Neural Networks 150 (2022) 237–258

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

A differential Hebbian framework for biologically-plausible motor control

Sergio Verduzco-Flores *, William Dorrell, Erik De Schutter

Computational Neuroscience Unit, Okinawa Institute of Science and Technology, Okinawa, Japan

ARTICLE INFO

Article history:
Received 21 April 2021
Received in revised form 15 January 2022
Accepted 3 March 2022
Available online 10 March 2022

Keywords:
Synaptic plasticity
Motor control
Reinforcement learning
Feedback control

ABSTRACT

In this paper we explore a neural control architecture that is both biologically plausible, and capable of fully autonomous learning. It consists of feedback controllers that learn to achieve a desired state by selecting the errors that should drive them. This selection happens through a family of differential Hebbian learning rules that, through interaction with the environment, can learn to control systems where the error responds monotonically to the control signal. We next show that in a more general case, neural reinforcement learning can be coupled with a feedback controller to reduce errors that arise non-monotonically from the control signal. The use of feedback control can reduce the complexity of the reinforcement learning problem, because only a desired value must be learned, with the controller handling the details of how it is reached. This makes the function to be learned simpler, potentially allowing learning of more complex actions. We use simple examples to illustrate our approach, and discuss how it could be extended to hierarchical architectures.

1. Introduction

Understanding animal motor control holds the promise of improving therapies for people with motor deficits. Moreover, complex motor control in animals remains superior to current artificial systems, so insights from animal motor control may one day improve state-of-the-art artificial control. To reach such understanding, we need models that obey strong biological plausibility constraints, but still perform increasingly complex motor tasks.

We believe that serious attempts at biological plausibility should consider the following points:

- Modeling the full sensorimotor loop with a controller that only uses neurons. Learning consists of adjusting the weights of their synaptic connections.
- Learning rules use only information locally available at the postsynaptic neuron.
- The agent learns as its body interacts in real time with the environment. Rather than relying on labeled data, learning takes advantage of correlation between signals, and reinforcement learning mechanisms.
- Transmission delays and response latencies should be considered.
- No element of the model goes against current consensus in neuroscience.

We are not aware of motor control models that follow all these guidelines, and only a few follow most of them. This is because complications arise in biological models. The worst complication may be the one recently dubbed as the *supraspinal pattern formation problem* (Bizzi & Ajemian, 2020): how are the spinal cord components coordinated in time to generate goal-directed movements? A closely related complication is that many motor patterns may achieve the same motor outcome. This was originally known as the *DOF problem* (Bernstein, 1967), or more commonly as the *redundancy problem*.

In this paper we lay a framework for motor control that incorporates all the biological constraints above, while offering a viable solution to supraspinal pattern formation and redundancy. The key is to cast the problem in terms of finding the input–output structure of a Multiple-Input Multiple-Output (MIMO) feedback control system (Seborg, Edgar, Mellichamp, & III, 2016, Ch.18), or in other terms, solving the input–output decoupling problem (Nijmeijer & van der Schaft, 1990). This problem is about choosing the right actuator (controller output) in order to reduce the error for each controlled variable (controller input). Its main complication is that the actuators may affect several controlled variables, so using one of them to control a variable may cause unwanted interference in the state of other variables. In engineering systems this is usually addressed during the design stage, but at least in primates this is likely learned through experience.

The approach we use to find the input–output structure in MIMO feedback control relies on learning *sensitivity derivatives* using differential Hebbian learning with synaptic competition. The sensitivity derivatives are the values de_i/dc_j, where $\mathbf{c}$ =

* Corresponding author.
E-mail address: sergio.verduzco@gmail.com (S. Verduzco-Flores).

https://doi.org/10.1016/j.neunet.2022.03.002

$[c_1, \ldots, c_N]$ is the output vector produced by the controller in order to reduce an error vector $\mathbf{e} = [e_1, \ldots, e_M]$.

We will find that this can be an effective solution, but that it fails in cases where the relation between input and output changes for different contexts. To handle this scenario we will combine our feedback controller with a variant of the actor–critic architecture, which will allow it to self-configure for handling different contexts.

Most animal motor control models use a fixed input–output structure (e.g. Hayashibe & Shimoda, 2014; Kawato & Gomi, 1992; Porrill, Dean, & Anderson, 2013; Todorov, 2000). When asking how are motor errors defined and used, they assume that this is either genetically determined, or adjusted through an internal model. There is extensive evidence for the presence of internal forward models predicting the consequences of motor commands, and that they adapt when those consequences change due to perturbations (e.g. McNamee & Wolpert, 2019; Miall & Wolpert, 1996; Tanaka, Ishikawa, Lee, & Kakei, 2020). It is thus often assumed that motor corrections arising from errors are caused by a correction to a forward model (Jordan & Rumelhart, 1992; Wolpert, Ghahramani, & Jordan, 1995). An alternative that is not often considered is that the motor corrections are independent from the corrections to the forward models. Recent experiments suggest that this may be the case: errors in the sensory domain seem to generate motor corrections without using forward models (Hadjiosif, Krakauer, & Haith, 2021).

Sensitivity derivatives constitute a linear forward model, not of the system being controlled, but of the errors, which contain information about the desired outcome. As will be shown later, estimating a form of these values will directly produce error corrections, and adjust the control structure of the system. In contrast, approaches using internal models of the system being controlled (called the *plant*) need to train such models, and make them produce corrections; this usually requires a pre-existing control structure (e.g. Miyamoto, Kawato, Setoyama, & Suzuki, 1988; Porrill, Dean, & Stone, 2004), or a form of error backpropagation (Jordan & Rumelhart, 1992).

In addition of not depending on a forward model, the model we present consists entirely of neurons. Four control architectures using biologically-plausible neural networks are well known (Rokni, 2009), each presenting its own strengths and limitations. Direct inverse learning (Kuperstein, 1988) uses the correlations between muscle outputs and afferent inputs in order to approximate an inverse function that maps from desired afferent inputs to the muscle activity that produces them. A major drawback is that the relation between muscle activity and afferent inputs may not be invertible (e.g. many muscle activities producing the same results).

Distal supervised learning (Jordan & Rumelhart, 1992) is another neural network architecture for control. It relies on both forward and inverse models of the plant. In order to produce learning signals for the inverse model, the errors in the forward model must be backpropagated. Feedback error learning (Miyamoto et al., 1988) also uses an inverse model of the plant, but instead of relying on a forward model, it uses the error of a closed-loop feedback controller to train it. This avoids the need of a forward model as in distal supervised learning, but it relies on a pre-existing closed-loop controller.

The fourth architecture is Reinforcement Learning (RL), which avoids the limitations of the other architectures, but is generally slower to find a solution. Given the close ties between RL and differential Hebbian learning (Kolodziejski, Porr, & Wörgötter, 2008b), it is interesting to ask whether the correlations between inputs and outputs to the controller can be used to obtain a control law that is adaptive and biologically plausible. As far as we know this has not been attempted in order to obtain the sensitivity derivatives in closed-loop control (cf. Kolodziejski, Porr, & Wörgötter, 2008a).

We are aware of one single work concerned with finding sensitivity derivatives in a biologically plausible manner. In Abdelghani, Lillicrap, and Tweed (2008) the sensitivity derivatives are represented as the firing rates in a separate network doing expansive recoding of appropriate context variables, together with a variant of the LMS learning rule. The authors in this work were unable to represent the sensitivity derivatives without using fast weight transport (which is biologically implausible), so they had to represent them as firing rates. The approach that we will present below is capable of using synaptic weights to represent something analogous to the sensitivity derivatives. This permits memory of the learned variables. Moreover, we show that in a feedback architecture many learning rules can achieve this, with approaches within and outside of the RL framework.

There are in fact four models presented in this paper. In the Methods we first show a heuristic derivation of the differential Hebbian learning rules, and then describe each of the four models.

The learning rules we derive allow a proportional feedback control system to adjust so as to reduce an arbitrary error, as long as the error and the motor commands have a monotonic relation. In other words, the motor command should not cause the error to increase in one context, and to decrease in a different one.

The first model we present is a direct application of these learning rules to find the input–output structure of high-dimensional linear plants with varying levels of redundancy in the actuators. From this we will observe that the tolerance to redundancy is on par with some offline analytical approaches.

The second model uses one of our learning rules to control the angle of a pendulum. The pendulum is a 2-dimensional plant, so finding the input–output structure of a controller is not particularly hard. On the other hand, even if our feedback controller has the right input–output structure, it only provides proportional control, which is insufficient to deal with the pendulum's momentum. We thus modify the architecture of the feedback controller to incorporate velocity in the error through the *input correlation* learning rule (Porr & Wörgötter, 2006), resulting in a biologically-plausible, self-configuring proportional-derivative controller.

The third model illustrates a way that the limitation of monotonic errors mentioned above may be overcome. We again control a pendulum, but the signal that represents its angle has a discontinuity as the pendulum completes a full revolution, something that negative feedback control cannot compensate by changing its input–output structure. We thus enhance the controller with a *critic* component that indicates which angle representation to use for each context.

The RL methods we use in the third model are fairly standard: a neural implementation of TD-learning (Schultz, Dayan, & Montague, 1997), and reward-modulated Hebbian learning. However, the times at which the reward-modulated Hebbian rule updates are non-standard. The fourth model in this paper is meant to show that this is not arbitrary, as it can be useful in solving temporal credit assignment problems. To this end, in the fourth model a very simple controller uses reward-modulated Hebbian learning to solve the inverted pendulum problem.

The Results section illustrates the performance of the four models described in the Methods.

All models in this paper are meant to illustrate and provide proof-of-concept for the ideas in our approach to motor control. Application to the control of a more realistic biological system is presented in a subsequent paper Verduzco-Flores and De Schutter (2021).

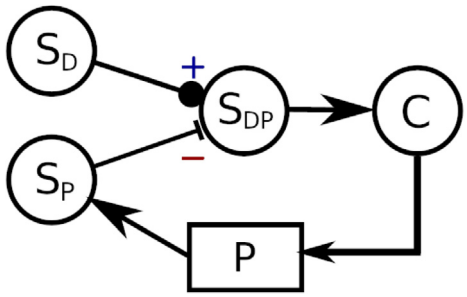

Fig. 1. A negative feedback controller. The circles represent populations of neural units whose output is a scalar value between 0 and 1 (e.g. firing rate neurons). Excitatory connections end with a closed circle, inhibitory connections with a bar. Connections with arrows can have inhibitory and excitatory components.

2. Methods

Simulations for all models were implemented in the Dracula neural simulator (Verduzco-Flores & De Schutter, 2019). The values for parameters appearing in this paper are reported in Appendix E. The Supplementary Material to this paper includes the source code, where these and other parameter values are contained within Python dictionaries.

2.1. Differential hebbian learning rules

Consider a negative feedback controller as depicted in Fig. 1. The goal of this controller is to make the activity of the S_P neural population equal to that of a population S_D that provides desired values. The output of the S_{DP} population is an M-dimensional error vector $\mathbf{e} = [e_1, \ldots, e_M]$. Population C contains N units whose activity is in the vector $\mathbf{c} = [c_1, \ldots, c_N]$. We assume that

$$\tau_c \dot{c}_i = \sigma\left(\sum_{j=1}^{M} \omega_{ij} e_j\right) - c_i, \tag{1}$$

where:

$$\sigma(x) = \frac{1}{1 + e^{-\beta(x-\eta)}}. \tag{2}$$

The parameter τ_c is a time constant controlling the response latency of the controller's units. ω_{ij} is the synaptic weight for the connection from e_j to c_i. β is the "slope" of the sigmoidal activation function, and η is its "threshold".

For this derivation we assume that internal connections within neurons of the same population have a negligible effect (although this restriction is not necessary, Verduzco-Flores & De Schutter, 2021). All synaptic connections are static, except those from S_{DP} to C, where we assume all-to-all connectivity. The result of this subsection will be two different alternatives for learning in the weights ω_{ij} of these connections. These learning rules are in the following two equations:

$$\dot{\omega}_{ij}(t) = -\alpha\Big(\dot{e}_j(t) - \langle \dot{e}(t)\rangle\Big)\Big(\dot{c}_i(t - \Delta t) - \langle \dot{c}(t - \Delta t)\rangle\Big), \tag{3}$$

$$\dot{\omega}_{ij}(t) = -\alpha\Big(\ddot{e}_j(t) - \langle \ddot{e}(t)\rangle\Big)\Big(\dot{c}_i(t - \Delta t) - \langle \dot{c}(t - \Delta t)\rangle\Big). \tag{4}$$

In both equations α is a learning rate parameter, and Δt is a parameter that approximates the time required for a control signal to propagate around the loop. In other words, a change $\dot{c}_i$ in one of the controller outputs will roughly take Δt seconds to manifest as a change $\dot{e}_j$ or $\ddot{e}_j$ in the errors. The brackets used in the equations indicate an average over all the units in the same population: $\langle \dot{e}(t)\rangle \equiv \frac{1}{M}\sum_k \dot{e}_k(t)$, $\langle \dot{c}(t)\rangle \equiv \frac{1}{N}\sum_k \dot{c}_k(t)$.

Rather than coming from a loss function, the rules in Eqs. (3) and (4) are the result of an informal heuristic procedure, which is described next.

First, we should notice that setting the ω_{ij} weights so they minimize the error is in fact solving the input–output structure problem for the proportional controller in Fig. 1. To reduce the error, we want e_j to activate c_i when c_i's activity reduces e_j. This is tantamount to having the weight ω_{ij} from e_j to c_i be proportional to the negative of their sensitivity derivative:

$$\omega_{ij} \propto -\partial e_j / \partial c_i. \tag{5}$$

In this way the errors that arise will trigger an action to cancel them.

We remain agnostic about the properties of the plant and how its state is transformed into perceived values in S_P, but we assume that the sensitivity derivatives maintain their signs, and that the propagation constant Δt does not change significantly.

Our aim is not to have accurate estimates $\omega_{ij} \approx -\partial e_j/\partial c_i$, but rather to give ω_{ij} a magnitude that is appropriate for feedback control. The Relative Gain Array (RGA) criterion (Bristol, 1966) is a classical method to achieve this, inspiring some of the procedure below (see Appendix A for more details), but due to reasons of biological plausibility we do not exactly implement it.

The most straightforward way to obtain estimates for ω_{ij} may be to let the system settle into a fixed point, and then to produce a perturbation Δc_i, resulting in a change Δe_j for the errors. Weights can be adapted as $\Delta\omega_{ij} \propto -\Delta e_j/\Delta c_i$. While this is feasible, and suggestive of possible learning taking place in unborn mammals (e.g. Brumley, Kauer, & Swann, 2015; Hamburger, 1973) we are interested in the case of online learning, where ω_{ij} is adapted during performance of a behavior.

A simple approach to online learning is to use the correlation of the first derivatives. This provides a measure of whether c_i and e_j change together, in a way that is invariant to their mean values. The resulting learning rule is:

$$\dot{\omega}_{ij}(t) = -\alpha \dot{e}_j(t)\dot{c}_i(t - \Delta t).$$

where Δt is an approximation to the time it takes c_i to change the perceived error e_j, and α is a learning rate.

This approach has three main limitations. Firstly, during behavior the whole $\dot{\mathbf{c}}$ vector acts as the perturbation, so it is unclear which of the c_i units is responsible for an observed change $\dot{e}_j$. Secondly, an observed change $\dot{e}_j$ may not be the effect of any recent $\dot{c}_i$ change, but rather part of the normal flow in state space for the current state. Thirdly, the magnitudes $\frac{\partial e_j}{\partial c_i}$ are functions of the state x_P of the plant (and potentially of $\mathbf{c}$), so they could change sign for different contexts.

We will address each of these 3 limitations. In short, to mitigate the first one we will introduce synaptic competition in the learning rule, and the second one will be handled by introducing a second order derivative, turning Eq. (3) into Eq. (4). The third limitation is more subtle, and will require that we divide our approach into the case when $\frac{\partial e_j}{\partial c_i}$ does not change sign (monotonic control), and the case when the sign changes. Nonmonotonic control will be handled by introducing a reinforcement learning mechanism that changes the configuration of the controller in different regions of state space.

Next we introduce synaptic competition in the learning rule. Using the term $(\dot{c}_i - \langle \dot{c}\rangle)$ rather than $\dot{c}_i$ we expect that on average, weights corresponding to the largest sensitivity derivatives will be enlarged, whereas weights with below-average sensitivity derivatives will shrink. This should allow for errors to be reduced by the c_i units that have the largest effect on them. Notice that lateral connections among the C units is what make the c_k values locally available.

As explained in Appendix A, the RGA criterion relies on a vector perturbation $\Delta\mathbf{c}^j$ that alters only one of the errors (e.g. $\Delta e_l = 0$ for $l \neq j$). The gain of this perturbation is used to select the inputs to the controller, with the idea that when the e_j error arises, the controller response that causes the least interference should be aligned with $\Delta\mathbf{c}^j$. A simple, biologically plausible version of this approach does not seem likely, but a further application of synaptic competition may achieve a similar purpose.

By using $(\dot{e}_j - \langle\dot{e}\rangle)$ in the learning equation rather than just $\dot{e}_j$ we may select only the controller units that have a large effect on e_j. Together with the previous use of synaptic competition, this creates a sparser response that hopefully mitigates the creation of new errors when reducing e_j. Introducing this change leads us to Eq. (3).

The rule in Eq. (3) can effectively configure the feedback loop of simple MIMO systems (see Section 3.1), but it can further be improved. In particular, we may replace $\dot{e}_j$ by $\ddot{e}_j$ in order to remove the effect of changes where $\dot{e}_j$ comes from momentum in the plant rather than the action of a controller. The resulting rule is also what we would obtain from the previous discussion, if we had assumed that a change $\dot{c}_i$ in the output produced a response $\ddot{e}_j$ in the jth error. This simple change leads to Eq. (4).

Eq. (4) is better suited for the control of systems where the plant's dynamics are important. For example, $\mathbf{c}$ may be a force, and $\mathbf{e}$ a displacement or a velocity, so if the plant follows Newton's laws we should expect the correlations to appear among derivatives of different orders.

For the models in this paper, Eqs. (3) and (4) include two additional modifications: connection weights do not change sign, and the sum of weights remains constant. In order to maintain the initial sign of the weights, the whole learning equation is multiplied by ω_{ij}, a strategy called "soft weight-bounding". To maintain the sum constant, a normalization term was included in the equation.

The normalization term leveraged two requirements. First, that all weights from projections starting from the same S_{DP} unit should add to w_{sa}. Second, the sum of all S_{DP}-to-C weights terminating in the same C unit should add to w_{sb}. Let $\zeta_j^{sa} \equiv w_{sa}/\sum_k \omega_{kj}$, and $\zeta_i^{sb} \equiv w_{sb}/\sum_k \omega_{ik}$. Eqs. (3) and (4), using soft-weight bounding and normalization, had the form:

$$\dot{\omega}_{ij} = \omega_{ij}\left(\Omega + \alpha\lambda\left[1 - \frac{\zeta_j^{sa} + \zeta_i^{sb}}{2}\right]\right), \tag{6}$$

where Ω is the right-hand side of either Eq. (3) or Eq. (4), and λ is a scalar parameter. This type of normalization is meant to reflect the competition for resources among synapses, both at the presynaptic and postsynaptic level.

To obtain the derivatives used in the learning rules in a biologically-plausible manner, we approximated rates of change as the difference of two first-order low-pass filters. We assumed $\dot{c}(t) \propto \mathbf{c}_{fast} - \mathbf{c}_{slow}$, where

$$\tau_f\dot{\mathbf{c}}_{fast} = c - \mathbf{c}_{fast}, \tag{7}$$

$$\tau_s\dot{\mathbf{c}}_{slow} = c - \mathbf{c}_{slow}, \tag{8}$$

and $\tau_f \ll \tau_s$.

Elements like $\mathbf{c}_{fast}$ and $\mathbf{c}_{slow}$ can come from feedback connections (cf. Eq. 19 in Lim & Goldman, 2014), but it is also possible that they could represent the concentration of molecules involved in the postsynaptic depolarization, and the subsequent chemical cascades. For example, intracellular calcium concentration has been described as a possible indicator of firing rate, using leaky integrator dynamics (Helmchen, 1999).

Eqs. (3), and (4) are by no means the only options to self-configure a feedback loop. In Appendix B we present two alternative derivations. The first one is meant to explore whether the established reinforcement learning methods are adequate for this problem. The other derivation in Appendix B is based on stability considerations. It is shown that neither of those rules was more effective than Eqs. (3) and (4).

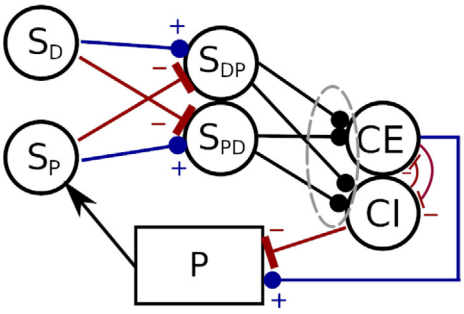

Fig. 2. Negative feedback controller with dual populations, and synaptic weights that are either excitatory or inhibitory. Connections inside the gray dashed oval are adjusted using the learning rules of Section 2.1. Blue circles indicate excitatory connections, red bars inhibitory connections, and arrows are afferent inputs that can be excitatory or inhibitory, but do not change sign. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

2.2. Linear MIMO system controller

The first application of our learning rules (Eqs. (3), (4), (6)) is in the control of a linear plant.

2.2.1. The controller

Unit activities are non-negative, but the controller needs to know the sign of the error. Two basic options for this are: (1) to have units in the S_{DP} population signal negative values as deviations below a baseline level, and positive values as deviations above this level; or (2) to have two separate populations, one for each sign of the error. In other words, this last option amounts to have one population with activity monotonically related to $\max(\mathbf{0}, \mathbf{s}_D - \mathbf{s}_P)$, and another population whose activity is a monotonic function of $\max(\mathbf{0}, \mathbf{s}_P - \mathbf{s}_D)$, where $\mathbf{s}_D$, $\mathbf{s}_P$ are the activities of S_D and S_P, respectively.

We believe our learning rules can work with either solution, but for the purpose of this paper we found the second option to be more appropriate. Accordingly, we modified the architecture of Fig. 1 by separating S_{DP} and C into two separate populations each, resulting in the architecture of Fig. 2. In this figure S_{DP} is excited by S_D, and inhibited by S_P. We assume this inhibition happens through local interneurons, not explicitly modeled. S_{PD} receives the opposite activation of S_{DP}, so that when an error has a positive sign (e.g $s_D > s_P$), a unit in S_{DP} will activate, whereas a negative error will activate a corresponding unit in S_{PD}. In this way the error activities e_j will always be positive, but also capable of signaling errors in either direction. Having two separate populations to represent sensory events, one being inhibited while the other is excited, is termed *dual representation* in this paper.

Units in the S_P, S_{DP}, and S_{PD} populations use sigmoidal units whose activity follows dynamics like those in Eq. (1). To increase biological plausibility and help avoid synchronization, the threshold and slope of sigmoidals in these 3 populations used heterogeneous values, with a random component that ranged from −10% to 10% of their original value.

It can be shown that using linear units and a learning rule as in Eq. (3) in a feedback controller allows convergence to fixed points with non-zero error (see Appendix C). To avoid this the architecture of Fig. 2 uses *CE* and *CI* units that output the integral

of their inputs, in addition to displaying intrinsic noise. Their equations are:

$$\tau_x \dot{x}(t) = x(t)(I_{DP} + I_C x(t))(1 - x(t)), \tag{9}$$

$$\tau_c \dot{c}(t) = x(t) - c(t) + \zeta. \tag{10}$$

$I_{DP} \equiv \sum_k \omega_k^{PD} s_k$, representing the sum of inputs from S_{DP}, S_{PD} times their synaptic weights. $I_C \equiv \sum_k \omega_k^C c_k$ is the sum of inputs arising from CE, CI times their weights; τ_x, τ_u are time constants, and ζ is a white noise process.

Integration of inputs is a basic neuronal computation (Izhikevich, 2000). In Eq. (9) this integration is combined with soft weight bounding to keep the integration factor x between 0 and 1. The term $(I_{DP} + xI_C)$ is an input sum where "lateral" inputs are reduced for small x values. This avoids "winners-take-all" dynamics in C. Eq. (10) simply slows down convergence of the firing rate to the integral, and adds noise. This Langevin equation was solved using the Euler–Maruyama method, whereas all the other equations were solved with the forward Euler method.

One undesired consequence of soft weight-bounding as in Eq. (9), is that when $x(t)$ is very close to 0 or 1 the inputs have little effect, and the unit may stay stuck at that value. To avoid this, if $x(t)$ ever surpassed 0.97 its derivative would become $0.9 - x(t)$. Furthermore, to enhance numerical stability, the derivative of $c(t)$ was clipped if its absolute value became larger than 1.

2.2.2. The plant

The linear plant P is defined by associating each unit c_j^e in CE with a vector $\mathbf{v}_j$, whereas the corresponding unit c_j^i in CI is associated with $-\mathbf{v}_j$. The plant's response was updated as:

$$\tau_p \dot{\mathbf{p}} = \left[\sum_j (c_j^e - c_j^i)\mathbf{v}_j\right] - \mathbf{p}, \tag{11}$$

where c_j^e, c_j^i are also used to denote the activity of those units.

The amount of redundancy in the controller can be adjusted through the number of units in CE, CI, and by the specific values of the $\mathbf{v}_j$ vectors. This information is contained in the connection matrix from C to P, denoted by W_{CP}. Notice that the columns of W_{CP} come from the $\mathbf{v}_j$ vectors.

We used 4 different W_{CP} matrices for our tests. The first one tests the performance of the learning rules in a system with no redundancy. Because of dual representation, W_{CP} was the following block matrix:

$$W_{CP}^{id} = \begin{bmatrix} I_N & -I_N \end{bmatrix}, \tag{12}$$

where I_N is the $N \times N$ identity matrix, and N is the dimension of the plant.

The second W_{CP} matrix was built by using the vectors of an N-dimensional Haar basis (Strang, 1993) as the $\mathbf{v}_j$ vectors . These vectors form an orthogonal basis with positive and negative entries. It is defined for linear spaces where the dimension is a power of 2, so we tested the cases where N is equal to 2, 4, and 8. Quite importantly, all the vectors of the Haar basis have several non-zero entries, so the action of any unit c_j will affect several of the plant variables, but the plant should still be controllable.

Let H_N represent the N-dimensional Haar matrix where the columns are normalized to have unit norm. Our second W_{CP} matrix is the following $N \times 2N$ block matrix:

$$W_{CP}^{Haar} = \begin{bmatrix} H_N & -H_N \end{bmatrix}. \tag{13}$$

The third W_{CP} matrix we used is meant to increase the redundancy in W_{CP}^{Haar}. To this end we increased the number of units in the CE and CI populations, from N to $2N$. Let R_N be an $N \times N$ matrix whose columns are random vectors with unit norm. We used the following connection matrix:

$$W_{CP}^{oc} = \begin{bmatrix} R_N & H_N & -R_N & -H_N \end{bmatrix}. \tag{14}$$

The fourth matrix, W_{CP}^{oc2}, is used to test a worst-case scenario, where redundancy is high, and controllability is not ensured. In this case CE and CI each had $3N$ units. The $\mathbf{v}_j$ vectors were random vectors with unit norm.

All the other static connections used either the identity weight matrix I_N (P-to-S_P, S_P-to-S_{PD}, S_D-to-S_{DP}), or its negative $-I_N$ (S_P-to-S_{DP}, S_D-to-S_{PD}).

2.2.3. Analytical approaches

In order to evaluate the performance of our learning rules, we compared it with two analytical approaches. The first one is based on the Moore–Penrose pseudoinverse. Let W_{SC} be the connection matrix from (S_{PD}, S_{DP}) to (CE, CI). If we set $W_{SC} = -W_{CP}^{-1}$, then, ignoring the sigmoidal nonlinearities, the joint action of the controller and the plant would be akin to the applying the linear transformation $W_{SC}W_{CP} = -W_{CP}^{-1}W_{CP} = -I_N$. Therefore, if W_{CP} is invertible, the controller may be able to achieve decoupled proportional control. Since W_{CP} may not be invertible, or square, we set W_{SC} as the negative of the Moore–Penrose pseudoinverse.

The second approach to obtain W_{CP} is the RGA criterion, as described in Appendix A. In this procedure the designer personally assigns a controller output for each plant variable that requires control. This is done by searching entries that are close to 1 in the relative gain array matrix. The values chosen, however, are to some degree arbitrary. For example, these are the RGA matrices corresponding to the Haar matrices of dimensions 2 and 4:

$$W_{RGA2} = \begin{bmatrix} .5 & .5 \\ .5 & .5 \end{bmatrix},$$

$$W_{RGA4} = \begin{bmatrix} .25 & .25 & .25 & .25 \\ .25 & .25 & .25 & .25 \\ .5 & .5 & 0 & 0 \\ 0 & 0 & .5 & .5 \end{bmatrix}.$$

In order to create W_{SC} connection matrices from the RGA matrices, for each error in S_{PD}, S_{DP} we assigned one C unit. To choose this unit, for each column in the RGA matrix (corresponding to one error) we chose the row whose value was closest to one, and had not been chosen before. If a unit c_i in CE was chosen for error e_j in S_{DP} then the connection from e_j to c_i was 1, and otherwise it was zero. c_i also received a -1 connection from the dual of e_j in S_{PD}. Moreover, a unit c_i' in CI received the same connections as c_i, but with the signs of the weights reversed. When there were more rows than columns, rows not chosen corresponded to units in C that were not assigned to control an error, and received inhibition (a -1 connection weight) from all S_{PD}, S_{DP} units.

The RGA matrices came from this expression:

$$W_{RGA} = W_{CP} \otimes (W_{CP}^{-1})^T, \tag{15}$$

where W_{CP}^{-1} is the Moore–Penrose pseudoinverse of W_{CP}, and $\otimes$ denotes the element-by-element product.

2.3. Monotonic pendulum controller

The second plant model we tested consisted of a pendulum that cannot rotate across a certain angle. This means it bounces back when approaching $\pm\pi$ radians, so the angles stay in the $(-\pi, \pi)$ range.

The pendulum was modeled after a homogeneous rod of 1 kilogram mass, and 50 centimeters length. Gravity was only included for the simulations in Appendix A. Angular acceleration is equal to a torque divided by an inertia moment. The torque

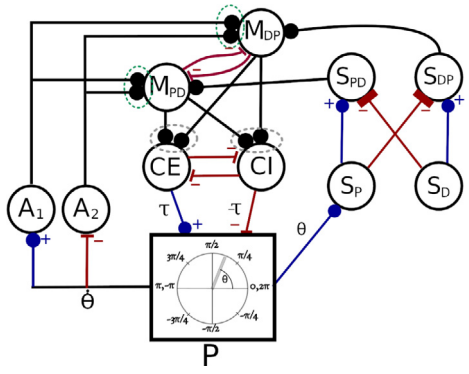

Fig. 3. Basic architecture for control of a pendulum. Each circle represents a single neuron, whereas the square represents the plant *P*. Blue connections are excitatory, red ones are inhibitory. θ represents the current angle in radians, whereas $\dot{\theta}$ is the angular velocity. $\dot{\theta}$ is transformed into positive values by logarithmic units in the *A* population (Eq. (17)). Both units in the *M* population receive all *A* signals. The connections from *A* to *M* (green dotted ovals) evolve following the input correlation rule, and the connections from *M* to *C* units (gray dotted ovals) evolve using the rule from Eq. (4). The output of the *C* units is mapped into either a positive or a negative torque (τ). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

had four components: (1) torque generated from the inputs, (2) viscous friction; (3) and (4) extra torque and viscosity appearing when the angle approached π. These last two torques prevented the pendulum from going across the π angle, causing it to bounce, and increasing the friction while it bounced. Denoting these torques as τ_3, τ_4, their equations are:

$$\tau_3 = -0.001 \tan\left((\theta \% (2\pi))/2\right)^3,$$

$$\tau_4 = -\frac{0.05\ \dot{\theta}}{\left[(\theta + \pi)\%(2\pi) + 10^{-5}\right]^2};$$

where θ is the angle, and % represents the modulo operator.

As explained in Section 3.2, the architecture of Fig. 3 is used for pendulum control. *C* uses two units, one providing clockwise, and another counterclockwise torque. The value in S_D represents a given angle, and the task is to move the pendulum to that angle so activity in S_P and S_D can be equal. S_D does not specify a desired velocity. In order to adaptively incorporate the velocity information into the control loop we introduced a population *M* receiving the afferent activity *A*, consisting of the angular velocity $\dot{\theta}$ in its non-negative (dual) representation. In addition, each *M* unit received one error signal, either s_{DP}, or s_{PD}. The *M* units used the *input correlation* rule (Porr & Wörgötter, 2006) (Eq. (16)) to potentiate angular velocity inputs that correlate with their error input. This allows *M* to send *C* a composite error, resulting in a self-configuring proportional-derivative controller.

The input correlation rule is:

$$\dot{w} = \alpha_{IC} w I_A \dot{I}_{DP}, \tag{16}$$

where I_A is the scaled sum of inputs from the *A* population, α_{IC} is the learning rate, and I_{DP} is either s_{DP} or s_{PD} times a synaptic weight.

The basic rule in Eq. (16) was modified to avoid weights changing signs, and to keep the sum of the weights constant. To make the sum of weights for connections from *A* to *M* equal to w_s, at every simulation step we multiplied the weight times $\zeta_s \equiv w_s / \sum_k \omega_k$. Weight clipping was used to keep individual weights from becoming too large. This means that on every simulation step we set $w = \min(w, w_{max})$, where w_{max} is the largest weight value allowed.

Connections from *M* to *C* populations used the learning rule of Eq. (4), with the modifications of Eq. (6).

All the units (including *CE*, *CI*) in the architecture of Fig. 3 were sigmoidals as in Eqs. (1) and (2), with the exception of S_D, A_1, and A_2. The S_D unit was a predefined function of time containing the values that should appear in S_P for a random sequence of pendulum angles in the range $(-0.7\pi, 0.7\pi)$. The *A* population had units with a rectified logarithmic activation, modeling sensory transducers. Their dynamics followed this equation:

$$\tau_a \dot{a} = \log([1 + I - T]_+) - a, \tag{17}$$

where *I* is the scaled input sum, *T* is a constant threshold, and $[\cdot]_+$ is the "positive part" function (e.g. the identity function for positive arguments, zero for negative arguments).

The *CE*, *CI* units of the pendulum controller had an additional noise term in the dynamics of Eq. (2). They were integrated with the Euler–Maruyama method. Other units were integrated with the forward Euler method, but for the pendulum we used SciPy's (https://scipy.org/) explicit Runge–Kutta 5(4) method.

2.4. Nonmonotonic pendulum controller

The third plant model in this paper is the same pendulum described in Section 2.3, but the torques restricting the pendulum's rotation were removed.

As described in Section 3.3, the architecture of Fig. 3 is limited in how well it can perform under these conditions, but this can be improved if the controller can switch the angle representation it uses depending on the current and desired angles. This is done through an architecture with "actor" and "critic" components, shown in Fig. 4.

The actor component in the architecture of Fig. 4 is the same as the network of Section 3.2, but the torques restricting the pendulum's rotation were removed, and additional units were introduced in order to have an extra coordinate system that could be switched using the input from the *X* unit. The *T* unit of Fig. 4 transforms the θ angle provided by the plant so it uses the second coordinate system. S_P^1 and S_P^2 are just like the S_P unit of Fig. 3, but they differ in the coordinate system used in their inputs (S_P^1 is identical to S_P from Fig. 3, but S_P^2 is not). The two coordinate systems used are described in Fig. 5.

S_P^* is a unit that receives inputs from both S_P^1 and S_P^2; its output is one of those two inputs, selected according to the value of the *X* unit. In engineering terms, S_P^* acts like a multiplexer. When the input from *X* is smaller than 0.5 S_P^* outputs the value from S_P^1, and otherwise it outputs the value from S_P^2. The dynamics of S_P^* follow this equation:

$$\tau_P \dot{s} = I - s, \tag{18}$$

where *I* is S_P^1 when $X < 0.5$, and S_P^2 when $X > 0.5$. One way multiplexing can be achieved is through localized dendritic inhibition (Jadi, Polsky, Schiller, & Mel, 2012, e.g.).

S_D^* is the analog of S_P^*, providing the desired S_P using one of two possible angle representations. For simplicity, the S_D^1 and S_D^2 units were not included in Fig. 4. Desired angles were chosen in the $(0, 2\pi)$ range.

The first component in the critic is a distributed representation of the current perceived angle S_P, and desired angle S_D, provided by the S_1, and S_2 populations, respectively. S_1 and S_2 both consist of 20 units, each of which has a bell-shaped response that increases as the input gets closer to their preferred angle.

The information in S_1 and S_2 (and potentially other inputs) constitutes the state or context characterizing the current situation. The state information is combined in a single population *L*, with the purpose of associating its activity with the right configuration for the controller. To decide which configuration is

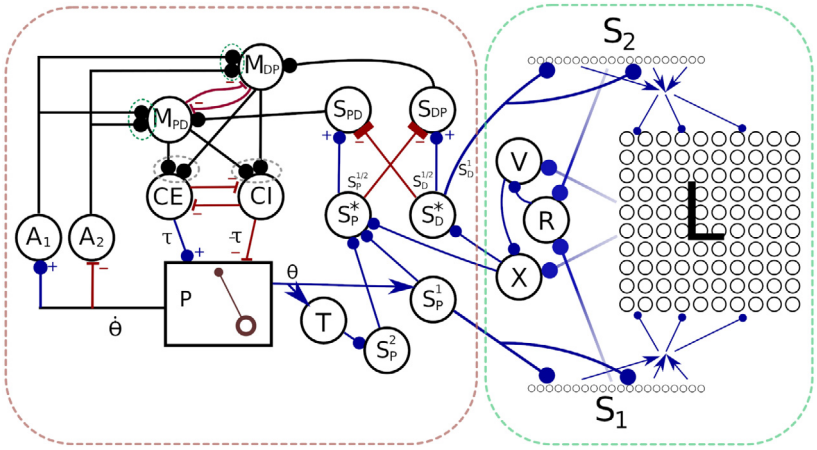

Fig. 4. Actor–critic architecture used in Section 3.3. The actor component (left, red box) is similar to the feedback controller in Fig. 3, but the desired and perceived angle (S_D and S_P) can use one of two different coordinate systems, selected by the input from the unit X in the critic. Moreover, the pendulum can rotate freely. The critic (right, green box) has distributed representations of the desired (S_2) and perceived (S_1) angles, which project to a state representation layer L. S_1 and S_2 also send projections to a unit R that provides a reward based on the similarity of their activation patterns (e.g. the reward is larger when $s_1 \approx s_2$). L sends projections to units V and X. V associates each state of the L layer with a value, using the TD-learning rule with the reward of unit R. X uses the value from V to implement a version of reward-modulated Hebbian learning that associates each state in L with an output. When the output of X is smaller than 0.5 the actor uses a coordinate system where the zero degree angle lies on the positive X-axis. Conversely, when X's output is smaller than 0.5 the actor's coordinate system has a zero degree angle aligned with the negative X-axis (see Fig. 5). The perceived angle in the coordinate system used when $X < 0.5$ is provided by the S_P^1 unit. The S_P^2 unit provides the perceived angle in the alternate coordinate system, which in the simulation is obtained by having a unit T that transforms the angle θ. The S_P^* unit outputs either S_P^1 or S_P^2 depending on the value of X. The S_D^* units performs a similar function for the desired angle.

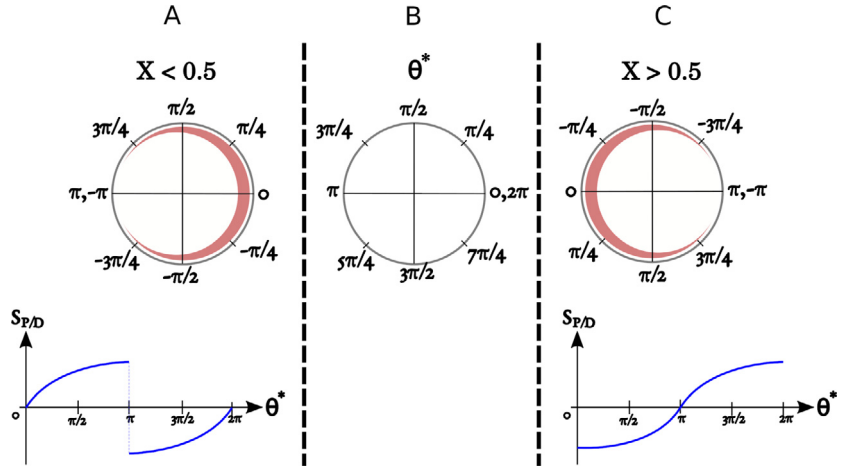

Fig. 5. The two coordinate systems used in the architecture of Fig. 4, and how they affect the activity in S_P and S_D. (A) Top: When the output of the X unit is smaller than 0.5 the first coordinate system is used. In this coordinate system the plant outputs an angle in the range $(-\pi, \pi]$ where the zero-degrees direction is aligned with the positive X-axis, as shown in the circle. The thickness of the red band inside the circle indicates that the system can have a higher effective gain when the desired angle is close to zero degrees. Bottom: the output of the S_P and S_D units as a function of the pendulum's angle, in the coordinate system shown in the center panel. (B) The plots in this figure, and the angles in Fig. 8 panels B and C are reported with respect to this coordinate system, where the angles are in the $[0, 2\pi]$ range. (C) Top: When the X output is larger than 0.5 the coordinate system undergoes a 180-degree rotation, so that the activity of the S_D and S_P units as a function of the pendulum's location is now as shown in the plot at the bottom of this panel.

best, we rely on reinforcement learning techniques. In particular, L provides inputs to a unit V that learns a *value* associated with the state using a version of the *temporal differences* learning rule (Schultz et al., 1997). The value provided by the V unit is used by another unit, called X in Fig. 4.

X learns to associate the state in L with an output that configures the feedback controller. So that X provides configurations that increase the value, the connections from L to X use a version of reward-modulated Hebbian learning, where the output of V is used as the reward (Eqs. (23), (24)).

The V unit has dynamics:

$$\tau_V \dot{v} = \sigma\left(\sum_j w_j^V L_j - \langle I_V \rangle\right) - v, \tag{19}$$

whereas the X unit has dynamics:

$$\tau_X \dot{x} = \sigma\left(\sum_j w_j^X L_j - \langle I_X \rangle\right) - x. \tag{20}$$

τ_V and τ_X are time constants, $\sigma(\cdot)$ is the sigmoidal function, L_j is the activity of the jth unit in L, and $\langle I_{V/X} \rangle$ is a low-pass filtered version of $\sum_j w_j^{V/X} L_j$.

In the Temporal Differences (TD) learning rule (Sutton & Barto, 2018) the value function is $V(s_t) = \langle \sum_{t=1} \gamma^{t-1} R(t) \rangle$, where γ is a discount factor that reduces the importance of later versus imminent rewards. The V unit learns to approximate this function in continuous time by adjusting its synaptic weights with the following equation:

$$\dot{w}_j(t) = \alpha_V \left[\bar{R} + \gamma v(t) - v(t - \Delta t_v)\right] L_j(t - \Delta t_v), \tag{21}$$

where $\bar{R} = (R(t)+R(t-\Delta t_v))/2$ approximates the integral of R for the past Δt_v seconds. Two additional terms were added to this equation in order to provide weight normalization and to have the sum of the weights near zero. The final equation had the form:

$$\dot{w}_j(t) = \Omega + \eta_1 w_j \left(\frac{W}{\sum_k |w_k|} - 1 \right) - \eta_2 \bar{w}, \tag{22}$$

where Ω is the RHS in Eq. (21), W is the desired value for the sum of the absolute value of the weights, η_1, η_2 are constants, and $\bar{w}$ is the mean of all w_j weights for connections from L.

To adjust the weights from L to X we introduce a version of reward-modulated Hebbian learning capable of handling the temporal credit assignment problem associated with tracking a target angle in real time. For this purpose the weights were updated intermittently, whenever the S_D value changed (e.g. whenever its derivative crossed a threshold), an event that we will call a *transition*. Let t^i be the time when a transition happens, and t^{i-1} be the time of the previous transition. Whether a weight is potentiated or depressed depends on two factors. The first one is the $V(t^i) - V(t^{i-1})$ difference, indicating whether the value increased between transitions. The second factor is whether a sufficiently high reward was reached, and how quickly. The concrete update equation is:

$$\dot{w}_j(t) = \alpha_X \Delta V(t) \left(L_j(t^{i-1}) - \bar{L}(t^{i-1}) \right) \left(X(t^{i-1}) - 0.5 \right), \tag{23}$$

$$\Delta V(t) \equiv \left[V(t) - V(t^{i-1}) + \eta_X (t - t^R) \right], \tag{24}$$

where η_X, α_X are constant parameters, and t^R is the last time when the reward value was above a given threshold. t^R is reset after each transition. $\bar{L}$ denotes the average over all the L_k inputs. It is assumed that X maintains a constant value between transitions, and the term $X(t^{i-1})$ refers to the value that X has in the interval (t^{i-1}, t^i).

The advantage of learning only at transition times for the problem of distal rewards is discussed Section 3.4.

Since the states in L must be associated with values or configurations, it greatly helps if the representations in L are linearly separable. To this end L does an expansive recoding of its inputs (Illing, Gerstner, & Brea, 2019) that permits V and X to learn functions of the state using a single layer. The L layer consisted of 100 units, arranged in a 10×10 grid. Each unit in L was maximally responsive to a particular combination of the desired and current angles, with its response decreasing exponentially according to the distance between the current state and its preferred angles.

The last component of the critic is the R unit, which provides a reward value based on how similar the patterns in S_1 and S_2 are. Computation of this reward is straightforward when S_1 and S_2 have the same structure, meaning that for each unit in S_1 there is a corresponding unit in S_2, and vice versa. This is possible, for example, when S_1 and S_2 are two different layers of the same cortical area, and their corresponding units are different populations from the same microcolumn (Mountcastle, 1997).

The critic, as originally designed, significantly slowed the simulation. We describe its original implementation, and how this was simplified.

In the original implementation of the critic the S_1 and S_2 populations were units that responded maximally when their input is close to a preferred value I_{max}. Their dynamics followed the equation:

$$\tau_s \dot{s} = \mathrm{e}^{-b(I-I_{max})^2} - s, \tag{25}$$

where τ_s is a time constant, b controls the sharpness of the tuning, and I is the scaled sum of inputs. The units in L were sigmoidals (Eqs. (1), (2)), but the connection matrices from S_1 and S_2 to L ensure that each unit in L responds maximally to a particular combination of S_1 and S_2 inputs. The resulting representation is similar to radial basis functions.

Both S_1 and S_2 had 20 units each, whereas L contained 100 units. Independently simulating the dynamics and delayed transmissions for these 140 units slowed down the simulation by an order of magnitude. Thus, for practical reasons, the implementation of the network used multidimensional ODEs that encapsulated the response of L in a vector function. The variables in the multidimensional ODEs do not represent the activation of the L units; instead they directly model the evolution of the synaptic weights from L to V, and from L to X. The V and X units have consequently 101-dimensional dynamics: 100 variables for the synaptic weights, and one variable for the output of the unit.

The activity of the L "units" in the multidimensional ODEs was calculated with:

$$a_L = \mathrm{e}^{-bd^2}, \tag{26}$$

where b controls the width of the tuning, and d is a measure of the distance between the current "state", and the preferred "state" of the system. This "state" is the pair (θ, θ_D), containing the current and desired angle. The distance was obtained using the L^2 norm, but taking into account that the angles are periodic.

The V and X units had dynamics as in Eqs. (19) and (20), respectively.

The R unit provides a reward value that indicates when the desired angle θ_D and the current angle θ are close. This unit was implemented as the function $r = \mathrm{e}^{-d^2}$. Given θ and θ_D in the $[0, 2\pi]$ interval:

$$d = \min\big(|\theta - \theta_D|,\ 2\pi - \max(\theta, \theta_D) + \min(\theta, \theta_D)\big).$$

Learning in the connections from L to V used the version of TD-learning in Eqs. (21), (22). Learning in the connections from L to X relied on Eq. (23). The software implementation of this equation uses slightly modified terms to deal with the fact that updates should happen during transitions (e.g. at time t^i), but they cannot happen instantaneously. In particular, the learning rate is modulated by a term that decays exponentially after a transition. As with learning of the weight in the V unit, Eq. (23) receives the additional terms in Eq. (22) to normalize the sum of weights and to make the weights have zero mean.

2.5. Inverted pendulum controller

The fourth plant model has the same pendulum with unrestricted rotation of the third model, but gravity is included.

The architecture used to control the pendulum is also much simpler, as described in Section 3.4 and in Fig. 9.

The output of the X unit approaches either 1 or -1, depending on whether the sum of its inputs times their synaptic weights is positive or negative, respectively:

$$\tau_X \dot{x} = \tanh\left(\beta \Big[\sum_j w_j^X S_j - \langle I_X \rangle \Big] \right) - x. \tag{27}$$

τ_X is a time constant, β is a slope parameter, S_j is the activity of the jth unit in S, and $\langle I_X \rangle$ is a low-pass filtered version of $\sum_j w_j^X L_j$.

An output of 1 produces a positive (counterclockwise) torque τ, and -1 produces a torque of $-\tau$. τ is not sufficient to raise the pendulum from its rest position (at $\frac{3\pi}{2}$ radians) to an angle beyond the horizontal line. X only changes its output value at the transition times. The reward unit R has $\sin(\theta)$ as its output, providing vertical height. The S population provides a distributed representation of the angle using 20 units, in the same manner as before.

Learning in the connections from L to X relies on Eq. (23). An additional term was used to maintain the sum of absolute weight values close to a value W, leading to the equation:

$$\dot{w}_j(t) = \Omega + \alpha_X w_j \left(\frac{W}{\sum_k |w_k|} - 1 \right), \tag{28}$$

where Ω is the RHS in Eq. (23).

As described in Section 3.4, this rule was applied at the times when R'' and R' were negative, and the time since the last transition was at least t_{trans} seconds.

2.6. Parameter adjustment

Parameters for all models were manually adjusted to obtain a reasonable dynamic range for each of the neuronal populations, and learning rates were adjusted so the task could be learned relatively fast. Any other parameter adjustments were done by trial and error, although little parameter search was required. There were two exceptions for this.

The delays in the learning rules were obtained by an analytical procedure described below.

The delay Δt in the $\dot{c}_i(t - \Delta t)$ terms of the learning rules is meant to synchronize an action in c_i with the consequent reaction in e_j. To this end, Δt should contain 4 transmission delays as the signal from C goes through P, S_P, S_{DP}, and back to C. Moreover, the units at each of these stages have a response latency. Since the equations of these units resemble those of a linear first-order low-pass filter (e.g. Eq. (1)), its phase shift can be used to approximate the response latency of the units. In particular, a signal $\sin(\upsilon t)$ has a filtered response $x(t)$ that is the solution of: $\tau\dot{x} = \sin(\upsilon t) - x$. This equation can be solved exactly, and its solution is a sinusoidal whose time delay with respect to the input is $\arctan(\tau\upsilon)/\upsilon$. Using the most dominant frequency observed in the activity of the units as υ, a term like this can be obtained for each of the populations that the signal goes through, providing response latencies that are added into the Δt delay.

Parameters for the X and V units were first tuned manually, and then further adjusted using 6 generations of a standard genetic algorithm, included in the source code.

3. Results

3.1. Adaptive control of a linear MIMO plant

As described in the Methods, we produced 2 learning rules (Eqs. (3), (4)) to infer the input–output structure of a feedback system. We now show how those rules performed when used to implement proportional control of a linear plant.

As described in the Methods (Section 2.2), the controller used the architecture in Fig. 2. The plant's response came from a linear combination of vectors $\mathbf{v}_j$, where each vector is scaled by the activity of a unit in CE or CI. These vectors defined the connection matrix W_{CP} from C to P, and the degree of redundancy in the controller would depend on that matrix.

We used 4 types of W_{CP} matrices. W_{CP}^{id} created a controller where each unit in C affects only one error. This connection matrix tests the simplest scenario, where the controller can act as several independent 1-dimensional controllers; it just needs to decide which output corresponds to which error.

The matrix W_{CP}^{Haar} tests the next scenario, in which the number of units in CE (or CI) is equal to the dimension of the plant, but the activity of each unit in the controller has an effect on more than one of the errors. The $\mathbf{v}_j$ vectors form an orthonormal basis (the Haar basis, Strang, 1993) so in theory C can produce any desired vector output in P, but our system must do it by choosing the right weights in the connections from (S_{DP}, S_{PD}) to (CE, CI).

For the third connection matrix (W_{CP}^{oc}), the number of units in CE and CI is twice the dimension of the plant. Half of the $\mathbf{v}_j$ vectors in CE to P connections are random unit vectors, and the other half are the $\mathbf{v}_j$ vectors used in W_{CP}^{Haar}. This increases the redundancy, not only in the sense of one controller activity c_i affecting more than one error signal e_j, but also in the sense that there are countless ways to achieve a desired output in the plant.

For the final type of connectivity (W_{CP}^{oc2}), all $\mathbf{v}_j$ vectors are random, and there are 3 for each unit in S_P. This is in general a much harder case, with greater redundancy and no guarantees of being solvable, used to illustrate a worst-case scenario.

Simulations are shown for 1, 2, 4, and 8 units in S_P, which is also the dimension of the plant, denoted as N in this section. Results are summarized in Fig. 6. The third and fourth types of connectivity are respectively labeled *overcomplete*, and *overcomplete2* in this figure.

In panel A of Fig. 6 the performance of the rules is measured as the norm of the $\|S_D - S_P\|$ error for the second half of the 400 s simulation. The norm of the difference of two unit vectors with random entries in the (0, 1) range is expected to be around 0.5. This is a first order approximation to the error we should expect for a system that has done no learning. We refine this control by running simulations with random initial weights and static synapses, resulting in the gray markers of the first two plots.

In order to put the performance of our learning rules into context, we also determined the input–output structure of the controller using two analytical methods (see Section 2.2.3). The first one places the Moore–Penrose pseudoinverse of the W_{CP} matrix in the W_{SP} matrix connecting (S_{DP}, S_{PD}) to (CE, CI). The second one uses a simple, automated version of the RGA criterion (Bristol, 1966).

Quite remarkably, panel A of Fig. 6 shows that the learning rules perform almost the same as the pseudoinverse method, and outperform the version of the RGA method we implemented.

Both the analytical methods and the learning rules perform almost optimally with the system that has no redundancy (the "identity" case, with the W_{CP}^{id} matrix). The error increases slightly for larger values of N, because proportional control is being done in a MIMO system with delays, response latencies, and noise, so it is inevitable that some error will accumulate for each controlled variable.

The type of error that accumulates can be observed in panels B–E of Fig. 6, showing simulation data for the "overcomplete" case (with the W_{CP}^{oc} connection matrix) with dimension $N = 2$, both for the pseudoinverse method, and for the rule of Eq. (3). The intrinsic noise of the CE, CI units causes most of the noisy appearance of the activity traces. Without this noise the system may not learn due to insufficient exploration.

In the case of W_{CP}^{Haar} (red triangles in panel A of Fig. 6), the pseudoinverse method and the two learning rules have virtually the same performance. From here on the RGA method largely fails, because in the simple form that we use each error is to be controlled by a single controller unit. This is unfeasible when each c_i unit affects many e_j values due to the structure of W_{CP}.

For the system with the W_{CP}^{oc} connection matrix, the pseudoinverse method and the learning rules also have similar performance. Despite redundancy, the local rules can perform a computation that is tantamount to inverting the connection matrix from C to P.

In the case of the redundant, random connection matrix W_{CP}^{oc2}, none of the methods performs well, as would be expected from a scenario with such level of random redundancy.

The amount of error in the system (panels B–E) is what should be expected for simple proportional control in this scenario. Animal motor control does not seem to rely on one monolithic controller that does both the input–output mapping, and ensures fast

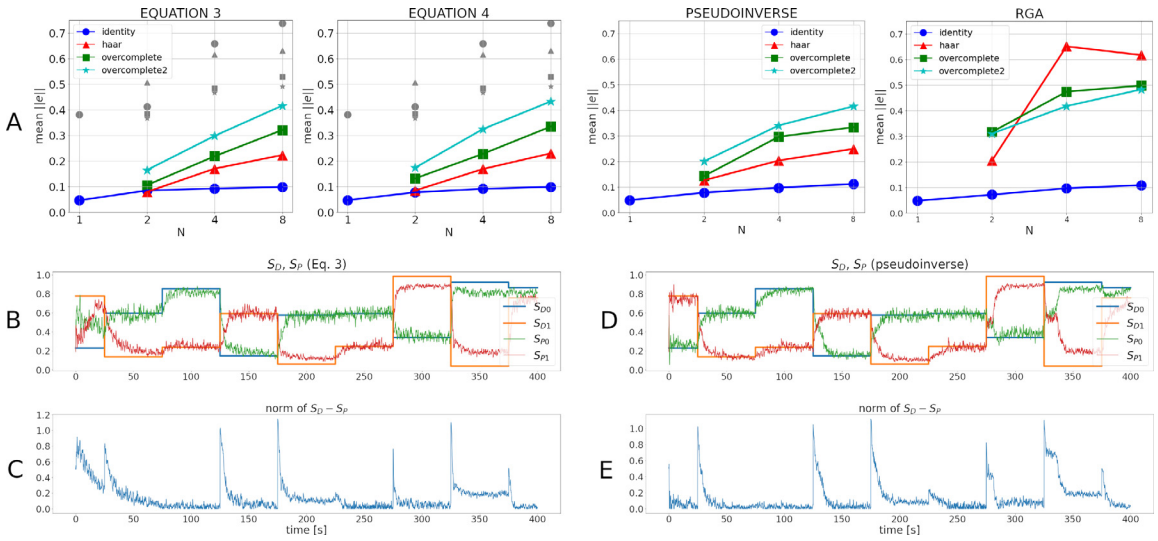

Fig. 6. (A) Simulation results for 4 types of connectivity matrices in a linear plant model for the two learning rules in Section 2.1, and for two analytical methods. The number of values in S_P is labeled N in the x-axis. The y-axis indicates the time average of the norm $\|\mathbf{s}_P - \mathbf{s}_D\|$ for the second half of the 400 s simulation, where $\mathbf{s}_P$ is the vector of activities in S_P, normalized so it has a unit norm for $N > 1$, and likewise for $\mathbf{s}_D$. Each marker is the average from 20 individual simulations with different random initial weights. Gray markers indicate the same mean error when a simulation with the same characteristics was run with static synapses. In the case $N = 1$ only the identity matrix is tested. (B) Activity of the S_D and S_P units for the first 400 s of an example case with N = 2 units in S_D and S_P, an "overcomplete" W_{CP} matrix, and the learning rule of Eq. (3), resulting in an average $\|S_D - S_P\|$ value of approximately 0.18 for the first half of the simulation, and 0.1 for the second half of the simulation. (C) $\|S_D - S_P\|$ norm for the simulation in panel B. (D) A simulation as in panel B, but the connection matrix from S_{DP}, S_{PD} to C comes from the pseudoinverse method. The $\|S_D - S_P\|$ average value was around 0.12 for both halves of the simulation. (E) $\|S_D - S_P\|$ norm for the simulation in panel D.

and accurate performance. Instead, there is a cerebellar system to compensate for things such as timing, momenta, and interaction torques (Bastian, Martin, Keating, & Thach, 1996; Manto et al., 2012). Many cerebellum models perform this type of supplementary control (e.g. Dean & Porrill, 2008; Kawato & Gomi, 1992; Porrill et al., 2004; Verduzco-Flores & O'Reilly, 2015), relying on a pre-existing feedback control structure.

Although these two learning rules do not explicitly consider the full error $\|\mathbf{e}\|$, reducing the components of $\mathbf{e}$ individually works well together with a type of weight normalization that keeps the L^1 norm (sum of absolute values) of the $\mathbf{e}$ vector constant. Normalizing incoming and outgoing weights (see Methods, Section 2.1) allows the network to scale its size without requiring parameter changes, and also maintains the balance between excitation and inhibition due to the architecture of Fig. 2.

One limitation of the approach in Section 2.1 is that it requires some knowledge of the Δt delays inherent in the system. This is reasonable for neurons that receive the effects of their activation with a short, and relatively fixed latency. This would be the case, for example, of spinal interneurons receiving feedback from muscle afferents and motor cortex. The fact that the delay can also depend on the frequency of the oscillation (see Methods) does not seem to impair the system, as only few dominant frequencies tend to naturally emerge.

3.2. Monotonic control of a pendulum

The linear plants in Section 3.1 show how that the learning rules can resolve moderate amounts of redundancy in the controller, but they are not representative of physical systems. Next we consider feedback control of a pendulum.

The error signal in this case is the difference between desired and current angles. So that this error remains monotonic we make the pendulum stop when it approaches $\pm\pi$ radians (see Methods). This, however, does not change the fact that simple proportional control (as in the architecture of Fig. 2) may be unstable, despite the addition of viscous friction. This is due to the delay in the control response, which is similar to the delays observed in human reflexes (Capaday, Forget, & Milner, 1994). Such an effect highlights the usefulness of including transmission delays and response latencies in this study.

As discussed previously, most cerebellar models assume a pre-existing feedback controller, whose performance they improve. And as discussed in Section 4.3, configuration of this feedback controller may not be innate. If this is the case, the feedback controller cannot rely on the cerebellum while it is learning its input–output structure, and must somehow compensate for its unstability.

In systems where proportional control is unstable, oftentimes proportional-derivative control can restore stability (Sontag, 2013). Animals can receive muscle contraction velocity and tension information from their muscle afferents (Shadmehr & Wise, 2005). We extended the architecture of Fig. 2 to include angular velocity information while still allowing for self-configuration using the learning rules of Section 2.1. The result is the architecture in Fig. 3.

The S_D population in Fig. 3 does not specify a desired velocity, so a velocity error cannot be produced in the same way as the angle error. In order to adaptively incorporate the velocity information into the control loop we created a network resembling the long-loop reflex of the animal motor system, which includes not only the spinal cord, but also the primary motor and sensory cortices.

In Fig. 3 we introduced a population M receiving the afferent activity A, consisting of the angular velocity $\dot{\theta}$ in its non-negative (dual) representation. In addition, each M unit received one error signal, either s_{DP}, or s_{PD}. The M units used the *input correlation* rule (Porr & Wörgötter, 2006) (Eq. (16)) to potentiate angular velocity inputs that correlate with their error input. This allows M to send C a composite error, resulting in a self-configuring proportional-derivative controller.

C uses two units, one providing clockwise, and another counterclockwise torque. The value in S_D represents a given angle, and the task is to move the pendulum to that angle so activity in S_P and S_D can be equal.

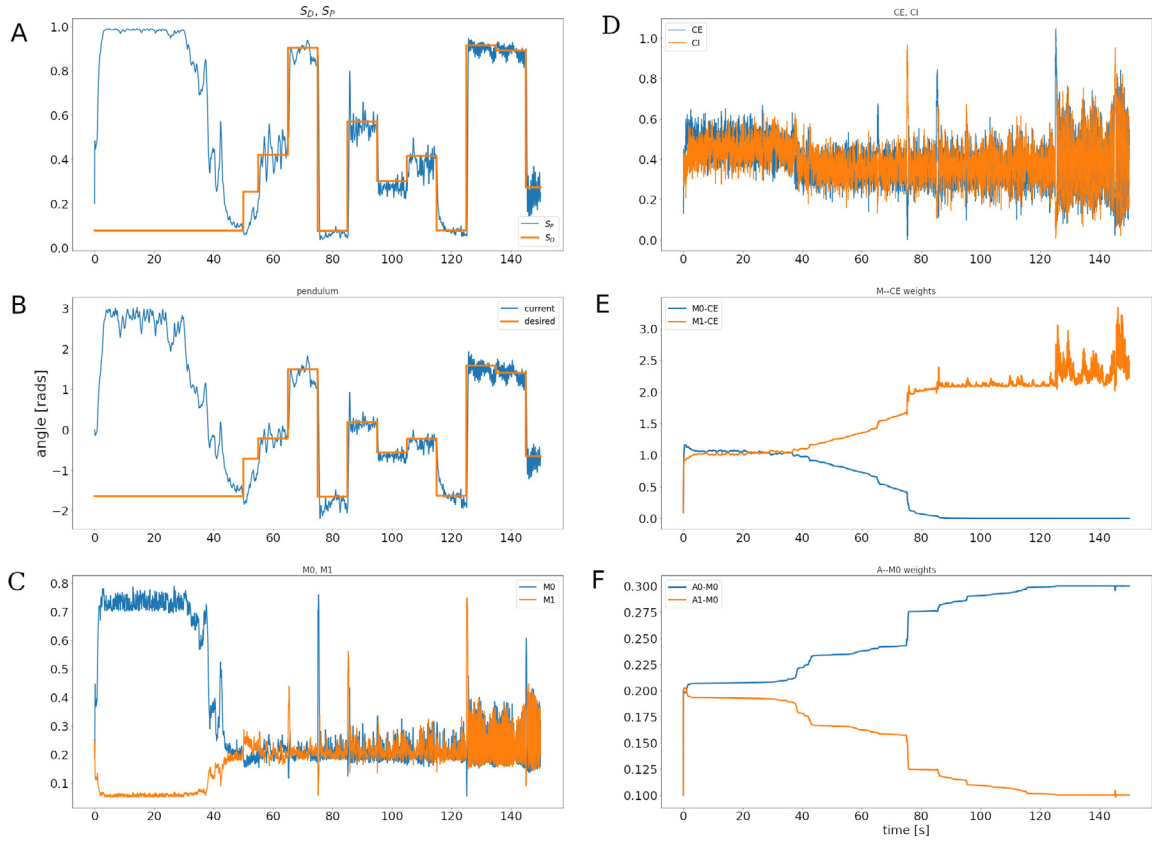

Fig. 7. First 150 s of a simulation where the architecture of Fig. 3 is used so a pendulum can track a desired angle (no gravity). The system learns to track the desired angle in about 60 s. (A) Activity of the S_P unit, with the perceived angle, and S_D, with the desired value for S_P. (B) Angle of the pendulum, and the desired angle. (C) Activities of the two units in population M. (D) Activities of the two units in population C. (E) Synaptic weights for the connections from the two M units to the CE unit. (F) Synaptic weights for the connections from the two A units to one of the M units.

Fig. 7 shows a representative simulation result, where the system learns to perceive a desired S_D angle in S_P using the learning rule from Eq. (4) in a pendulum with no gravity. A similar figure for the case when gravity is present is in Appendix D (Fig. D.12). Fig. 7 shows the appropriate weights emerging in seconds; this time depends on the initial conditions and the learning rates. After a couple of minutes the weights reach their final values, which remain stable thereafter.

An interesting feature of this system is the interplay between antagonist (dual) units, seeking a balance between excitation and inhibition. Panel B of Fig. 7 shows how each time the target changes one of the M units activates more than its dual, producing a correction. The magnitude of the error determines difference in the activity of dual M units. In the absence of gravity the error can remain close to zero without exerting any torque, and at this equilibrium point both M units have the same activation level, sending no net excitation to CE and CI.

All the units in Fig. 3 have a sigmoidal activation function, except for those in population A, which have a logarithmic activation (Eq. (17)). Sigmoidals have a non-zero output in the absence of input (Eq. (2)). Thus, in the absence of error the units may still have an output, but antagonist units will have the same activation level, resulting in no action. When gravity is present a constant torque is required to keep the error close to zero. Since the system exerts no action in the absence of error, gravity implies that either we will have a steady state with non-zero error, or the angles will oscillate around their target values. Which of these scenarios presents depends on the gain of the system, with higher gains tending to produce oscillations around the target. Moreover, the C units present intrinsic noise, used so the system can produce plasticity-inducing movements when learning begins. All of these factors explain the oscillations observed in Figs. 7 and D.12.

3.3. Non-monotonic control of a pendulum

The two terms in the synaptic learning rules of Eqs. (3) and (4) are monotonic functions of $\dot{e}$ (or $\ddot{e}_j$) and $\dot{c}_i$. If c_i activity can make e_j either grow or decrease depending on the context, correlations will be inconsistent, making the approach used by these equations unlikely to succeed.

A further complication is that the representation of sensory signals may not always be germane for negative feedback control. Muscle afferents use a firing rate code that provides information about the muscle's length, speed, and tension, but other afferents may provide a distributed representation, using a population of neurons where each one is tuned to a particular range of values (e.g. direction tuning in somatosensory cortex Pei, Hsiao, Craig, & Bensmaia, 2010, or retinotopic location tuning in posterior parietal cortex, Andersen, Essick, & Siegel, 1985).

It is evident that learning a static input–output structure for a feedback controller is not sufficient for the control of arbitrary plants. Much flexibility could be gained if the input–output structure could adapt according to the context. To this end, we borrow concepts from the *actor–critic architecture* used in reinforcement learning (Sutton & Barto, 2018). The general idea is to have a feedback controller as an *actor* component that can adapt its input–output structure. When entering a context where the current controller structure is not appropriate, a *critic* component can indicate this, so the controller alters its *configuration*.

The meaning of "altering the controller configuration" can have several interpretations (see Discussion, Section 4.2). We present one illustrative example in this section.

Consider the architecture in Fig. 3, and suppose the pendulum was able to rotate without restrictions. Given our choice of angle

representation in the S_P and S_D units (selected to mimic the representation of length and velocity used in muscle afferents), letting the pendulum rotate freely will produce a discontinuity around π radians, where a small variation in the angle creates a large variation in the firing rate. This simple change greatly alters the pendulum control problem from the previous section, in the sense that an optimal solution can no longer be achieved by a controller that responds proportionally to $(\theta_D - \theta)$, where θ_D is a desired angle, and θ is the current angle. This is because the proportional controller will not cross the angle where it has a representation discontinuity, so even if θ_D and θ are very close (say, 179°and 181°), the controller may not move the pendulum through the shortest path. An optimal solution is thus beyond the reach of the learning rules in Section 2.1, which cannot handle the non-monotonicity present in the angle discontinuity.

Because of this phenomenon we can test our ideas directly on the pendulum controller of the last section, with minimal modifications. In particular, we allow the pendulum to rotate freely, but we also add the possibility of using a different angle representation (inspired by how corticospinal signals can modulate ascending afferents through presynaptic inhibition (Goulding, Bourane, Garcia-Campmany, Dalet, & Koch, 2014)). In this way the synaptic learning rules from Section 2.1 can still be used as before. We also add a "critic" component to the architecture, used to select which angle representation is used. The result is shown in Fig. 4, and details are in the Methods section.

In abstract terms, the "critic" has a representation of the state, including the desired and perceived angles for the controllers. From this, it produces a value associated with each state, and this value is used to configure the controller, which in this case means selecting an angle representation (Fig. 5).

Allowing the critic to select the coordinate system for each state significantly increases the average value of the reward (the output of the R unit) in the case where the gain of the inputs from CE and CI to P is reduced. Optimal performance in this task has to leverage two limitations. First, as mentioned above, when the shortest path between the current and desired angles crosses either 0 or π radians, one of the angle representations makes the controller follow the longer path. This affects the time to approach the desired angle. Second, due to the limited dynamic range of the sigmoidal units, the gain of the controller is greatly reduced when the desired angle is away from the zero-degree direction (Fig. 5). The critic must thus choose a coordinate system that has enough gain near the desired angle. This affects the error in the steady state.

Fig. 8 shows the results of 20 simulations where the network was first run for 800 s with random X values (either $X \approx 0$ or $X \approx 1$ on each reach) to provide a mean reward $R1$. Next the network was run for 400 s with the X output being driven by the inputs from L, providing a mean reward $R2$. The average increase in reward was approximately 0.136 ($p < 1^{-10}$, paired T-test), and the largest $R1$ value in the 20 simulations was smaller than the smallest $R2$ value.

In the simulations presented in Fig. 8 a different S_D value was presented every 4 s. In the first 800 s the feedback controller would attempt to make $S_D = S_P$ using one of the two angle representations, selected randomly, and as it did so learning took place in the connections from L to V and from L to X. The V unit was learning to estimate the value of different states, and the X unit was learning which output was associated with an increase in this value.

The strategy that emerged through learning can be glimpsed from the weights in the projections from L to X, and the outputs that they implied, as shown in panels B and C of Fig. 8. In this figure the horizontal axis represents the current angle, and the vertical axis represents the desired angle. Each of the squares in this 10×10 grid correspond to the unit in L that is maximally responsive to the corresponding combination of angles. In panel B the color of the square encodes the magnitude of the synaptic weight in the projection of that L unit to X, with brighter squares having a larger weight. In panel C (right half) yellow squares indicate an output close to 1, which causes the second coordinate system to be used (Fig. 5C). As can be observed, this second coordinate system is preferred when the desired angle is close to π radians, whereas the first coordinate system is preferred when the desired angle is close to 0 or 2π radians. The effect that this has on the tracking performance can be observed by contrasting panels D and F.

3.4. Control of an inverted pendulum

In the actor–critic architecture of Section 3.3 the weights of the unit X are updated when the desired angle changes. We refer to these events as *transitions*. Let t_1 denote the time when a transition happens, and let t_0 be the time when the previous transition occurred. The weight update rule (Eq. (23)) only cares about the difference in values $V(t_1) - V(t_0)$, with a possible time penalization to discourage large $(t_1 - t_0)$ periods. Ignoring the intermediate $V(t)$ values allows the controller to explore the gradient of the value function in larger steps. In this subsection we present a simple example to illustrate how this idea can be exploited.

Consider the *inverted pendulum* problem, where the goal is to make the pendulum reach the vertical position, at $\frac{\pi}{2}$ radians in the coordinate system of panel B in Fig. 5. This problem is trivial using a controller as in the previous subsections, with enough gain to overcome gravity. To make this example illustrative we removed the controller and most of the critic from the architecture of Fig. 4, leading to the reduced system in Fig. 9A.

Learning in this model happens in the connections from S to X, using the reward-modulated Hebbian rule of Eqs. (23), (24). This system will generally not learn to point the pendulum upwards using random transition times; it is necessary to have a particular strategy. Denoting the output of X as the *configuration*, we outline our strategy as follows:

1. Adopt a configuration (e.g. give X a fixed output value).
2. Predict the time t^* when $V(t)$ will attain its maximum (updating the prediction online).
3. Perform a transition at time $t = t^*$.

The inverted pendulum problem is simple enough that a value function V and a controller C as those in Fig. 4 are not required. In the case of Fig. 9 R takes the place of V, and X takes the place of C. For this particular case the strategy above can be adapted into a simple rule: *if both* $R'' < 0$ *and* $R' < 0$, *do a transition every* t_{trans} *seconds.*

This rule comes from estimating $V(t)$ (in our case, $R(t)$) as a quadratic polynomial function of time: $V(t) = V''(t)t^2 + V'(t)t + V(t_0)$, using the latest observed values of $V''(t)$ and $V'(t)$. If we want to maximize $V(t)$, having $V'' > 0$ means that eventually the value will grow as time increases, so no transition should be made. On the other hand, if $V'' < 0$, the polynomial attains its maximum value at the point when V' becomes negative, so no transition should be done while $V' > 0$. The parameter t_{trans} (which could be a random value) determines how much time the controller is allowed to explore a configuration before a different one is potentially adopted.

The result of using this rule to decide when to apply weight updates with Eq. (23) is shown in Fig. 9. The controller only has two possible torques, the angle representation is not very precise, and there are temporal delays, so the best that can be expected is oscillations near the $\pi/2$ angle. Still, the system learns

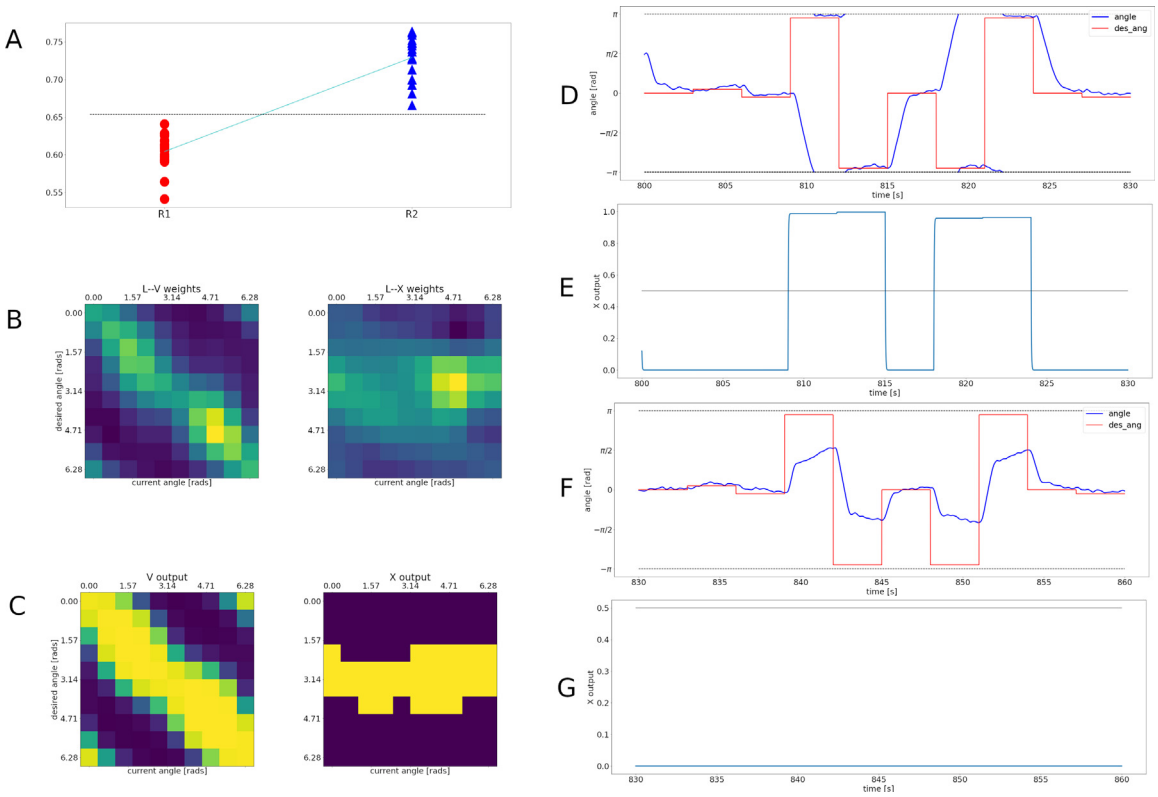

Fig. 8. Performance of the reinforcement learning model. (A) average reward in 20 simulations when the X value is randomly selected (R1, red circles) compared to the reward when the X value is produced by the inputs from L after training (R2, blue triangles). (B) Left: Connection weights for the projections from L to V after training. These weights were selected arbitrarily from one of the 20 simulations used for panel A. Right: Connection weights for the projections from L to X after training. (C) Left: Steady-state activation values for the V unit when the desired and current angles are those preferred by each of the 100 units in L. Right: the corresponding steady-state activation values for X. (D) The desired (red) and current (blue) angles through a 30 s simulation. This panel shows tracking of the desired angles after 800 s of learning with random X values. After this learning period the X values were determined by the input from L, and the 30 s simulation in this plot began. The desired angles were selected to illustrate the difference when using the actor–critic system, compared to simple feedback control as in Section 3.2. Angles in the y-axis are in the coordinate system of panel A in Fig. 5. (E) The output of X during the 30 s simulation of panel D. (F) A simulation with the same desired values as in panel D, but this time the output of X was fixed near 0, forcing the use of a single coordinate system. (G) The output of X during the 30 s of the simulation in panel F. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

to maintain the pendulum near the vertical position for extended periods of time, and it brings it back on top soon after it falls (Fig. 9B).

4. Discussion

4.1. From correlations to reinforcement learning

In this paper we presented synaptic learning rules that automatically configure a feedback control system. This control system is entirely agnostic about the plant being controlled, so its configuration involves finding the input–output structure of the controller, which is akin to finding the sensitivity derivatives, or the control Jacobian of the system.

In Section 2.1 we derived 2 different learning rules to find this input–output structure in the case of a monotonic relation between the control signals and the error, and 2 other variations are in Appendix B. The basic form of those four equations can be written as:

$$\dot{\omega}_{ij} = -\alpha \Psi(\mathbf{e}(t)) \Gamma_j(\mathbf{e}(t)) H_i(\mathbf{c}(t)), \tag{29}$$

where α is a learning rate, $\Psi(\mathbf{e}(t))$ is an operator to measure the error gradient, $\Gamma_j(\mathbf{e}(t))$ quantifies the input activity, and $H_i(\mathbf{c}(t))$ quantifies the postsynaptic activity. In the case of Eqs. (3), and (4) we have $\Psi = 1$ because the input is an error, and the term Γ_j can play the parts of both error gradient and input activity.

From this optic, the rules in this paper are not far from previous forms of node perturbation (Mazzoni, Andersen, & Jordan, 1991; Williams, 1992), reward modulated Hebbian learning (e.g Frémaux, Sprekeler, & Gerstner, 2010; Legenstein, Chase, Schwartz, & Maass, 2010), or Hebbian descent (Melchior & Wiskott, 2019). We went beyond previous approaches in order to deal with complications from continuous-time control with delays. This required using other elements like derivatives, time delays, and normalization.

The rules in Section 2.1 and Appendix B have two obvious drawbacks. One is that the error gradients do not take distal outcomes into account. In other words, the learning rules can only reduce errors that happen soon afterwards (on the order of Δt), but errors that happen later cannot be preemptively corrected. The second drawback is the restriction to monotonic control, since Eq. (29) has no context information beyond the **e** and **c** vectors.

Under this perspective, the actor–critic architecture in Section 3.3, through Eq. (23), improves over Eq. (29) by providing modulation that can handle temporal credit assignment, and can consider a more general context in order to produce an output.

Learning in Eq. (23) can solve the temporal credit assignment problem because of two traits. The first trait is the use of a value function, which considers future rewards when the discount factor is not zero. The second trait is that updates are performed intermittently, only during the transition times. This allows to flexibly span arbitrary lengths of time, but it opens the question

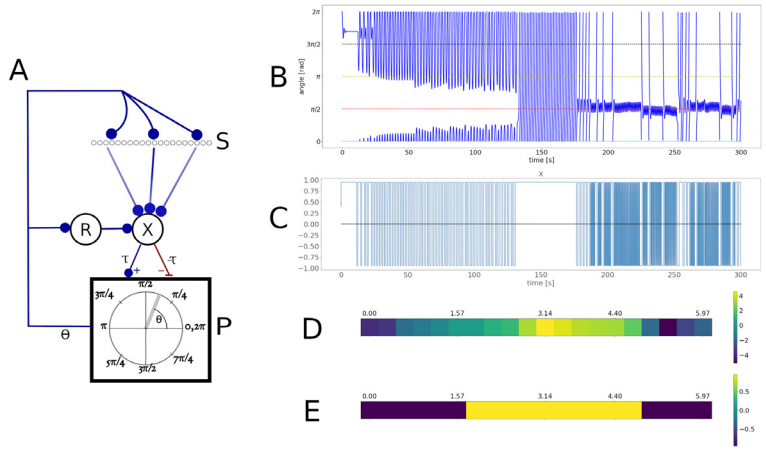

Fig. 9. An architecture for the inverted pendulum problem, and simulation results. (A) The network consists of a population S, plus units X, and R. S represents the angle of the pendulum using 20 units, each with a preferred angle. Unit R outputs the sine of the angle, which is received by unit X as a reward value. Unit X uses reward-modulated Hebbian learning (Eq. (23)) with the right update times to adjust the weights in the connections from the population S. (B) The pendulum's angle through the first 300 s of a simulation. Initially the pendulum oscillates at the bottom, around the $3\pi/2$ angle (black dotted line). Eventually the pendulum manages to complete a revolution, and begins to spin, until it starts to balance at the top, around the $\pi/2$ angle (red dotted line). (C) Output of the X unit, which is proportional to the torque applied. (D) Weights in the connections from S to X after 300 s. Each square corresponds to the weight of a particular unit in S, and the label at the top indicates the preferred angle of that unit. (E) Steady-state output of X when the angle is one of the preferred angles of the 20 units in S. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

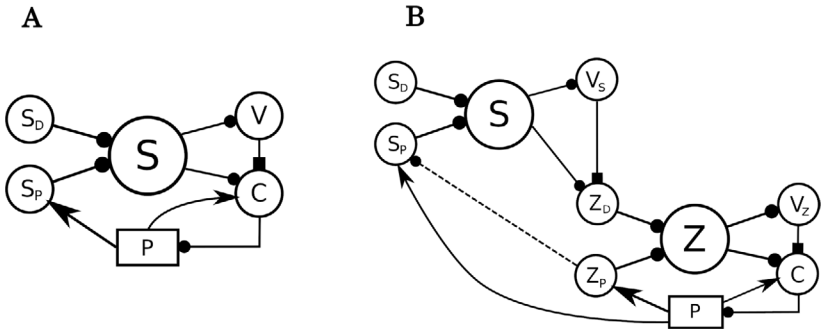

Fig. 10. (A) A reinterpretation of the architecture of Fig. 4 as a 2-level hierarchical control system. Arrowheads denote afferent connections, squares modulatory connections, and circles all other synaptic connections. The population C is considered as a final controller, possibly in the spinal cord. The loop from C to P and back represents a first-level controller using an error representation amenable to negative feedback control. The level on top of this provides the ability to use a distributed representation for the desired and perceived values. (B) A 3-level hierarchy of feedback controllers. A high-level desired perception S_D, together with the current perception S_P are expanded into a high-level state S, which is used to produce a value V_S. This value, the state S, and the current perception S_P can potentially be used to configure a controller lower in the hierarchy, whose target value is expressed by the Z_D population. The dotted connection from Z_P to S_P expresses that the representation in S_P could be constructed using lower level representations rather than state variables of the plant.

of when should the transitions happen. We began to address this question in Section 3.4.

Eq. (23) can handle general context dependencies because the state information is present in layer L, which uses an expansive recoding so that X can approximate arbitrary functions of the state. A possible problem with expansive recoding is that the number of units required scales poorly with the dimension of the input. Other possibilities could include special versions of self-organizing maps (Kohonen, 1995), or a more biological version of the state representations used in deep reinforcement learning. Notice also that the layer L is reminiscent of the sensory maps used in direct inverse learning (Kuperstein, 1988) to associate afferent inputs with muscle activities. L could be seen as a more general version of these maps, also representing desired values, and not necessarily being used to produce muscle activations, but control signals at a higher hierarchical level.

4.2. Hierarchies of feedback controllers

As mentioned in the Introduction, a promising idea on how to generate flexible motor control is to have a hierarchy of feedback controllers that ultimately regulate the value of homeostatic variables for the organism (Powers, 2005). A clear complication is that higher levels of the sensorimotor hierarchy may deal with abstract representations, where an error cannot be obtained by a mere subtraction operation. The architecture we have introduced in Section 3.3 may open the path to exert feedback control with complex representations.

The basic idea of a general feedback controller can be explained with the diagram in panel A of Fig. 10. S_P and S_D can use arbitrary distributed representations, but because these two layer have the same structure we can always detect when their activity is very similar, an event that would produce the reward signal used by V to learn. All of the relevant state information is present in a population S, and this is used by V, as well as by the controller C. The C circle in Fig. 10 is not a unit; it encompasses a feedback controller, and the elements that allow its configuration. In the case of the architecture of Fig. 4 this would include the actor and the X unit. C uses the value provided by V in order to learn its configuration, and the information in S in order to perceive the state.

In our example we set S_D to be the desired activation caused by the target angle in the controller C, but other things could

be encoded in S_D, such as the target in a different coordinate system for a more complex controller. The network comprising the S_D, S_P, S, and V populations could be considered as a separate control system, where V provides a measure of the distance in the activities of S_D and S_P, and this is used either to configure, or to set the target value of the controller C. Learning happens in stages, where the lower-level controllers learn first, and the higher-level controllers perform significant learning after the lower levels can match their target values. In the example of Section 3.3 the feedback controller is already operating while the reinforcement learning system refines its operation, a trait that should be useful for biological organisms.

Configuration of a controller using a value function can have several interpretations. In the example of Section 3.3 this meant selecting the afferent input. Alternatively, this could mean selecting a different controller altogether, which would provide a different implementation of ideas in the MOSAIC-MR model (Sugimoto, Haruno, Doya, & Kawato, 2011), where different RL controllers are used depending on the context. Controller selection has also been suggested as the main role of the basal ganglia (Yin, 2017; Yin & Knowlton, 2006).

Most interestingly, the architecture of Fig. 4, being a feedback controller that configures a feedback controller, naturally has a hierarchical extension, shown in the panel B of Fig. 10. A high-level controller with "S" populations is used to configure a lower level controller with "Z" populations, possibly setting the desired value Z_D. Transforming the pair S_D, S_P into a Z_D value is akin to a coordinate transformation, but in this setting it can also be conceived as a process of subgoal selection. By generating rewards for level S when a "$\mathbf{s}_P = \mathbf{s}_D$" event occurs we can learn a value function for the V_S unit. The output of V_S can be used to modulate plasticity in the descending connections from the S level to the Z level. This last level receives rewards when the "$\mathbf{z}_P = \mathbf{z}_D$" event happens. Having a natural reward function at each level, and the ability to deal with distal rewards gives the model the potential of tackling the problem of finding subgoals, which is common in the hierarchical reinforcement learning literature (e.g. Kulkarni, Narasimhan, Saeedi, & Tenenbaum, 2016; Vezhnevets et al., 2017).

One promising idea is to create sensory representations by grouping states that succeed with similar controller configurations. A direction of future research is thus to use the hierarchical architecture of this model to test whether this controllability criterion can facilitate the formation of perceptual categories.

4.3. The dividends of biological plausibility

Our model suggests a coherent set of hypotheses regarding animal motor control. We outline this below.

- Spinal cord plasticity, and how it coordinates with plasticity at other cortical and subcortical sites is a challenging issue (Brumley, Strain, Devine, & Bozeman, 2018; Norton & Wolpaw, 2018; Wolpaw, 1997). Plasticity rules like those of Section 2.1, if present in the spinal cord, could enable it to become a self-configuring feedback controller. This idea has been suggested before (Raphael, Tsianos, & Loeb, 2010), but a plausible plasticity mechanism has been missing. Furthermore, the model of Section 3.3 shows how plasticity at four different sites can coordinate in a hierarchical manner.
- Some motor control models, such as feedback error learning (Miyamoto et al., 1988), posit that knowledge about sensitivity derivatives $\frac{de}{dc}$ is innate, rather than learned. However, there is significant evidence that some systems recover when the relation between motor command and error is reversed, so $\frac{de}{dc}$ changes sign (Abdelghani & Tweed, 2010; Kuang & Gail, 2015; Lillicrap et al., 2013; Richter et al., 2002; Sachse et al., 2017; Sekiyama, Miyauchi, Imaruoka, Egusa, & Tashiro, 2000; Yamashita et al., 2012). Our model is consistent with this, and it further predicts that in some cases animals may be able learn to use opposite estimates of $\frac{de}{dc}$ depending on the context, but this learning should be much slower, as it depends on a reinforcement learning mechanism (cf. Lillicrap et al., 2013).
- Feedback control is naturally limited by response latencies, and gains that saturate, so a cerebellar module to improve performance is an ideal complement. We emphasize 3 facts: (1) the cerebellum is involved in estimating the timing of events (Bareš et al., 2019), (2) the cerebellum contains predictive signals in the scale of tens of milliseconds (Bareš et al., 2019; Ebner, 2013; Herzfeld, Kojima, Soetedjo, & Shadmehr, 2015; Tanaka et al., 2020; Tseng, Diedrichsen, Krakauer, Shadmehr, & Bastian, 2007), and (3) disynaptic or monosynaptic projections from the cerebellum can be found in spinal cord, as well as cerebral cortex and basal ganglia (Bostan & Strick, 2018; Liang, Paxinos, & Watson, 2011; Middleton & Strick, 1997; Nudo & Masterton, 1989). If the cerebellum relays signals anticipating events at the spinal cord, they could be inputs to the C units in our model, and learning would enable the spinal controller to use these signals to drive anticipated responses. An inverse model in the cerebellum is thus not required for this type of adaptation. On the other hand, supraspinal projections from the cerebellum could be involved in signaling transition times when particular events are anticipated. We thus hypothesize that cerebellar signals to the spinal cord can drive anticipatory responses, and that signals to the cortex and basal ganglia can change the timing of reward-modulated plasticity.
- Using only positive activations and weights that do not change sign motivates the use of dual representations, where the excitation in one neural population caused by a sensorimotor event should come together with inhibition in another population. This is not only consistent with experimental observations (e.g. Najafi et al., 2020; Shafi et al., 2007; Steinmetz, Zatka-Haas, Carandini, & Harris, 2019), but it also permits the function of learning rules as the ones in Section 2.1. When controllable signals exist in antagonistic pairs, it is natural that the activity of a unit does not necessarily produce an action; what matters is the balance between excitation and inhibition. Balance between excitation and inhibition (E/I balance) has received extensive experimental validation, and has been largely recognized as necessary for theoretical models to reproduce observed neuronal dynamics (Dehghani et al., 2016; Haider, Duque, Hasenstaub, & McCormick, 2006; Okun & Lampl, 2008, 2009). Our framework explains why concomitant excitatory and inhibitory responses to sensory events should be prevalent, and links it to the E/I balance using a functional model.

We took all these insights into a more comprehensive model of mammalian arm reaching, where the complexity of the plant and the biological realism of the controller were enhanced (Verduzco-Flores & De Schutter, 2021). While the detailed findings of this follow-up paper are outside the scope of this work, we can briefly mention that using the learning rule in Eq. (4) as a self-configuration mechanism for signals in the spinal cord, we can produce 2D reaching from scratch, and explain the emergence of directional tuning in motor cortex, among other phenomena.

4.4. Comparison with previous work

As mentioned in the Introduction, the closest approach to our work is in Abdelghani et al. (2008). This work required a separate network, and represented the sensitivity derivatives using firing rates. It does not address delays or response latencies, was implemented in discrete time steps, and also controls simple systems (the vestibulo-ocular reflex, and the forearm angle of a 2-joint arm). The learning times are similar to our models.

The review in Kolodziejski et al. (2008a) describes learning rules working in simple open-loop circuits. Two of these learning rules could potentially be compared to our own, namely the ISO (Porr, Ferber, & Wörgötter, 2003), and the ICO (Porr & Wörgötter, 2006) rules. When applied to control problems, both rules begin by assuming that there is an already established feedback control system whose performance is hindered by response delays. Both rules can autonomously learn how to improve the system's performance by applying predictive feedforward responses. The system is thus not learning sensitivity derivatives, or in other words, it does not learn which control signals are capable of reducing particular errors in the MIMO closed-loop setting, which is what our rules achieve. The ISO and ICO rules could thus be used in conjunction with our rules, which would be used to configure the underlying feedback control system.

A similar observation applies to work based on feedback-error learning (Kawato & Gomi, 1992), and on the recurrent architecture (Porrill et al., 2004) as they rely on a previously existing feedback controller whose output is used to train an inverse model. This feedback controller must already have the right input–output structure, or learning will fail. Finding this input–output structure can be done by our learning rules when this is not explicitly specified.

The distal learning approach of Jordan and Rumelhart (1992) does have the potential to fully perform controller configuration, but this relies on backpropagating an error signal through a forward model, which strains biological plausibility.

There is also a relatively large number of neurobiomechanical models that perform simple motor tasks. In general they are not relevant here due to one or more of the following reasons:

1. They do not address the problem of input–output configuration (e.g. finding sensitivity derivatives), or control a single degree of freedom, which sidesteps this problem.
2. Use non-neural systems to produce motor commands.
3. Do not model a biologically plausible form of synaptic learning.

For these reasons the approach we presented towards motor learning may be the most capable yet, in its ability to self-configure actuators while still respecting a large amount of biological constraints. Moreover, there is a clear vision on how to extend this model so it can tackle more complex tasks and controllers.

5. Conclusion

In this paper we have introduced the main ideas required for a class of motor control models that maintain a large degree of biological plausibility, while still being capable of performing non-trivial tasks. There are 3 key characteristics that make this possible: a feedback control architecture using dual excitatory–inhibitory representations, synaptic rules that find the direction of sensitivity derivatives, and a critic component that configures the controller using reinforcement learning mechanisms. The fact that these models have hierarchical extensions that could potentially be used to control homeostatic variables opens the possibility of our ideas producing highly adaptable autonomous agents. We will work towards this goal.

Supplementary material

The source code for this paper can be obtained from: https://gitlab.com/sergio.verduzco/public_materials in the `synaptic_approach` folder.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors want to thank Prof. Kenji Doya for numerous and helpful comments to initial versions of this manuscript.

Appendix A. Analogy with the Relative Gain Array criterion.

When presenting Eq. (3) in Section 2.1 it was mentioned that this has similarities to the Relative Gain Array (RGA) criterion. We explain that comment.

Assume a Multi-Input Multi-Output (MIMO) system where the plant is M-dimensional, and the controller has an N-dimensional output. Further assume that we want to create a decentralized control system, consisting of N individual feedback loops. In a control system like the one in Fig. 1 of the main text, the problem we face is knowing which controller should be assigned to control each state variable. Since control loops will be interacting with each other, performance will be degraded, but a good loop configuration (also called input/output selection) can largely attenuate this.

The RGA criterion (Bristol, 1966) offers a measure of the interaction between control variables and plant outputs (or in our case, elements of the error vector) that, among other things, has the desirable property of scale invariance. Consider a linearized, time-invariant control system $\dot{\bar{y}} = A\bar{y} + B\bar{u}$, where $\bar{u}$ is the N-dimensional control vector, and $\bar{y}$ is the M-dimensional observed plant output. To simplify the presentation we use a 2×2 system:

$$\begin{bmatrix}\dot{y}_1\\ \dot{y}_2\end{bmatrix} = \begin{bmatrix}a_{11} & a_{12}\\ a_{21} & a_{22}\end{bmatrix}\begin{bmatrix}y_1\\ y_2\end{bmatrix} + \begin{bmatrix}b_{11} & b_{12}\\ b_{21} & b_{22}\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix}.$$

By assumption, the system is stable for constant $\bar{u}^*$ controls, so that at a fixed point we have $A\bar{y}^* + B\bar{u}^* = 0$. We may thus write:

$$\bar{y}^* = A^{-1}B\bar{u}^* \equiv K\bar{u} = \begin{bmatrix}k_{11} & k_{12}\\ k_{21} & k_{22}\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix},$$

where K is a steady-state gain matrix. The RGA method uses K to produce a matrix Λ whose entries are defined to be:

$$\lambda_{i,j} = \frac{(\Delta y_i/\Delta u_j)_{\Delta u_j}}{(\Delta y_i/\Delta u_j)_{\Delta y_i}}.$$

$\lambda_{i,j}$ is a measure of the interaction between y_i and u_j, arising from the ratio of two gains. The gain $(\Delta y_i/\Delta u_j)_{\Delta u_j}$ is $(\Delta y_i/\Delta u_j)$ when $\Delta u_l = 0$ for $l \neq j$. In other words, this gain is produced from the plant's outputs when Δu_j is the only non-zero perturbation. $(\Delta y_i/\Delta u_j)_{\Delta y_i}$ is $(\Delta y_i/\Delta u_j)$ when $\Delta y_l = 0$ for $l \neq i$. For example, to find $(\Delta y_1/\Delta u_1)_{\Delta u_1}$ we set the equation:

$$\begin{bmatrix}y_1^* + \Delta y_1\\ y_2^* + \Delta y_2\end{bmatrix} = \begin{bmatrix}k_{11} & k_{12}\\ k_{21} & k_{22}\end{bmatrix}\begin{bmatrix}u_1^* + \Delta u_1\\ u_2^*\end{bmatrix},$$

finding that $\Delta y_1 = k_{11}\Delta u_1$, so $(\Delta y_1/\Delta u_1)_{\Delta u_1} = k_{11}$.

To find $(\Delta y_1/\Delta u_1)_{\Delta y_1}$ we set

$$\begin{bmatrix}y_1^* + \Delta y_1\\ y_2^*\end{bmatrix} = \begin{bmatrix}k_{11} & k_{12}\\ k_{21} & k_{22}\end{bmatrix}\begin{bmatrix}u_1^* + \Delta u_1\\ u_2^* + \Delta u_2\end{bmatrix}.$$

Some simple algebra shows that $\Delta y_1 = \left(k_{11} - \frac{k_{12}k_{21}}{k_{22}}\right)\Delta u_1$. Therefore $\lambda_{1,1} = k_{11}\left(k_{11} - \frac{k_{12}k_{21}}{k_{22}}\right)^{-1}$. It is easy to show that, in general: $\Lambda = K \otimes (K^{-1})^T$, where $\otimes$ is the element-by-element product. It is not difficult to prove that the rows and columns of Λ add to one. Moreover, Λ is invariant to scaling of the gain in any controller, and permutation of the controllers only causes the same permutation in Λ. Some stability properties of the controller can be proven when integral action dominates, but these are not the focus of the current exposition.

Returning to our 2×2 example, we had calculated $\lambda_{1,1} = k_{11}\left(k_{11} - \frac{k_{12}k_{21}}{k_{22}}\right)^{-1}$. The appearance of k_{11} is simple to interpret: it is the ratio of the reaction Δy_i divided by the perturbation Δu_j, as implied by the steady state gain matrix. This ratio of reaction to perturbation could be captured in a learning rule where $\dot{\omega}_{ij} = -\alpha \dot{e}_i(t)\dot{u}_j(t - \Delta t)$. This would work if controllers did not interact (e.g. columns of K only had a single non-zero element), but in general the action of one controller may disrupt the action of the others.

To handle interaction among controllers the RGA criterion considers the vector $\bar{c}_i$ that is orthogonal to every row of K, save for the ith one. If we wanted to control y_i without perturbing any other variable, then a control output along the direction of $\bar{c}_i$ could do this. The term $\left(k_{11} - \frac{k_{12}k_{21}}{k_{22}}\right)$ is the value of the first entry in $\bar{c}_i$ for the 2×2 case. It would be ideal if this value was of the same magnitude as k_{11}. In general, values of $\lambda_{j,k} \gg 1$ are a sign that the kth controller would cause excessive interference if used to control the jth variable, whereas $\lambda_{j,k} \ll 1$ indicates that this controller has little effect on y_j.

It is not obvious how to calculate $(\Delta y_i/\Delta u_j)_{\Delta y_i}$ using a biologically-plausible network. Instead, we could approximate $\lambda_{j,k}$ by making the weight of the connection from e_j to c_k increase according to how much e_j changes in reaction to c_k, but downgrade or upgrade this increase according to how much the other controllers are also changing e_j. This is the aim of the synaptic competition introduced in Eq. (3) of the main text.

Appendix B. Alternative learning rules for monotonic control

The rules we derive here have a Hebbian-like form where the synaptic weight ω_{ij} for the connection from e_j to c_i has a time derivative:

$$\dot{\omega}_{ij}(t) = -\alpha G_j(\mathbf{e}(t))H_i(\mathbf{c}(t)), \tag{B.1}$$

where α is a learning rate, and G_j, H_i are delay-differential operators. In this appendix we present two more equations of this type, and show a test of their performance.

For a different approach to produce a learning rule, consider using some form of reinforcement learning in order to train the controller of Fig. 2. We have to consider that the method we choose has to act in continuous time, learn on-policy (e.g. as it performs its task), and result in the adjustment of the ω_{ij} weights.

As a first consideration, providing rewards only when $S_D = S_P$ could result in very slow learning, so a form of reward shaping is desirable. For this purpose we can use $\|\mathbf{e}\|$ as a measure of distance to the target, which can be a negative reward. Training individual synapses can be addressed by a policy gradient method, with synaptic weights being the parameter, and presynaptic rates being the state. The weight perturbation method (Werfel, Xie, & Seung, 2005) uses this logic: a perturbation in the ω_{ij} weights causes a change in the error, which allows to estimate the gradient of the error with respect to the weights. As pointed out in Werfel et al. (2005), weight perturbation can be much slower than node perturbation, which can still be relatively slow when used to find sensitivity derivatives (Abdelghani et al., 2008).

To explain the node perturbation scheme (as in the REINFORCE framework, Williams, 1992), consider a linear system with M-dimensional inputs x, and N dimensional outputs y, related by an N × M weight matrix W, so that $y = Wx$. For each input x we have a desired output d, and we assume that there is a teacher matrix W^* such that $d = W^*x$. The error function is $E = \frac{1}{2}|y - d|^2 = \frac{1}{2}|(W - W^*)x|^2 = \frac{1}{2}|\Delta W x|^2$, where $\Delta W \equiv W - W^*$.

Node perturbation consists of adding noise to the outputs y so we can get a new error E'_{NP}, and then we change the weights following the gradient of that error. More precisely, let ξ be an N-dimensional vector drawn from a Gaussian distribution with 0 mean and variance σ^2. Define $E'_{NP} = \frac{1}{2}|\Delta W x + \xi|$. Weights are changed according to $\Delta W_{NP} = -\frac{\alpha}{\sigma^2}(E'_{NP} - E)\xi x^T$.

A problem that comes with an on-policy, continuous-time implementation of this would be to produce the same inputs twice so we can observe the error gradient $(E'_{NP} - E)$ using errors with and without the ξ output perturbation. The scheme we propose is to use $\dot{c}$ as a proxy for ξ, and $\frac{d\|\mathbf{e}(t)\|}{dt} \equiv \|\mathbf{e}(t)\|'$ as a proxy for $(E'_{NP} - E)$, leading to a rule like:

$$\dot{\omega}_{ij} = -\alpha\|\mathbf{e}(t)\|'\dot{c}_i(t - \Delta t)e_j(t - \Delta t).$$

Simulations show that this rule is still not effective. In the first place, the e_j inputs are always positive, which is unlike node perturbation in the REINFORCE framework. This can be addressed by using the term $(e_j(t-\Delta t) - \langle \mathbf{e} \rangle)$ instead, where $\langle \mathbf{e} \rangle = \sum_k e_k(t - \Delta t)$. Secondly, this rule tends to produce much better results when heterosynaptic competition is also introduced for the $\dot{c}_i$ term, in as in the previous cases. The rule we will test in this paper is thus:

$$\dot{\omega}_{ij} = -\alpha\|\mathbf{e}(t)\|'\big(\dot{c}_i(t - \Delta t) - \langle \dot{c} \rangle\big)\big(e_j(t - \Delta t) - \langle \mathbf{e} \rangle\big). \tag{B.2}$$

We will derive one final rule. To understand it we must first consider that the units in population C may act as integrators of their input, a design that is justified in Appendix C, but can also be understood from the following discussion. We will write simplified equations for the system of Fig. 1. Assume that the plant P is linear, with an output $\mathbf{p} = W_P\mathbf{c}$. Let the output of the C population consist of the vector $\int W(\mathbf{s}_D(\xi) - \mathbf{s}_P(\xi))d\xi$, where W is a matrix of synaptic weights, for which we want to find a learning rule. If we assume the transmission delays and latencies of the system can be absorbed into the response latency of the S_P population with dynamics $\tau_s\dot{\mathbf{s}}_P(t) = \mathbf{p} - \mathbf{s}_P$, the simplified system's equations can be written as:

$$\tau_s\dot{\mathbf{s}}_P(t) = W_P\int_0^t W(\mathbf{s}_D(\xi) - \mathbf{s}_P(\xi))d\xi - \mathbf{s}_P(t), \tag{B.3}$$

$$\tau_w\dot{W}(t) = G(\mathbf{s}_D(t) - \mathbf{s}_P(t))H\left(\int_0^t W(\mathbf{s}_D(\xi) - \mathbf{s}_P(\xi))d\xi\right), \tag{B.4}$$

where G, H are the matrix versions of the operators in Eq. (B.1). We would like to have a stable fixed point such that $\mathbf{s}_D = \mathbf{s}_P$ (looking at Eq. (B.3), this fixed point may not make sense without integration in the units of C). This implies there is a time t^* so that $\mathbf{e}(t^*) = \mathbf{s}_D(t^*) - \mathbf{s}_P(t^*) \approx \mathbf{0}$. Stability of this fixed point should imply that if there is a perturbation $\delta\mathbf{e}$ away from the fixed point $\mathbf{e} = \mathbf{0}$, then the control signal would move $\mathbf{e}(t)$ back towards $\mathbf{0}$, which would be the case if $\dot{\mathbf{e}}(t + \Delta t) \approx -\delta\mathbf{e}$ for some small enough Δt. A different way to state this condition is that the vector $\mathbf{c}(t) = \int_0^t W(\xi)\mathbf{e}(\xi)d\xi$ is in the space generated by all eigenvectors of W_P with negative eigenvalues. Assuming $\mathbf{e}(0) = \mathbf{0}$, it is necessary that $W(\xi)\mathbf{e}(\xi)$ is also in this negative eigenspace for most values of $\xi \in (0, t)$.

In short, we want to modify $W(t)$ so that $W_PW(t)\mathbf{e}(t)$ aligns with $-\mathbf{e}(t)$. A measure of that alignment can come from the inner product $\mathbf{e}(t) \cdot \dot{\mathbf{e}}(t + \Delta t)$, where Δt is roughly the time it takes the $\mathbf{e}$ signal to go through a loop in the feedback system, causing

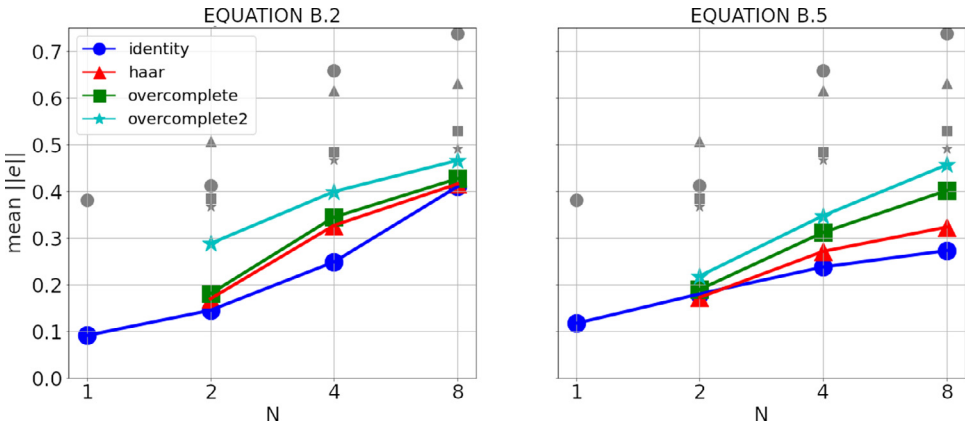

Fig. B.11. Simulation results for 4 types of connectivity matrices using the two learning rules of this section. The format of this figure is the same as of Fig. 6 in the main text.

a change $\dot{\mathbf{e}}$. In practice it is better to use $\mathbf{e}(t+\Delta t)\cdot\dot{\mathbf{e}}(t+\Delta t)$, which states that we want the controller response to correct the $\mathbf{e}(t+\Delta t)$ error, rather than the outdated $\mathbf{e}(t)$ error. Ideally we would like to have this inner product close to its minimum value $-\|\mathbf{e}\|\|\dot{\mathbf{e}}\|$. On the other hand, when the inner product is positive, we want to change W so it produces the opposite change.

Assume that an error signal $\mathbf{e}(t)$ causes a controller response $\dot{\mathbf{c}}(t+\Delta_0 t)$, and this activity eventually creates a change in the error signal with rate $\dot{\mathbf{e}}(t+\Delta_2 t)$. When the inner product $\mathbf{e}(t+\Delta_2 t)\cdot\dot{\mathbf{e}}(t+\Delta_2 t)$ is positive, we can attempt to reduce it by subtracting the outer product $\dot{\mathbf{c}}(t+\Delta_0)\mathbf{e}^T(t)$ from W, leading to the following rule:

$$\tau_\omega\dot{\omega}_{ij}(t) = -\alpha\left[\mathbf{e}(t)\cdot\dot{\mathbf{e}}(t)\right]\dot{\mathbf{c}}_i(t-\Delta_1 t)\mathbf{e}_j(t-\Delta_2 t).$$

In this equation the time shifts were made negative to avoid using future values. $\Delta_1 t$ is the time it takes for the activity in C to cause a change in the input to C, and $\Delta_2 t$ is $\Delta_1 t$ plus the response latency in C. The result is similar to Eq. (B.2) with a different measure of the error gradient. As before, from the activities in $\dot{\mathbf{c}}$ and $\mathbf{e}$ we can subtract the mean values $\langle\dot{\mathbf{c}}\rangle = \sum_k \dot{c}_k$, and $\langle\mathbf{e}\rangle = \sum_k e_k$:

$$\tau_\omega\dot{\omega}_{ij}(t) = -\alpha\left[\mathbf{e}(t)\cdot\dot{\mathbf{e}}(t)\right](\dot{\mathbf{c}}_i(t-\Delta_1 t)-\langle\dot{\mathbf{c}}\rangle)(\mathbf{e}_j(t-\Delta_2 t)-\langle\mathbf{e}\rangle). \quad \text{(B.5)}$$

The rules in Eqs. (B.2) and (B.5) were tested with the same linear plant as used in Section 3.1, Fig. 6. Results can be observed in Fig. B.11.

The worst performance is obtained with Eq. (B.2), which scales poorly to larger values of N. We still have to pinpoint the exact cause of this, although we do not discard that different parameters could change the outcome. This rule is included because it is the best adaptation we have found of an established RL method that can be used within our framework. A general understanding of why performance degrades will require a comprehensive convergence analysis.

Appendix C. Non-convergence of a simple linear model

In Section 3.1 of the main text it is stated that using the architecture of Fig. 1, together with linear units and plastic synapses as in Eq. (3) will lead to a network that can converge to states with non-zero error. This is shown next for the first learning rule of Section 2.1.

Consider the system of Fig. 1, with the first plant of Section 3.1. Namely, each unit c_j of the controller has a vector $\mathbf{v}_j$ associated with it, and the output of the plant is $\mathbf{p} = \sum_k c_k\mathbf{v}_k$, where c_k is also used to denote the activity of the kth unit. We define $\mathbf{V}$ as the M × N matrix whose N columns are the $\mathbf{v}_j$ vectors, so we may write $\mathbf{p} = \mathbf{Vc}$.

We assume that S_P has an M-dimensional activity vector $\mathbf{s}_p = \mathbf{p}(t-d)$, where d incorporates the transmission delays. S_{DP} activity is a 2M-dimensional vector with elements $s_j^{DP} = \text{sgn(j)}(s_j^D - s_j^P)$, where sgn(j) = 1 for $j \le M$, and sgn(j) = −1 for $j > M$. In vector notation this can be written $\mathbf{s}_{DP} = \mathbf{s}_{D(2)} - \mathbf{W}\mathbf{s}_P$, where $\mathbf{s}_{D(2)}$ is a 2M column vector with two stacked copies of $\mathbf{s}_D$, and $\mathbf{W}$ is a 2M × M matrix consisting of the M × M identity matrix stacked on top of its negative.

We also define $\boldsymbol{\Omega}$ to be the N × 2M matrix of connections from S_{DP} to C.

The system has the following equations

$$\tau_C\dot{c}_j(t) = \left(\sum_k \omega_{jk}s_k^{DP}(t-d_1)\right) - c_j(t),$$

$$\tau_\omega\dot{\omega}_{jk}(t) = (\dot{c}_j(t-d_2) - \langle\dot{\mathbf{c}}(t-d_2)\rangle)\left(\dot{s}_k^{DP}(t-d_1) - \langle\dot{\mathbf{s}}_{DP}(t-d_1)\rangle\right)$$

which in vector notation become:

$$\tau_C\dot{\mathbf{c}}(t) = \boldsymbol{\Omega}\mathbf{s}_{DP}(t-d_1) - \mathbf{c}(t), \quad \text{(C.1)}$$

$$\tau_\omega\dot{\boldsymbol{\Omega}}(t) = \left(\mathbf{I_N} - \frac{1}{N}\mathbf{1_N}\right)\dot{\mathbf{c}}(t-d_2) \times \left[\left(\mathbf{I_{2N}} - \frac{1}{2M}\mathbf{1_{2M}}\right)\dot{\mathbf{s}}_{DP}(t-d_1)\right]^T, \quad \text{(C.2)}$$

where $\mathbf{I_N}$ is the N × N identity matrix, $\mathbf{1_N}$ is the N × N matrix where all entries are 1, $\mathbf{I_{2M}}$ is a M × 2M matrix of the form $[\mathbf{I_M}\mathbf{I_M}]$, and $\mathbf{1_{2M}}$ is an M × 2M matrix of the form $[\mathbf{1_M}\mathbf{1_M}]$.

Eq. (C.2) is proportional to derivatives on both sides, and will vanish in steady state. Also, from Eq. (C.1) it is evident that if $\mathbf{s}_{DP} = \mathbf{0}$ in the steady state, this implies $\mathbf{c} = \mathbf{0}$, which in turn implies $\mathbf{s}_D = \mathbf{0}$. Clearly this is not a general solution.

The actual fixed point can be found by replacing $\mathbf{s}_{DP}$ with $\mathbf{s}_{D(2)} - \mathbf{WVc}$ in Eq. (C.1):

$$\tau_C\dot{\mathbf{c}}(t) = \mathbf{0} = \boldsymbol{\Omega}\mathbf{s}_{D(2)}(t-d_1) - (\boldsymbol{\Omega}\mathbf{WV} - \mathbf{I_N})\,\mathbf{c}(t),$$

so $\mathbf{c} = (\boldsymbol{\Omega}\mathbf{WV} - \mathbf{I_N})^{-1}\boldsymbol{\Omega}\mathbf{s}_{D(2)}$. Whether this fixed point is attractive depends on the eigenvalues of the system of Eqs. (C.1), (C.2), where (C.1) is used to write $\dot{\mathbf{c}}$ and $\dot{\mathbf{s}}_{DP}$ in terms of $\mathbf{c}$. This analysis, however, would provide little further insight.

One final point is that Eq. (C.2) shows that homogeneous derivatives will cause similar changes for all weights, reducing learning in the network. It is thus necessary to avoid synchronization, which is aided by the use of heterogeneous parameters for the sigmoidal units, as well as heterogeneous oscillation frequencies for the controllers (see Methods).

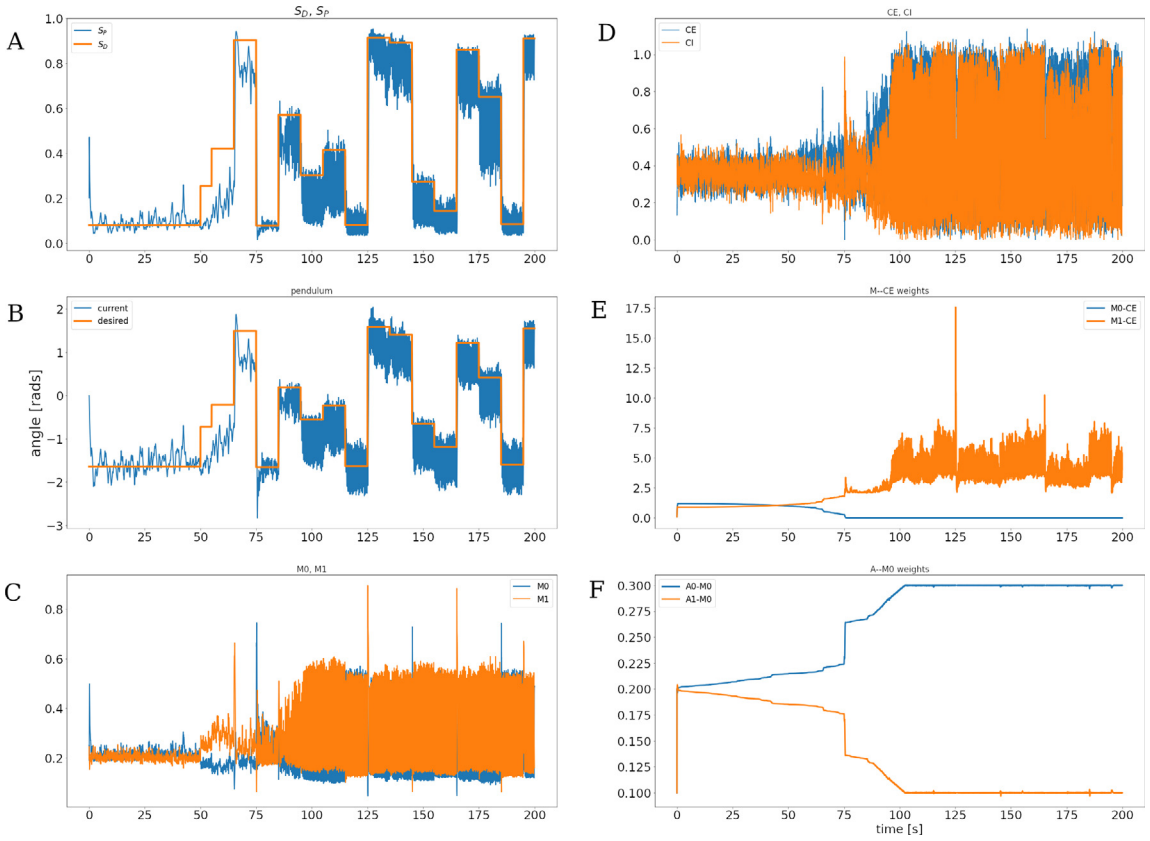

Fig. D.12. First 200 s of a simulation where the architecture of Fig. 3 is used so a pendulum can track a desired angle (gravity is present). (A) Activity of the S_P unit, with the perceived angle, and S_D, with the desired value for S_P. (B) Angle of the pendulum, and the desired angle. (C) Activities of the two units in population M. (D) Activities of the two units in population C. (E) Synaptic weights for the connections from the two M units to the CE unit. (F) Synaptic weights for the connections from the two A units to one of the M units.

Appendix D. Simulations of the pendulum with gravity

Simulation of the systems in Section 3.2 was done when the pendulum experienced gravity. All other parameters in the system were identical, with the exception of the input gain of the plant, which was increased (see Appendix E).

Fig. D.12 is the analog of Fig. 7, but simulated with gravity.

The gravity force accelerates the pendulum towards the angle $-\pi/2$. Target angles near this value require a lower gain in order to be reached, whereas targets near 0 and π require a larger gain. As explained in Section 3.2, the system is driven by error, and as the error decreases the torque produced is not sufficient to fully reach the target. It is for this reason that for certain targets the system presents a larger steady-state error.

The gain of the system is closely related to the slope in the sigmoidal activation functions for the units in the feedback loop. Steep slopes will produce high gain, but reduce the dynamic range of the system. In other words, when the slope is steep, large or small angles will produce values very close to 1 or 0 respectively, and angle differences in these ranges will not be perceived, leading to imprecise responses. This is a challenge that is often not addressed by non-neural models.

A related problem with the system in Section 3.2 is that for certain initial conditions the pendulum will initially rotate close to π radians until the forces that restrict pendulum rotation bring it to a stop. At this point movement of the pendulum is very limited, slowing down learning. This is compounded by the fact that close to π the angle is at a range where, due to the sigmoidal activation functions, oscillation amplitudes are greatly reduced. Both of these factors can make learning extremely slow, and leave the pendulum "stuck" at π or $-\pi$ radians.

Certain parameter regimes can be used to avoid this problem. In particular, very fast learning rates in the connections from M to C, large plant input gains, and desired values that change slowly, all contribute to make the arm avoid getting stuck at a limit angle. This motivated some of the parameter selection, and for this reason the first target presentation lasts 50 s, whereas the subsequent targets are presented for 10 s. It should be noticed that this problem is a particular trait of using a pendulum as our test system, and is probably not relevant in a biological setting.

When the pendulum is not constrained in its rotation this problem once again emerges, although with a different form. When the pendulum crosses π radians the s_P value experiences a sudden shift between 0 and 1. For certain initial conditions, crossing π causes the error to change its sign, making the pendulum return to π, again changing the sign of the error. The result is that the pendulum oscillates around π radians indefinitely. The same parameter regimes as before can help avoid this problem, which is once more irrelevant for biological learning.

Appendix E. Parameter values

Note: for parameters with heterogeneous values, the reported value is the one before noise is added. All dual populations use the same parameters.

For the learning equations in Section 2.1:

Parameter	Equations	Sections	Value
Δt	(3), (4), (B.2), (B.5)	All	140 [ms]
α	(3), (4), (B.2), (B.5)	3.1	.15
	(4)	3.2	2.5
		3.3	.5
λ	(3), (6)	All	0.05
	(4), (6)	All	0.03
τ_f	(7)	3.1, 3.2 and 3.3	10 [ms]
		3.1 for $\dot{c}_j$; 3.2 for $\dot{I}_{DP}$; 3.3 for $\dot{I}_{DP}$	5 [ms]
τ_s	(7)	3.1, 3.2 and 3.3	50 [ms]
		3.1 for $\dot{e}_j$	200 [ms]

For the model in Section 3.1:

Parameter	Equation	Population	Value
τ_s	(2)	S_P, S_{PD}	50 [ms]
β	(2)	S_P	1
		S_{PD}	4
η	(2)	S_P	0
		S_{PD}	0.4
τ_x	(9)	CE, CI	200 [ms]
τ_c	(10)	CE, CI	200 [ms]
τ_p	(11)	P	50 [ms]

For the model in Section 3.2:

Parameter	Equation	Population	Value
τ_s	(2)	CE, CI, S_P, S_{PD}	20 [ms]
		M	10 [ms]
β	(2)	CE, CI	2
		M	2.5
		S_P	1.5
		S_{PD}	5
η	(2)	CE, CI	0.2
		M, S_{PD}	0.5
		S_P	0
τ_a	(17)	A	10 [ms]
T	(17)	A	0
α_{IC}	(16)	—	0.025

For the model in Section 3.3:

Parameter	Equation	Population	Value
τ_s	(2)	$CE, CI, S_P, S_{PD}, V, X$	20 [ms]
		M	10 [ms]
β	(2)	CE, CI, S_P	2
		M	2.5
		S_{PD}, X	5
		V	1.5
η	(2)	CE, CI	0.2
		M, S_{PD}	0.5
		S_P	0
		V, X	0
τ_V	(19)	V	20 [ms]
τ_X	(20)	X	20 [ms]
α_V	(21)	—	0.005
Δt_v	(21)	—	3 [s]
γ	(21)	—	0.6
α_X	(23)	—	0.15
η_X	(24)	—	0.2
τ_a	(17)	A	10 [ms]
T	(17)	A	0
α_{IC}	(16)	—	0.025
τ_P	(18)	S_P^*	10 [ms]
b	(26)	V, X	1.59
η_1	(22)	V, X	0.1
η_2	(22)	V, X	0.005

For the model in Section 3.4:

Parameter	Equation	Population	Value
τ_X	(27)	X	20 [ms]
β	(27)	X	5
α_X	(23)	—	0.4
η_X	(24)	—	0.1
b	(25)	X	5

For the models in Section 3.2, Section 3.3, and 3.4, the plant was a homogeneous pendulum with mass 1 [kg], and length 0.5 [m]. The viscous friction coefficient was 1 [kg · m^2/s], except for Section 3.4, where the value 0.2 [kg · m^2/s] was used. When gravity is present, its value is 9.81 [m/s^2].

For the model in Section 3.2, the gain was 4, meaning that an input of magnitude one would produce a torque of 4 [kg· m^2/s^2]. The equivalent simulation in Appendix D used a gain of 7. Section 3.3 used a gain of 2, and Section 3.4 used a gain of 1.5.

References

Abdelghani, M. N., Lillicrap, T. P., & Tweed, D. B. (2008). Sensitivity derivatives for flexible sensorimotor learning. *Neural Computation*, *20*(8), 2085–2111. http://dx.doi.org/10.1162/neco.2008.04-07-507, URL https://www.mitpressjournals.org/doi/10.1162/neco.2008.04-07-507.

Abdelghani, M. N., & Tweed, D. B. (2010). Learning course adjustments during arm movements with reversed sensitivity derivatives. *BMC Neuroscience*, *11*(1), 150. http://dx.doi.org/10.1186/1471-2202-11-150.

Andersen, R. A., Essick, G. K., & Siegel, R. M. (1985). Encoding of spatial location by posterior parietal neurons. *Science*, *230*(4724), 456–458. http://dx.doi.org/10.1126/science.4048942, URL https://science.sciencemag.org/content/230/4724/456.

Bareš, M., Apps, R., Avanzino, L., Breska, A., D'Angelo, E., Filip, P., et al. (2019). Consensus paper: decoding the contributions of the cerebellum as a time machine. From neurons to clinical applications. *The Cerebellum*, *18*(2), 266–286. http://dx.doi.org/10.1007/s12311-018-0979-5.

Bastian, A. J., Martin, T. A., Keating, J. G., & Thach, W. T. (1996). Cerebellar ataxia: Abnormal control of interaction torques across multiple joints. *Journal of Neurophysiology*, *76*(1), 492–509. http://dx.doi.org/10.1152/jn.1996.76.1.492, URL https://journals.physiology.org/doi/abs/10.1152/jn.1996.76.1.492.

Bernstein, N. (1967). *The co-ordination and regulation of movements*. Pergamon Press, Google-Books-ID: MUhzjwEACAAJ.

Bizzi, E., & Ajemian, R. (2020). From motor planning to execution: A sensorimotor loop perspective. *Journal of Neurophysiology*, *124*(6), 1815–1823. http://dx.doi.org/10.1152/jn.00715.2019, URL https://journals.physiology.org/doi/full/10.1152/jn.00715.2019.

Bostan, A. C., & Strick, P. L. (2018). The basal ganglia and the cerebellum: Nodes in an integrated network. *Nature Reviews Neuroscience*, *19*(6), 338–350. http://dx.doi.org/10.1038/s41583-018-0002-7, URL https://www.nature.com/articles/s41583-018-0002-7.

Bristol, E. (1966). On a new measure of interaction for multivariable process control. *IEEE Transactions on Automatic Control*, *11*(1), 133–134. http://dx.doi.org/10.1109/TAC.1966.1098266.

Brumley, M. R., Kauer, S. D., & Swann, H. E. (2015). Developmental plasticity of coordinated action patterns in the perinatal rat. *Developmental Psychobiology*, *57*(4), 409–420. http://dx.doi.org/10.1002/dev.21280, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/dev.21280. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/dev.21280.

Brumley, M. R., Strain, M. M., Devine, N., & Bozeman, A. L. (2018). The spinal cord, not to be forgotten: The final common path for development, training and recovery of motor function. *Perspectives on Behavior Science*, *41*(2), 369–393. http://dx.doi.org/10.1007/s40614-018-00177-9.

Capaday, C., Forget, R., & Milner, T. (1994). A re-examination of the effects of instruction on the long-latency stretch reflex response of the flexor pollicis longus muscle. *Experimental Brain Research*, *100*(3), 515–521. http://dx.doi.org/10.1007/BF02738411.

Dean, P., & Porrill, J. (2008). Adaptive-filter models of the cerebellum: Computational analysis. *The Cerebellum*, *7*(4), 567–571. http://dx.doi.org/10.1007/s12311-008-0067-3, URL http://link.springer.com/article/10.1007/s12311-008-0067-3.

Dehghani, N., Peyrache, A., Telenczuk, B., Le Van Quyen, M., Halgren, E., Cash, S. S., et al. (2016). Dynamic balance of excitation and inhibition in human and monkey neocortex. *Scientific Reports*, *6*(1), Article 23176. http://dx.doi.org/10.1038/srep23176, URL https://www.nature.com/articles/srep23176.

Ebner, T. J. (2013). Cerebellum and internal models. In M. Manto, J. D. Schmahmann, F. Rossi, D. L. Gruol, & N. Koibuchi (Eds.), *Handbook of the cerebellum and cerebellar disorders* (pp. 1279–1295). Springer Netherlands, URL http://link.springer.com/referenceworkentry/10.1007/978-94-007-1333-8_56.

Frémaux, N., Sprekeler, H., & Gerstner, W. (2010). Functional requirements for reward-modulated spike-timing-dependent plasticity. *Journal of Neuroscience*, *30*(40), 13326–13337. http://dx.doi.org/10.1523/JNEUROSCI.6249-09.2010, URL http://www.jneurosci.org/content/30/40/13326.

Goulding, M., Bourane, S., Garcia-Campmany, L., Dalet, A., & Koch, S. (2014). Inhibition downunder: An update from the spinal cord. *Current Opinion in Neurobiology*, *26*, 161–166. http://dx.doi.org/10.1016/j.conb.2014.03.006, URL http://www.sciencedirect.com/science/article/pii/S0959438814000567.

Hadjiosif, A. M., Krakauer, J. W., & Haith, A. M. (2021). Did we get sensorimotor adaptation wrong? implicit adaptation as direct policy updating rather than forward-model-based learning. *Journal of Neuroscience*, *41*(12), 2747–2761. http://dx.doi.org/10.1523/JNEUROSCI.2125-20.2021, URL https://www.jneurosci.org/content/41/12/2747.

Haider, B., Duque, A., Hasenstaub, A. R., & McCormick, D. A. (2006). Neocortical network activity in vivo is generated through a dynamic balance of excitation and inhibition. *Journal of Neuroscience*, *26*(17), 4535–4545. http://dx.doi.org/10.1523/JNEUROSCI.5297-05.2006, URL http://www.jneurosci.org/cgi/doi/10.1523/JNEUROSCI.5297-05.2006.

Hamburger, V. (1973). Anatomical and physiological basis of embryonic motility in birds and mammals. In G. Gottlieb (Ed.), *Behavioral embryology*: *Vol. 1*, *Studies on the development of behavior and the nervous system* (pp. 51–76). Elsevier, http://dx.doi.org/10.1016/B978-0-12-609301-8.50009-X, URL https://www.sciencedirect.com/science/article/pii/B978012609301850009X.

Hayashibe, M., & Shimoda, S. (2014). Synergetic motor control paradigm for optimizing energy efficiency of multijoint reaching via tacit learning. *Frontiers in Computational Neuroscience*, *8*, 21. http://dx.doi.org/10.3389/fncom.2014.00021, URL http://journal.frontiersin.org/Journal/10.3389/fncom.2014.00021/abstract.

Helmchen, F. (1999). Dendrites as biochemical compartments. In *Dendrites* (p. 376). Oxford, England: Oxford University Press, URL https://pure.mpg.de/pubman/faces/ViewItemOverviewPage.jsp?itemId=item_2537436.

Herzfeld, D. J., Kojima, Y., Soetedjo, R., & Shadmehr, R. (2015). Encoding of action by the Purkinje cells of the cerebellum. *Nature*, *526*(7573), 439–442. http://dx.doi.org/10.1038/nature15693, URL https://www.nature.com/articles/nature15693.

Illing, B., Gerstner, W., & Brea, J. (2019). Biologically plausible deep learning — But how far can we go with shallow networks? *Neural Networks*, *118*, 90–101. http://dx.doi.org/10.1016/j.neunet.2019.06.001, URL http://www.sciencedirect.com/science/article/pii/S0893608019301741.

Izhikevich, E. M. (2000). Neural excitability, spiking and bursting. *International Journal of Bifurcation and Chaos*, *10*(06), 1171–1266. http://dx.doi.org/10.1142/S0218127400000840, URL https://www.worldscientific.com/doi/abs/10.1142/S0218127400000840.

Jadi, M., Polsky, A., Schiller, J., & Mel, B. W. (2012). Location-dependent effects of inhibition on local spiking in pyramidal neuron dendrites. *PLoS Computational Biology*, *8*(6), Article e1002550. http://dx.doi.org/10.1371/journal.pcbi.1002550.

Jordan, M. I., & Rumelhart, D. E. (1992). Forward models: Supervised learning with a distal teacher. *Cognitive Science*, *16*(3), 307–354. http://dx.doi.org/10.1207/s15516709cog1603_1, URL https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog1603_1. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1207/s15516709cog1603_1.

Kawato, M., & Gomi, H. (1992). A computational model of four regions of the cerebellum based on feedback-error learning. *Biological Cybernetics*, *68*(2), 95–103. http://dx.doi.org/10.1007/BF00201431, URL http://link.springer.com/article/10.1007/BF00201431.

Kohonen, T. (1995). Variants of SOM. In T. Kohonen (Ed.), *Springer series in information sciences*, *Self-organizing maps* (pp. 143–173). Berlin, Heidelberg: Springer, http://dx.doi.org/10.1007/978-3-642-97610-0_5.

Kolodziejski, C., Porr, B., & Wörgötter, F. (2008a). Mathematical properties of neuronal TD-rules and differential Hebbian learning: A comparison. *Biological Cybernetics*, *98*(3), 259. http://dx.doi.org/10.1007/s00422-007-0209-6.

Kolodziejski, C., Porr, B., & Wörgötter, F. (2008b). On the asymptotic equivalence between differential Hebbian and temporal difference learning. *Neural Computation*, *21*(4), 1173–1202. http://dx.doi.org/10.1162/neco.2008.04-08-750.

Kuang, S., & Gail, A. (2015). When adaptive control fails: slow recovery of reduced rapid online control during reaching under reversed vision. *Vision Research*, *110*, 155–165. http://dx.doi.org/10.1016/j.visres.2014.08.021, URL https://www.sciencedirect.com/science/article/pii/S0042698914002089.

Kulkarni, T. D., Narasimhan, K., Saeedi, A., & Tenenbaum, J. (2016). Hierarchical deep reinforcement learning: integrating temporal abstraction and intrinsic motivation. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems 29* (pp. 3675–3683). Curran Associates, Inc., URL http://papers.nips.cc/paper/6233-hierarchical-deep-reinforcement-learning-integrating-temporal-abstraction-and-intrinsic-motivation.pdf.

Kuperstein, M. (1988). Neural model of adaptive hand-eye coordination for single postures. *Science*, *239*(4845), 1308–1311. http://dx.doi.org/10.1126/science.3344437, URL https://www.sciencemag.org/lookup/doi/10.1126/science.3344437.

Legenstein, R., Chase, S. M., Schwartz, A. B., & Maass, W. (2010). A reward-modulated Hebbian learning rule can explain experimentally observed network reorganization in a brain control task. *Journal of Neuroscience*, *30*(25), 8400–8410. http://dx.doi.org/10.1523/JNEUROSCI.4284-09.2010, URL https://www.jneurosci.org/content/30/25/8400.

Liang, H., Paxinos, G., & Watson, C. (2011). Projections from the brain to the spinal cord in the mouse. *Brain Structure and Function*, *215*(3), 159–186. http://dx.doi.org/10.1007/s00429-010-0281-x.

Lillicrap, T. P., Moreno-Briseño, P., Diaz, R., Tweed, D. B., Troje, N. F., & Fernandez-Ruiz, J. (2013). Adapting to inversion of the visual field: A new twist on an old problem. *Experimental Brain Research*, *228*(3), 327–339. http://dx.doi.org/10.1007/s00221-013-3565-6, URL http://link.springer.com/10.1007/s00221-013-3565-6.

Lim, S., & Goldman, M. S. (2014). Balanced cortical microcircuitry for spatial working memory based on corrective feedback control. *The Journal of Neuroscience*, *34*(20), 6790–6806. http://dx.doi.org/10.1523/JNEUROSCI.4602-13.2014, URL http://www.jneurosci.org/content/34/20/6790.

Manto, M., Bower, J. M., Conforto, A. B., Delgado-García, J. M., Guarda, S. N. F. d., Gerwig, M., et al. (2012). Consensus paper: Roles of the cerebellum in motor control—The diversity of ideas on cerebellar involvement in movement. *The Cerebellum*, *11*(2), 457–487. http://dx.doi.org/10.1007/s12311-011-0331-9, URL http://link.springer.com/article/10.1007/s12311-011-0331-9.

Mazzoni, P., Andersen, R. A., & Jordan, M. I. (1991). A more biologically plausible learning rule for neural networks. *Proceedings of the National Academy of Sciences*, *88*(10), 4433–4437. http://dx.doi.org/10.1073/pnas.88.10.4433, URL https://www.pnas.org/content/88/10/4433.

McNamee, D., & Wolpert, D. M. (2019). Internal models in biological control. *Annual Review of Control, Robotics, and Autonomous Systems*, *2*(1), 339–364. http://dx.doi.org/10.1146/annurev-control-060117-105206, URL https://www.annualreviews.org/doi/10.1146/annurev-control-060117-105206.

Melchior, J., & Wiskott, L. (2019). Hebbian-descent. URL arXiv:1905.10585.

Miall, R. C., & Wolpert, D. M. (1996). Forward models for physiological motor control. *Neural Networks*, *9*(8), 1265–1279. http://dx.doi.org/10.1016/S0893-6080(96)00035-4, URL https://www.sciencedirect.com/science/article/pii/S0893608096000354.

Middleton, F. A., & Strick, P. L. (1997). Chapter 32 dentate output channels: Motor and cognitive components. In C. I. De Zeeuw, P. Strata, & J. Voogd (Eds.), *The cerebellum: From structure to control*: *vol. 114*, *Progress in brain research* (pp. 553–566). Elsevier, http://dx.doi.org/10.1016/S0079-6123(08)63386-5, URL https://www.sciencedirect.com/science/article/pii/S0079612308633865.

Miyamoto, H., Kawato, M., Setoyama, T., & Suzuki, R. (1988). Feedback-error-learning neural network for trajectory control of a robotic manipulator. *Neural Networks*, *1*(3), 251–265. http://dx.doi.org/10.1016/0893-6080(88)90030-5, URL http://www.sciencedirect.com/science/article/pii/0893608088900305.

Mountcastle, V. B. (1997). The columnar organization of the neocortex. *Brain*, *120*(4), 701–722. http://dx.doi.org/10.1093/brain/120.4.701, URL https://academic.oup.com/brain/article/120/4/701/372118.

Najafi, F., Elsayed, G. F., Cao, R., Pnevmatikakis, E., Latham, P. E., Cunningham, J. P., et al. (2020). Excitatory and inhibitory subnetworks are equally selective during decision-making and emerge simultaneously during learning. *Neuron*, *105*(1), 165–179.e8. http://dx.doi.org/10.1016/j.neuron.2019.09.045, URL https://www.sciencedirect.com/science/article/pii/S0896627319308487.

Nijmeijer, H., & van der Schaft, A. (1990). The input-output decoupling problem. In H. Nijmeijer, & A. van der Schaft (Eds.), *Nonlinear dynamical control systems* (pp. 223–250). New York, NY: Springer, http://dx.doi.org/10.1007/978-1-4757-2101-0_8.

Norton, J. J., & Wolpaw, J. R. (2018). Acquisition, maintenance, and therapeutic use of a simple motor skill. *Current Opinion in Behavioral Sciences*, *20*, 138–144. http://dx.doi.org/10.1016/j.cobeha.2017.12.021, URL http://www.sciencedirect.com/science/article/pii/S235215461730219X.

Nudo, R. J., & Masterton, R. B. (1989). Descending pathways to the spinal cord: II. Quantitative study of the tectospinal tract in 23 mammals. *Journal of Comparative Neurology*, *286*(1), 96–119. http://dx.doi.org/10.1002/cne.902860107, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cne.902860107. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cne.902860107.

Okun, M., & Lampl, I. (2008). Instantaneous correlation of excitation and inhibition during ongoing and sensory-evoked activities. *Nature Neuroscience*, *11*(5), 535–537. http://dx.doi.org/10.1038/nn.2105, URL https://www.nature.com/articles/nn.2105.

Okun, M., & Lampl, I. (2009). Balance of excitation and inhibition. *Scholarpedia*, *4*(8), 7467. http://dx.doi.org/10.4249/scholarpedia.7467, URL http://www.scholarpedia.org/article/Balance_of_excitation_and_inhibition.

Pei, Y. -C., Hsiao, S. S., Craig, J. C., & Bensmaia, S. J. (2010). Shape invariant coding of motion direction in somatosensory cortex. *PLoS Biology*, *8*(2), http://dx.doi.org/10.1371/journal.pbio.1000305, URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2814823/.

Porr, B., Ferber, C. v., & Wörgötter, F. (2003). ISO learning approximates a solution to the inverse-controller problem in an unsupervised behavioral paradigm. *Neural Computation*, *15*(4), 865–884. http://dx.doi.org/10.1162/08997660360581930, URL https://www.mitpressjournals.org/doi/10.1162/08997660360581930.

Porr, B., & Wörgötter, F. (2006). Strongly improved stability and faster convergence of temporal sequence learning by using input correlations only. *Neural Computation*, *18*(6), 1380–1412. http://dx.doi.org/10.1162/neco.2006.18.6.1380.

Porrill, J., Dean, P., & Anderson, S. R. (2013). Adaptive filters and internal models: multilevel description of cerebellar function. *Neural Networks*, *47*, 134–149. http://dx.doi.org/10.1016/j.neunet.2012.12.005, URL http://www.sciencedirect.com/science/article/pii/S0893608012003206.

Porrill, J., Dean, P., & Stone, J. (2004). Recurrent cerebellar architecture solves the motor-error problem. *Proceedings of the Royal Society of London, Series B*, *271*(1541), 789–796. http://dx.doi.org/10.1098/rspb.2003.2658, URL http://classic.rspb.royalsocietypublishing.org/content/271/1541/789.

Powers, W. T. (2005). *Behavior: The Control of Perception (2nd ed. rev. & exp.), Vol. xiv*. New Canaan, CT, US: Benchmark Press.

Raphael, G., Tsianos, G. A., & Loeb, G. E. (2010). Spinal-like regulator facilitates control of a two-degree-of-freedom wrist. *Journal of Neuroscience*, *30*(28), 9431–9444. http://dx.doi.org/10.1523/JNEUROSCI.5537-09.2010, URL https://www.jneurosci.org/content/30/28/9431.

Richter, H., Magnusson, S., Imamura, K., Fredrikson, M., Okura, M., Watanabe, Y., et al. (2002). Long-term adaptation to prism-induced inversion of the retinal images. *Experimental Brain Research*, *144*(4), 445–457. http://dx.doi.org/10.1007/s00221-002-1097-6.

Rokni, U. (2009). Neural networks for control. *Encyclopedia of Neuroscience*, 2592–2596. http://dx.doi.org/10.1007/978-3-540-29678-2_3795, URL https://link.springer.com/referenceworkentry/10.1007/978-3-540-29678-2_3795.

Sachse, P., Beermann, U., Martini, M., Maran, T., Domeier, M., & Furtner, M. R. (2017). "The world is upside down" – The innsbruck goggle experiments of Theodor Erismann (1883–1961) and Ivo Kohler (1915–1985). *Cortex*, *92*, 222–232. http://dx.doi.org/10.1016/j.cortex.2017.04.014, URL https://www.sciencedirect.com/science/article/pii/S0010945217301314.

Schultz, W., Dayan, P., & Montague, P. R. (1997). A neural substrate of prediction and reward. *Science*, *275*(5306), 1593–1599. http://dx.doi.org/10.1126/science.275.5306.1593, URL https://science.sciencemag.org/content/275/5306/1593.

Seborg, D. E., Edgar, T. F., Mellichamp, D. A., & III, F. J. D. (2016). *Process dynamics and control, 4th edition*. Wiley.

Sekiyama, K., Miyauchi, S., Imaruoka, T., Egusa, H., & Tashiro, T. (2000). Body image as a visuomotor transformation device revealed in adaptation to reversed vision. *Nature*, *407*(6802), 374–377. http://dx.doi.org/10.1038/35030096, URL https://www.nature.com/articles/35030096.

Shadmehr, R., & Wise, S. P. (2005). *The computational neurobiology of reaching and pointing: a foundation for motor learning*. MIT Press.

Shafi, M., Zhou, Y., Quintana, J., Chow, C., Fuster, J., & Bodner, M. (2007). Variability in neuronal activity in primate cortex during working memory tasks. *Neuroscience*, *146*(3), 1082–1108. http://dx.doi.org/10.1016/j.neuroscience.2006.12.072, URL http://www.sciencedirect.com/science/article/pii/S0306452206017593.

Sontag, E. D. (2013). *Mathematical control theory: Deterministic finite dimensional systems*. Springer Science & Business Media, Google-Books-ID: F9XiBwAAQBAJ.

Steinmetz, N. A., Zatka-Haas, P., Carandini, M., & Harris, K. D. (2019). Distributed coding of choice, action and engagement across the mouse brain. *Nature*, *576*(7786), 266–273. http://dx.doi.org/10.1038/s41586-019-1787-x, URL https://www.nature.com/articles/s41586-019-1787-x.

Strang, G. (1993). Wavelet transforms versus Fourier transforms. *American Mathematical Society. Bulletin*, *28*(2), 288–305. http://dx.doi.org/10.1090/S0273-0979-1993-00390-2, URL https://www.ams.org/bull/1993-28-02/S0273-0979-1993-00390-2/.

Sugimoto, N., Haruno, M., Doya, K., & Kawato, M. (2011). MOSAIC for multiple-reward environments. *Neural Computation*, *24*(3), 577–606. http://dx.doi.org/10.1162/NECO_a_00246, URL https://www.mitpressjournals.org/doi/10.1162/NECO_a_00246.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT Press, Google-Books-ID: SWV0DwAAQBAJ.

Tanaka, H., Ishikawa, T., Lee, J., & Kakei, S. (2020). The cerebro-cerebellum as a locus of forward model: A review. *Frontiers in Systems Neuroscience*, *14*, http://dx.doi.org/10.3389/fnsys.2020.00019, URL https://www.frontiersin.org/articles/10.3389/fnsys.2020.00019/full.

Todorov, E. (2000). Direct cortical control of muscle activation in voluntary arm movements: A model. *Nature Neuroscience*, *3*(4), 391–398. http://dx.doi.org/10.1038/73964, URL https://www.nature.com/articles/nn0400_391. Bandiera_abtest: A Cg_type: Nature Research Journals.

Tseng, Y.-w., Diedrichsen, J., Krakauer, J. W., Shadmehr, R., & Bastian, A. J. (2007). Sensory prediction errors drive cerebellum-dependent adaptation of reaching. *Journal of Neurophysiology*, *98*(1), 54–62. http://dx.doi.org/10.1152/jn.00266.2007, URL http://jn.physiology.org/content/98/1/54.

Verduzco-Flores, S., & De Schutter, E. (2019). Draculab: A Python simulator for firing rate neural networks with delayed adaptive connections. *Frontiers in Neuroinformatics*, *13*, http://dx.doi.org/10.3389/fninf.2019.00018, URL https://www.frontiersin.org/articles/10.3389/fninf.2019.00018/full.

Verduzco-Flores, S., & De Schutter, E. (2021). Adaptive plasticity in the spinal cord can produce reaching from scratch and reproduces motor cortex directional tuning. q-bio.

Verduzco-Flores, S. O., & O'Reilly, R. C. (2015). How the credit assignment problems in motor control could be solved after the cerebellum predicts increases in error. *Frontiers in Computational Neuroscience*, *9*, http://dx.doi.org/10.3389/fncom.2015.00039, URL https://www.frontiersin.org/articles/10.3389/fncom.2015.00039/full.

Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., et al. (2017). FeUdal networks for hierarchical reinforcement learning. URL arXiv:1703.01161 [cs].

Werfel, J., Xie, X., & Seung, H. S. (2005). Learning curves for stochastic gradient descent in linear feedforward networks. *Neural Computation*, *17*(12), 2699–2718. http://dx.doi.org/10.1162/089976605774320539.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*, 229–256. http://dx.doi.org/10.1007/BF00992696.

Wolpaw, J. R. (1997). The complex structure of a simple memory. *Trends in Neurosciences*, *20*(12), 588–594. http://dx.doi.org/10.1016/S0166-2236(97)01133-8, URL http://www.sciencedirect.com/science/article/pii/S0166223697011338.

Wolpert, D. M., Ghahramani, Z., & Jordan, M. I. (1995). An internal model for sensorimotor integration. *Science*, *269*(5232), 1880–1882. http://dx.doi.org/10.1126/science.7569931, URL http://www.sciencemag.org/content/269/5232/1880.

Yamashita, H., Chen, S., Komagata, S., Hishida, R., Iwasato, T., Itohara, S., et al. (2012). Restoration of contralateral representation in the mouse somatosensory cortex after crossing nerve transfer. *PLoS One*, *7*(4), Article e35676. http://dx.doi.org/10.1371/journal.pone.0035676, URL https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0035676.

Yin, H. H. (2017). The basal ganglia in action. *The Neuroscientist*, *23*(3), 299–313. http://dx.doi.org/10.1177/1073858416654115.

Yin, H. H., & Knowlton, B. J. (2006). The role of the basal ganglia in habit formation. *Nature Reviews Neuroscience*, *7*(6), 464–476. http://dx.doi.org/10.1038/nrn1919, URL https://www.nature.com/articles/nrn1919.