t) - \bar{y})^2}\,, \tag{D.1}$$

where $\bar{y} = \frac{1}{T}\sum_{t=1}^{T} y(t)$ and R^2 is the commonly used coefficient of determination (which is often called simply 'R-squared'). Therefore, an error of $\epsilon = 1$ implies that the performance is as bad as if the output $\boldsymbol{z}$ were equal to the mean of the target $\boldsymbol{y}$ and thus does not capture any variations in output. When we use multiple readout units, we take the mean error ϵ across all outputs. We use this definition of error throughout the entire thesis.

Bibliography

L. F. Abbott and S. B. Nelson. Synaptic plasticity: Taming the beast. *Nature Neuroscience*, 3:1178–1183, 2000. (Cited on page 5.)

A. Afshar, G. Santhanam, B. M. Yu, S. I. Ryu, M. Sahani, and K. V. Shenoy. Single-trial neural correlates of arm movement preparation. *Neuron*, 71(3):555–564, 2011. (Cited on pages 4 and 21.)

C. I. Bargmann. Beyond the connectome: How neuromodulators shape neural circuits. *BioEssays*, 34(6):458–465, 2012. (Cited on page 137.)

D. S. Bassett and O. Sporns. Network neuroscience. *Nature Neuroscience*, 20(3): 353–364, 2017. (Cited on page 137.)

E. Bizzi and V. C. K. Cheung. The neural origin of muscle synergies. *Frontiers in Computational Neuroscience*, 7(51):1–6, 2013. (Cited on pages 91, 99, and 100.)

T. V. P. Bliss and G. L. Collingridge. A synaptic model of memory: long-term potentiation in the hippocampus. *Nature*, 361(6407):31–39, 1993. (Cited on page 3.)

Y. L. Boureau and P. Dayan. Opponency revisited: Competition and cooperation between dopamine and serotonin. *Neuropsychopharmacology*, 36(1):74–97, 2011. (Cited on page 62.)

V. Braitenberg and A. Schüz. *Anatomy of the Cortex*. Springer, 1991. (Cited on page 14.)

M. Breakspear. Dynamic models of large-scale brain activity. *Nature Neuroscience*, 20(3):340–352, 2017. (Cited on pages 9 and 136.)

J. M. Carmena, M. A. Lebedev, R. E. Crist, J. E. O'Doherty, D. M. Santucci, D. F. Dimitrov, P. G. Patil, C. S. Henriquez, and M. A. L. Nicolelis. Learning to control a brain-machine interface for reaching and grasping by primates. *PLoS Biology*, 1(2):e42, 2003. (Cited on page 101.)

F. S. Chance, L. F. Abbott, and A. D. Reyes. Gain modulation from background synaptic input. *Neuron*, 35(4):773–782, 2002. (Cited on pages 12, 24, 25, 86, 123, 125, and 135.)

M. M. Churchland and J. P. Cunningham. A dynamical basis set for generating reaches. *Cold Spring Harbor Symposia on Quantitative Biology*, 79:67–80, 2014. (Cited on pages 4, 21, and 22.)

M. M. Churchland, J. P. Cunningham, M. T. Kaufman, S. I. Ryu, and K. V. Shenoy. Cortical preparatory activity: Representation of movement or first cog in a dynamical machine? *Neuron*, 68(3):387–400, 2010. (Cited on pages 4, 21, 44, 54, 67, 134, and 136.)

M. M. Churchland, J. P. Cunningham, M. T. Kaufman, J. D. Foster, P. Nuyujukian, S. I. Ryu, and K. V. Shenoy. Neural population dynamics during reaching. *Nature*, 487(7405):1–8, 2012. (Cited on pages 4, 17, 21, 23, 29, 39, 44, 54, 56, 57, 69, 72, 110, 111, 124, 134, and 136.)

G. L. Collier and C. E. Wright. Temporal rescaling of simple and complex ratios in rhythmic tapping. *Journal of Experimental Psychology: Human Perception and Performance*, 21(3):602–627, 1995. (Cited on page 117.)

R. Cools, K. Nakamura, and N. D. Daw. Serotonin and dopamine: Unifying affective, activational, and decision functions. *Neuropsychopharmacology*, 36(1): 98–113, 2011. (Cited on page 62.)

J. P. Coxon, J. W. Stinear, and W. D. Byblow. Amplitude of muscle stretch modulates corticomotor gain during passive movement. *Brain Research*, 1031(1): 109–117, 2005. (Cited on page 28.)

D. A. Crowe, W. Zarco, R. Bartolo, and H. Merchant. Dynamic representation of the temporal and sequential structure of rhythmic movements in the primate medial premotor cortex. *The Journal of Neuroscience*, 34(36):11972–11983, 2014. (Cited on page 123.)

S. J. Cruikshank and N. M. Weinberger. Evidence for the Hebbian hypothesis in experience-dependent physiological plasticity of neocortex: a critical review. *Brain Research Reviews*, 22(3):191–228, 1996. (Cited on page 3.)

P. Damier, E. C. Hirsch, Y. Agid, and A. M. Graybiel. The substantia nigra of the human brain: II. Patterns of loss of dopamine-containing neurons in Parkinson's disease. *Brain*, 122(8):1437–1448, 1999. (Cited on pages 43 and 45.)

P. Dayan and L. Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. MIT Press, 2001. (Cited on pages 4, 5, 6, 7, 8, 13, and 14.)

M. de Bono and A. Villu Maricq. Neuronal substrates of complex behaviors in C. elegans. *Annual Review of Neuroscience*, 28(1):451–501, 2005. (Cited on page 1.)

B. DePasquale, M. M. Churchland, and L. F. Abbott. Using firing-rate dynamics to train recurrent networks of spiking model neurons. *arXiv:1601.07620*, 2016. (Cited on page 137.)

B. DePasquale, C. J. Cueva, K. Rajan, G. S. Escola, and L. F. Abbott. full-FORCE: A target-based method for training recurrent networks. *PLoS One*, 13 (2):e0191527, 2018. (Cited on page 137.)

P. Dorato, C. T. Abdallah, V. Cerone, and D. H. Jacobson. *Linear–Quadratic Control: An Introduction*. Prentice Hall, 1998. (Cited on page 142.)

K. Doya. Metalearning and neuromodulation. *Neural Networks*, 15(4–6):495–506, 2002. (Cited on page 62.)

E. Eldar, J. D. Cohen, and Y. Niv. The effects of neural gain on attention and learning. *Nature Neuroscience*, 16(8):1146–1153, 2013. (Cited on page 26.)

T. Flash and B. Hochner. Motor primitives in vertebrates and invertebrates. *Current Opinion in Neurobiology*, 15(6):660–666, 2005. (Cited on pages 91 and 100.)

N. Frémaux and W. Gerstner. Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules. *Frontiers in Neural Circuits*, 9:85, 2016. (Cited on pages 62, 63, and 136.)

K. J. Friston. Functional and effective connectivity: A review. *Brain Connectivity*, 1 (1):13–36, 2011. (Cited on page 134.)

J. A. Gallego, M. G. Perich, L. E. Miller, and S. A. Solla. Neural manifolds for the control of movement. *Neuron*, 94(5):978–984, 2017. (Cited on pages 4 and 119.)

W. Gerstner and W. M. Kistler. *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002. (Cited on pages 4 and 136.)

W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski. *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press, 2014. (Cited on pages 4, 5, 6, 8, and 14.)

A. Gilra and W. Gerstner. Predicting nonlinear dynamics by stable local learning in a recurrent spiking neural network. *eLife*, 6:e28295, 2017. (Cited on page 137.)

V. L. Girko. Circular law. *Theory of Probability and its Applications*, 29(4):694–706, 1983. (Cited on pages 14 and 142.)

S. F. Giszter. Motor primitives — new data and future questions. *Current Opinion in Neurobiology*, 33:156–165, 2015. (Cited on pages 91, 99, and 100.)

M. D. Golub, P. T. Sadtler, E. R. Oby, K. M. Quick, S. I. Ryu, E. C. Tyler-kabara, A. P. Batista, S. M. Chase, and B. M. Yu. Learning by neural reassociation. *Nature Neuroscience*, 21(4):607–616, 2018. (Cited on pages 54, 90, 99, and 101.)

Q. Gu. Neuromodulatory transmitter systems in the cortex and their role in cortical plasticity. *Neuroscience*, 111(4):815–835, 2002. (Cited on page 136.)

N. F. Hardy and D. V. Buonomano. Encoding time in feedforward trajectories of a recurrent neural network model. *Neural Computation*, 30(2):378–396, 2018. (Cited on page 108.)

N. F. Hardy, V. Goudar, J. L. Romero-Sosa, and D. V. Buonomano. A model of temporal scaling correctly predicts that Weber's law is speed-dependent. *bioRxiv:159590*, 2017. (Cited on pages 108, 117, 124, and 126.)

J. Hass and D. Durstewitz. Time at the center, or time at the side? Assessing current models of time perception. *Current Opinion in Behavioral Sciences*, 8: 238–244, 2016. (Cited on page 124.)

D. O. Hebb. *The Organization of Behavior*. New York: Wiley, 1949. (Cited on page 3.)

G. Hennequin. *Stability and Amplification in Plastic Cortical Circuits*. PhD thesis, Ecole Polytechnique Federale de Lausanne, 2013. (Cited on pages 16 and 31.)

G. Hennequin, T. P. Vogels, and W. Gerstner. Non-normal amplification in random balanced neuronal networks. *Physical Review E*, 86(1):e011909, 2012. (Cited on pages 14 and 16.)

G. Hennequin, T. P. Vogels, and W. Gerstner. Optimal control of transient dynamics in balanced networks supports generation of complex movements. *Neuron*, 82 (6):1394–1406, 2014. (Cited on pages 9, 11, 12, 14, 15, 17, 18, 19, 21, 22, 27, 28, 29, 30, 36, 37, 39, 40, 56, 57, 58, 67, 72, 77, 92, 110, 113, 124, 126, 134, 137, 141, 142, 143, 145, 146, and 147.)

G. Hennequin, Y. Ahmadian, D. B. Rubin, M. Lengyel, and K. D. Miller. The dynamical regime of sensory cortex: Stable dynamics around a single stimulus-tuned attractor account for patterns of noise variability. *Neuron*, 98(4):846–860, 2018. (Cited on pages 45 and 135.)

S. Hernandez-Lopez, T. Tkatch, E. Perez-Garci, E. Galarraga, J. Bargas, H. Hamm, and D. J. Surmeier. D2 dopamine receptors in striatal medium spiny neurons reduce L-type Ca2+ currents and excitability via a novel PLC[beta]1-IP3-calcineurin-signaling cascade. *The Journal of Neuroscience*, 20(24):8987–8995, 2000. (Cited on pages 24, 26, 31, 43, 45, 62, 76, 100, 108, 125, and 135.)

N. J. Higham. The scaling and squaring method for the matrix exponential revisited. *SIAM Journal on Matrix Analysis and Applications*, 26(4):1179–1193, 2005. (Cited on page 10.)

G. M. Hoerzer, R. Legenstein, and W. Maass. Emergence of complex computational structures from chaotic neural networks through reward-modulated Hebbian learning. *Cerebral Cortex*, 24(3):677–690, 2014. (Cited on pages 12, 13, 39, 54, 58, 62, 63, 69, and 135.)

J. A. Hosp, A. Pekanovic, M. S. Rioult-Pedotti, and A. R. Luft. Dopaminergic projections from midbrain to primary motor cortex mediate motor skill learning. *Jour-*

nal of Neuroscience, 31(7):2481–2487, 2011. (Cited on pages 54, 62, 63, 76, and 86.)

G. W. Huntley, J. H. Morrison, A. Prikhozhan, and S. C. Sealfon. Localization of multiple dopamine receptor subtype mRNAs in human and monkey motor cortex and striatum. *Molecular Brain Research*, 15(3-4):181–188, 1992. (Cited on pages 63, 76, and 86.)

V. Kaasinen, J. Joutsa, T. Noponen, and M. Päivärinta. Akinetic crisis in Parkinson's disease is associated with a severe loss of striatal Dopamine transporter function: A report of two cases. *Case Reports in Neurology*, 6(3):275–280, 2014. (Cited on page 126.)

H. Kambara, D. Shin, and Y. Koike. A computational model for optimal muscle activity considering muscle viscoelasticity in wrist movements. *Journal of Neurophysiology*, 109(8):2145–2160, 2013. (Cited on page 73.)

J. C. Kao, P. Nuyujukian, S. I. Ryu, M. M. Churchland, J. P. Cunningham, and K. V. Shenoy. Single-trial dynamics of motor cortex and their applications to brain-machine interfaces. *Nature Communications*, 6:7759, 2015. (Cited on pages 4, 29, 56, 69, and 136.)

H. Kida and D. Mitsushima. Mechanisms of motor learning mediated by synaptic plasticity in rat primary motor cortex. *Neuroscience Research*, 128:14–18, 2018. (Cited on pages 23, 54, 55, 62, and 137.)

T. Komiyama, T. R. Sato, D. H. Oconnor, Y. X. Zhang, D. Huber, B. M. Hooks, M. Gabitto, and K. Svoboda. Learning-related fine-scale specificity imaged in motor cortex circuits of behaving mice. *Nature*, 464(7292):1182–1186, 2010. (Cited on page 23.)

R. Laje and D. V. Buonomano. Robust timing and motor patterns by taming chaos

in recurrent neural networks. *Nature Neuroscience*, 16(7):925–933, 2013. (Cited on pages 108 and 124.)

A. H. Lara, J. P. Cunningham, and M. M. Churchland. Different population dynamics in the supplementary motor area and motor cortex during reaching. *Nature Communications*, 9:2754, 2018. (Cited on pages 22, 29, 56, and 69.)

S. Lefort, C. Tomm, J. C. Floyd Sarria, and C. C. Petersen. The excitatory neuronal network of the C2 barrel column in mouse primary somatosensory cortex. *Neuron*, 61(2):301–316, 2009. (Cited on page 14.)

R. Legenstein, S. M. Chase, A. B. Schwartz, and W. Maass. A reward-modulated hebbian learning rule can explain experimentally observed network reorganization in a brain control task. *Journal of Neuroscience*, 30(25):8400–8410, 2010. (Cited on pages 54, 62, and 63.)

H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. Visualizing the loss landscape of neural nets. *arXiv:1712.09913*, 2018. (Cited on page 87.)

N. Li, T.-W. Chen, Z. V. Guo, C. R. Gerfen, and K. Svoboda. A motor cortex circuit for motor planning and movement. *Nature*, 519(7541):51, 2015. (Cited on page 67.)

A. R. Luft and S. Schwarz. Dopaminergic signals in primary motor cortex. *International Journal of Developmental Neuroscience*, 27(5):415–421, 2009. (Cited on pages 62 and 136.)

V. Mante, D. Sussillo, K. V. Shenoy, and W. T. Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, 503(7474):78–84, 2013. (Cited on pages 9 and 136.)

E. Marder. Neuromodulation of neuronal circuits: Back to the future. *Neuron*, 76 (1):1–11, 2012. (Cited on pages 136 and 137.)

C. D. Marsden. Slowness of movement in Parkinson's disease. *Movement Disorders*, 4(1):S26–S37, 1989. (Cited on page 126.)

A. R. O. Martins and R. C. Froemke. Coordinated forms of noradrenergic plasticity in the locus coeruleus and primary auditory cortex. *Nature Neuroscience*, 18 (10):1483–1492, 2015. (Cited on page 136.)

P. Mazzoni, R. A. Andersen, and M. I. Jordan. A more biologically plausible learning rule for neural networks. *Proceedings of the National Academy of Sciences*, 88 (10):4433–4437, 1991. (Cited on page 62.)

T. Miconi. Biologically plausible learning in recurrent neural networks for flexible decision tasks. *eLife*, 6:e20899, 2017. (Cited on pages 54, 62, 63, 69, and 73.)

K. D. Miller and F. Fumarola. Mathematical equivalence of two common forms of firing rate models of neural networks. *Neural Computation*, 24(1):25–31, 2012. (Cited on page 5.)

K. Molina-Luna, A. Pekanovic, S. Rohrich, B. Hertler, M. Schubring-Giese, M. S. Rioult-Pedotti, and A. R. Luft. Dopamine in motor cortex is necessary for skill learning and synaptic plasticity. *PloS One*, 4(9):e7082, 2009. (Cited on pages 54, 62, 63, 76, 86, and 100.)

J. M. Montgomery and D. V. Madison. Discrete synaptic states define a major mechanism of synapse plasticity. *Trends in Neurosciences*, 27(12):744–750, 2004. (Cited on page 5.)

B. K. Murphy and K. D. Miller. Balanced amplification: A new mechanism of selective amplification of neural activity patterns. *Neuron*, 61(4):635–648, 2009. (Cited on pages 15 and 16.)

P. R. Murphy, E. Boonstra, and S. Nieuwenhuis. Global gain modulation generates

time-dependent urgency during perceptual choice in humans. *Nature Communications*, 7:13526, 2016. (Cited on pages 26 and 135.)

W. Nicola and C. Clopath. Supervised learning in spiking neural networks with FORCE training. *Nature Communications*, 8(1):2208, 2017. (Cited on page 137.)

Y. Niv, N. D. Daw, D. Joel, and P. Dayan. Tonic dopamine: Opportunity costs and the control of response vigor. *Psychopharmacology*, 191(3):507–520, 2007. (Cited on pages 43, 45, and 125.)

B. Panigrahi, K. A. Martin, Y. Li, A. R. Graves, A. Vollmer, L. Olson, B. D. Mensh, A. Y. Karpova, and J. T. Dudman. Dopamine is required for the neural representation and control of movement vigor. *Cell*, 162(6):1418–1430, 2015. (Cited on pages 43, 45, and 125.)

T. Pereira. Stability of synchronized motion in complex networks. *arXiv:1112.2297*, 2011. (Cited on pages 30 and 47.)

M. G. Perich, J. A. Gallego, and L. E. Miller. A neural population mechanism for rapid learning. *bioRxiv:138743*, 2017. (Cited on pages 23 and 54.)

A. J. Peters, S. X. Chen, and T. Komiyama. Emergence of reproducible spatiotemporal activity during motor learning. *Nature*, 510(7504):263–267, 2014. (Cited on pages 23 and 54.)

K. Rajan and L. F. Abbott. Eigenvalue spectra of random matrices for neural networks. *Physical Review Letters*, 97(18):2–5, 2006. (Cited on pages 14 and 15.)

K. Rajan, L. F. Abbott, and H. Sompolinsky. Stimulus-dependent suppression of chaos in recurrent neural networks. *Physical Review E*, 82(1):011903, 2010. (Cited on pages 12, 13, 14, 28, 56, and 110.)

K. Rajan, C. D. Harvey, and D. W. Tank. Recurrent network models of sequence generation and memory. *Neuron*, 90(1):128–142, 2016. (Cited on page 108.)

A. Rakitianskaia, E. Bekker, K. M. Malan, and A. Engelbrecht. Analysis of error landscapes in multi-layered neural networks for classification. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 5270–5277, 2016. (Cited on page 87.)

J.-A. Rathelot and P. L. Strick. Subdivisions of primary motor cortex based on cortico-motoneuronal cells. *Proceedings of the National Academy of Sciences*, 106(3):918–923, 2009. (Cited on page 39.)

R. L. Redondo and R. G. M. Morris. Making memories last: The synaptic tagging and capture hypothesis, 2011. (Cited on page 5.)

E. D. Remington, D. Narain, E. A. Hosseini, and M. Jazayeri. Flexible sensorimotor computations through rapid reconfiguration of cortical dynamics. *Neuron*, 98(5): 1005–1019, 2018. (Cited on pages 108, 123, 124, 125, and 126.)

D. A. Rosenbaum. *Human motor control*. Academic press, 2009. (Cited on page 54.)

J. S. Rothman, L. Cathala, V. Steuber, and R. A. Silver. Synaptic depression enables neuronal gain control. *Nature*, 457(7232):1015–1018, 2009. (Cited on pages 12 and 26.)

D. B. Rubin, S. D. Van Hooser, and K. D. Miller. The stabilized supralinear network: A unifying circuit motif underlying multi-input integration in sensory cortex. *Neuron*, 85(2):402–417, 2015. (Cited on page 45.)

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. (Cited on pages 74 and 115.)

A. A. Russo, S. R. Bittner, S. M. Perkins, J. S. Seely, B. M. London, A. H. Lara, A. Miri, N. J. Marshall, A. Kohn, T. M. Jessell, L. F. Abbott, J. P. Cunningham, and

M. M. Churchland. Motor cortex embeds muscle-like commands in an untangled population response. *Neuron*, 97(4):1–14, 2018. (Cited on pages 4, 39, 57, 58, 72, and 123.)

P. T. Sadtler, K. M. Quick, M. D. Golub, S. M. Chase, S. I. Ryu, E. C. Tyler-Kabara, B. M. Yu, and A. P. Batista. Neural constraints on learning. *Nature*, 512(7515): 423–426, 2014. (Cited on pages 23, 54, 90, 99, and 101.)

H. Saito, K. Katahira, K. Okanoya, and M. Okada. Statistical mechanics of structural and temporal credit assignment effects on learning in neural networks. *Physical Review E*, 83(5), 2011. (Cited on page 63.)

E. Salinas and N. M. Bentley. *Gain modulation as a mechanism for switching reference frames, tasks, and targets*, pages 121–142. Springer New York, New York, NY, 3 edition, 2009. (Cited on page 26.)

E. Salinas and T. J. Sejnowski. Gain modulation in the central nervous system: Where behavior, neurophysiology, and computation meet. *Neuroscientist*, 7(5): 430–440, 2001. (Cited on pages 24, 27, and 55.)

E. Salinas and P. Thier. Gain modulation: A major computational principle of the central nervous system. *Neuron*, 27(1):15–21, 2000. (Cited on pages 24, 28, and 55.)

J. N. Sanes and J. P. Donoghue. Plasticity and primary motor cortex. *Annual Review of Neuroscience*, 23(1):393–415, 2000. (Cited on pages 23 and 54.)

E. S. Schaffer, S. Ostojic, and L. F. Abbott. A complex-valued firing-rate model that approximates the dynamics of spiking networks. *PLOS Computational Biology*, 9(10):e1003301, 2013. (Cited on page 137.)

K. V. Shenoy, M. Sahani, and M. M. Churchland. Cortical control of arm move-

ments: a dynamical systems perspective. *Annual Review of Neuroscience*, 36: 337–359, 2013. (Cited on pages 4, 21, 23, 44, 54, 67, 134, and 136.)

S. Soares, B. V. Atallah, and J. J. Paton. Midbrain dopamine neurons control judgment of time. *Science*, 354(6317):1273–1277, 2016. (Cited on pages 108 and 123.)

H. Sompolinsky, A. Crisanti, and H. J. Sommers. Chaos in random neural networks. *Physical Review Letters*, 61(3):259–262, 1988. (Cited on pages 12, 13, 14, 30, 32, and 67.)

D. A. Spampinato, H. J. Block, and P. A. Celnik. Cerebellar-M1 connectivity changes associated with motor learning are somatotopic specific. *Journal of Neuroscience*, 37(9):2377–2386, 2017. (Cited on page 100.)

G. Strang. *Linear Algebra and its Applications*. Wellesley Cambridge Press, fifth edition, 2016. (Cited on pages 4, 10, and 139.)

P. Strata and R. Harvey. Dale's principle. *Brain Research Bulletin*, 50(5):349–350, 1999. (Cited on pages 6 and 14.)

S. H. Strogatz. *Nonlinear Dynamics and Chaos*. Westview Press, second edition, 2014. (Cited on page 4.)

D. Sussillo and L. F. Abbott. Generating coherent patterns of activity from chaotic neural networks. *Neuron*, 63(4):544–557, 2009. (Cited on pages 13, 14, 39, 54, 58, 67, 69, 134, and 135.)

D. Sussillo and O. Barak. Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Computation*, 25(3):626–649, 2013. (Cited on page 135.)

D. Sussillo, M. M. Churchland, M. T. Kaufman, and K. V. Shenoy. A neural network that finds a naturalistic solution for the production of muscle activity. *Nature*

Neuroscience, 18(7):1025–1033, 2015. (Cited on pages 9, 23, 39, 54, 57, 58, 67, 72, and 134.)

C. D. Swinehart and L. F. Abbott. Supervised learning through neuronal response modulation. *Neural Computation*, 17(3):609–631, 2005. (Cited on page 136.)

C. D. Swinehart, K. Bouchard, P. Partensky, and L. F. Abbott. Control of network activity through neuronal response modulation. *Neurocomputing*, 58:327–335, 2004. (Cited on pages 24, 27, and 55.)

G. Teschl. *Ordinary Differential Equations and Dynamical Systems*. American Mathematical Society, 2012. (Cited on pages 4, 10, 11, 12, 30, 35, 47, 50, and 140.)

K. A. Thoroughman and R. Shadmehr. Learning of action through adaptive combination of motor primitives. *Nature*, 407(6805):742–747, 2000. (Cited on pages 91 and 100.)

K. Thurley, W. Senn, and H.-R. Lüscher. Dopamine increases the gain of the input-output response of rat prefrontal pyramidal neurons. *Journal of Neurophysiology*, 99(6):2985–2997, 2008. (Cited on pages 11, 12, 24, 26, 31, 43, 45, 62, 76, 86, 100, 108, 123, 126, 135, and 137.)

L. N. Trefethen and M. Embree. *Spectra and Pseudospectra: The Behavior of Nonnormal Matrices and Operators*. Princeton University Press, 2005. (Cited on page 16.)

J. Vanbiervliet, B. Vandereycken, W. Michiels, S. Vandewalle, and M. Diehl. The smoothed spectral abscissa for robust stability optimization. *SIAM Journal on Optimization*, 20(1):156–171, 2009. (Cited on pages 11, 18, 142, and 144.)

M. Vestergaard and R. W. Berg. Divisive gain modulation of motoneurons by in-

hibition optimizes muscular control. *Journal of Neuroscience*, 35(8):3711–3723, 2015. (Cited on pages 23, 26, 27, 28, 39, 45, 55, 62, 112, and 115.)

T. P. Vogels, K. Rajan, and L. Abbott. Neural network dynamics. *Annual Review of Neuroscience*, 28:357–376, 2005. (Cited on pages 13 and 14.)

T. P. Vogels, H. Sprekeler, F. Zenke, C. Clopath, and W. Gerstner. Inhibitory plasticity balances excitation and inhibition in sensory pathways and memory networks. *Science*, 334(6062):1569–1573, 2011. (Cited on page 31.)

J. Wang, D. Narain, E. A. Hosseini, and M. Jazayeri. Flexible timing by temporal scaling of cortical responses. *Nature Neuroscience*, 21(1):102–110, 2018. (Cited on pages 9, 24, 39, 58, 108, 123, 124, 126, 135, and 136.)

K. Wei, J. I. Glaser, L. Deng, C. K. Thompson, I. H. Stevenson, Q. Wang, T. G. Hornby, C. J. Heckman, and K. P. Kording. Serotonin affects movement gain control in the spinal cord. *Journal of Neuroscience*, 34(38):12690–12700, 2014. (Cited on pages 23, 24, 26, 27, 28, 31, 45, 55, 62, 76, 100, and 135.)

J. Werfel, X. Xie, and H. S. Seung. Learning Curves for Stochastic Gradient Descent in Linear Feedforward Networks. *Neural Computation*, 17(12):2699–2718, 2005. (Cited on page 79.)

J. G. White, E. Southgate, J. N. Thomson, and S. Brenner. The structure of the nervous system of Caenorhabditis elegans. *Philos. Trans. R. Soc. Lond.*, 314 (1165):1–340, 1986. (Cited on page 2.)

S. Wiggins. *Introduction to Applied Nonlinear Dynamical Systems and Chaos*. Springer, second edition, 2003. (Cited on pages 4 and 30.)

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992. (Cited on page 62.)

H. R. Wilson and J. D. Cowan. Excitatory and inhibitory interactions in localized populations of model neurons. *Biophysical Journal*, 12(1):1–24, 1972. (Cited on page 5.)

D. M. Wolpert. The real reason for brains [Video file]. *Retrieved from www.ted.com/talks/daniel_wolpert_the_real_reason_for_brains#t-69912*, 2011. (Cited on page 2.)

D. M. Wolpert and J. Flanagan. Motor prediction. *Current Biology*, 11(18):729–732, 2001. (Cited on page 90.)

D. M. Wolpert, R. C. Miall, and M. Kawato. Internal models in the cerebellum. *Trends in Cognitive Sciences*, 2(9):338–347, 1998. (Cited on pages 90, 99, and 100.)

T. Xu, X. Yu, A. J. Perlik, W. F. Tobin, J. A. Zweig, K. Tennant, T. Jones, and Y. Zuo. Rapid formation and selective stabilization of synapses for enduring motor memories. *Nature*, 462(7275):915–919, 2009. (Cited on page 23.)

J. Zhang and L. F. Abbott. Gain modulation of recurrent networks. *Neurocomputing*, 32:623–628, 2000. (Cited on pages 24, 27, and 55.)

L. Ziegler, F. Zenke, D. B. Kastner, and W. Gerstner. Synaptic consolidation: From synapses to behavioral modeling. *Journal of Neuroscience*, 35(3):1319–1334, 2015. (Cited on page 5.)