frontiers in Neurorobotics

METHODS
published: 09 August 2017
doi: 10.3389/fnbot.2017.00037

A Functional Subnetwork Approach to Designing Synthetic Nervous Systems That Control Legged Robot Locomotion

Nicholas S. Szczecinski[1]*, *Alexander J. Hunt*[2] and *Roger D. Quinn*[1]

[1]Biologically Inspired Robotics Laboratory, Department of Mechanical and Aerospace Engineering, Case Western Reserve University, Cleveland, OH, United States, [2]Department of Mechanical and Materials Engineering, Portland State University, Portland, OR, United States

A dynamical model of an animal's nervous system, or synthetic nervous system (SNS), is a potentially transformational control method. Due to increasingly detailed data on the connectivity and dynamics of both mammalian and insect nervous systems, controlling a legged robot with an SNS is largely a problem of parameter tuning. Our approach to this problem is to design functional subnetworks that perform specific operations, and then assemble them into larger models of the nervous system. In this paper, we present networks that perform addition, subtraction, multiplication, division, differentiation, and integration of incoming signals. Parameters are set within each subnetwork to produce the desired output by utilizing the operating range of neural activity, R, the gain of the operation, k, and bounds based on biological values. The assembly of large networks from functional subnetworks underpins our recent results with MantisBot.

Keywords: synthetic nervous system, design tools, functional subnetworks, leaky integrator, arithmetic, differentiator, memory

OPEN ACCESS

Edited by:
Manish Sreenivasa, Heidelberg University, Germany

Reviewed by:
Shinya Aoi, Kyoto University, Japan
Sander Bohte, Centrum Wiskunde & Informatica, Netherlands
Elisa Donati, Sant'Anna School of Advanced Studies, Italy

***Correspondence:**
Nicholas S. Szczecinski
nicholas.szczecinski@case.edu

Received: *30 December 2016*
Accepted: *17 July 2017*
Published: *09 August 2017*

Citation:
Szczecinski NS, Hunt AJ and Quinn RD (2017) A Functional Subnetwork Approach to Designing Synthetic Nervous Systems That Control Legged Robot Locomotion. Front. Neurorobot. 11:37. doi: 10.3389/fnbot.2017.00037

1. INTRODUCTION

The development of robotic control that can closely match the dexterity and adaptability found in the animal kingdom has so far remained elusive. This is because the control of locomotion is a complex process controlled by dynamic systems which are not fully understood. However, recent advances in neural imaging and recording has lead to an increase in the abundance and detail of our knowledge of how an animal's nervous system controls its body within the context of its environment (for a recent review, see Buschmann et al. (2015)).

These advances have lead to an explosion of bio-inspired robotic systems in recent years (for a review, see Ijspeert (2014)). These models can be broadly categorized into a range of template and anchor models. In a template model, biological principles are abstracted, such using as a spring-loaded inverted pendulum (SLIP) model to investigate bipedal locomotion (Blickhan, 1989) or using Whegs to investigate insect locomotion (Allen et al., 2003; Schroer et al., 2004). These models seek to explain how specific characteristics of animal locomotion lead to desired behaviors, or they exploit certain principles of animal locomotion for more agile robotic systems. Anchor models, in contrast, seek to directly mimic particular mechanical or control mechanisms from animals, in order to understand how they function. Robots such as Pleurobot (Karakasiliotis et al., 2016), Puppy (Hunt et al., 2017), MantisBot (Szczecinski and Quinn, 2017; Szczecinski et al., 2017a), and others are relevant anchor models, because they seek to use highly articulated robots with central

pattern generator controllers to understand how specific animals are capable of providing adaptable locomotion with their unique morphology and physical constraints.

The template versus anchor model distinction is not limited to physical models; it can also be applied to control systems. The majority of robotic controllers so far have been template models, either mathematical abstractions of neural systems, or black box artificial neural networks. This is because effective tools for setting parameters in more realistic, dynamic neural models to produce reliable behavior in a robotic system do not yet exist. In spite of growing knowledge about the neural connectivity that underlies locomotion control, detailed data for tuning these systems (neural time constants, ion channel conductivities, synaptic conductivities, etc.) remain largely unavailable, requiring the modeler or engineer to tune these parameter values. However, this is an inherently difficult task because there are many parameters to be tuned in a model, and likely many different parameter combinations that lead to indistinguishable performance (Prinz et al., 2004; Marder and Taylor, 2011). Thus, the emphasis in choosing parameter values should not be on selecting the singular "correct" values, but rather sufficiently "effective" values. In this work, we tune parameter values in functional subnetworks for addition, subtraction, multiplication, division, differentiation, and integration of incoming signals and use analytical techniques to identify *constraints* on the parameter values that must be met for the intended calculations to occur. Larger networks can then be assembled from these subnetworks with no additional tuning (Szczecinski and Quinn, 2017; Szczecinski et al., 2017a). In this manuscript, "tuning" refers to selecting the static parameter values for a network; "learning" refers to changing the parameter values while the model performs a task, either in simulation or in a robot, based on its performance; and "adapting" refers to a model qualitatively changing its behavior (e.g., walking more slowly), either with or without "learning."

Neuromechanical models may be tuned in a supervised or unsupervised way. A supervised tuning method adjusts the parameters of the model until the model replicates animal data. This includes tuning the model by hand (Daun-Gruhn and Tóth, 2010; Szczecinski et al., 2014; Markin et al., 2016), which is a time-consuming and imprecise process. Such imprecision may be acceptable in simulation studies, but provides many difficulties for robots that must interact with real environments. Techniques do exist for tuning controllers based on animal locomotion data (Schilling et al., 2013; Hunt et al., 2015b, 2017; Karakasiliotis et al., 2016). However, collecting kinematic and dynamic data from animals is time-consuming and expensive, and once collected, must be further processed to scale the dynamics of the animal to the robot (Karakasiliotis et al., 2016; Hunt et al., 2017). In addition, using cross-individual average values for tuning dynamical neural models may fail in many cases, because the average may not represent any one individual (Golowasch et al., 2002; Marder and Taylor, 2011). Large amounts of animal data may be used to tune a control network of abstracted artificial neural networks (Schilling et al., 2013). Methods like back-propagation can be used to adjust synaptic weights in the network until it captures the animal data arbitrarily closely, if it has enough connections (Trappenberg, 2009). However, because the control network is abstracted, so are the biological insights gained from the model.

Unsupervised tuning methods, in contrast, tune the model based on how well the model accomplishes a task, such as navigating toward a goal, without comparison to animal data. These methods frequently use genetic algorithms (GAs) (Beer and Gallagher, 1992; Haferlach et al., 2007; Agmon and Beer, 2013; Izquierdo and Beer, 2013) or reservoir computing (RC) (Dasgupta et al., 2015) to test many different networks and parameter values, based on a simulated agent's performance. GAs can be effective at finding networks that perform specific operations, such as oscillating (Beer and Gallagher, 1992), navigating (Haferlach et al., 2007), or switching between foraging tasks (Agmon and Beer, 2013). However, this approach has some drawbacks. Specifically, the evolution process is slow, requiring the simulation of hundreds or thousands of parameter combinations (Agmon and Beer, 2013), which may take days without great computing power. The speed and likelihood of success can be increased by embedding functional subnetworks in the network (Pasemann et al., 2001; Haferlach et al., 2007), which may be identified by brute-force (Prinz et al., 2003), dynamical systems analysis (Hunt et al., 2017), or constraints on network connectivity and parameter values (Haferlach et al., 2007). In this paper, we analytically derive parameter constraints to eliminate the need for GAs altogether, and guarantee network performance.

RC methods simulate large "reservoirs" of randomly connected dynamical neuron models, and then use optimization methods to map reservoir activity to learned useful values. While this method can produce capable robotic controllers (Dasgupta et al., 2015), the final system is likely more complicated than is ultimately necessary, increasing its computational cost to implement. In addition, the final system is a black box, which does not provide any insights about nervous system function. The methods in this paper enable the direct assembly and tuning of dynamical networks without the need of large reservoirs of neurons.

This work analytically derives constraints that govern the behavior of synthetic nervous systems (SNSs) built from dynamical neural networks. These constraints were derived as a result of our previous network design work (Szczecinski et al., 2017b) and have enabled the rapid assembly and testing of our recent robot control networks (Szczecinski and Quinn, 2017; Szczecinski et al., 2017a). An SNS designer can apply these constraints to find parameter values needed for a functional network. Section 2 presents the neural and synaptic models and explains how the neural system encodes mechanical inputs and outputs. Section 3 derives two basic synapse types, "signal transmission" and "signal modulation," and uses them to derive constraints on synaptic parameters in networks performing addition, subtraction, multiplication, and division of two incoming signals. Section 4 derives constraints on neural and synaptic parameters in networks that differentiate and integrate incoming signals as a function of time. Results showing that the networks perform as intended are provided throughout the manuscript, and **Tables 1** and **2** summarize the design constraints. Finally, Sec. 6 explores how these techniques may be used to tune robot controllers and neuromechanical models of animals, and how they may be improved in the future.

2. METHODS: MODELS AND APPROACH

We model neurons as non-spiking Hodgkin–Huxley compartments (Cofer et al., 2010), the same basic model as used in continuous-time recurrent neural networks (Haferlach et al., 2007; Agmon and Beer, 2013). The leaky integrator dynamics capture the most basic behavior of neurons and allow more complex behaviors to be added with additional ion channels, if desired. This work is not concerned with the specifics on how action potentials are generated and have left out Hodgkin–Huxley sodium and potassium currents. The membrane voltage, V, may be seen as a proxy for the spiking frequency of a spiking neuron. V varies according to the differential equation

$$C_m \frac{dV}{dt} = I_{leak} + I_{syn} + I_{app} \tag{1}$$

where

$$I_{leak} = G_m \cdot (E_r - V), \tag{2}$$

$$I_{syn} = \sum_{i=1}^{n} G_{s,i} \cdot (E_{s,i} - V), \tag{3}$$

and I_{app} is an optional external stimulus. Equations (2) and (3) define the leak and synaptic currents, respectively. Both follow the same basic form of a conductance G multiplied by the difference between the current membrane voltage, V, and a constant reference voltage (i.e., reversal potential), E. E_r is the resting potential of the neuron, and C_m and G_m are the capacitance and conductance of the cell membrane, respectively. Unless otherwise noted, all units in this paper are scaled to nA for current, mV for potentials, nF for capacitances, and μS for conductances.

Neurons communicate via synapses. The conductance, $G_{s,i}$ in equation (3), is a threshold linear function of the ith incoming (i.e., presynaptic) neuron's voltage. Synapses communicate via piecewise-linear functions described as

$$G_{s,i} = \begin{cases} 0, & \text{if } V_{pre} < E_{lo}, \\ g_{s,i} \cdot \frac{V_{pre} - E_{lo}}{E_{hi} - E_{lo}}, & \text{if } E_{lo} < V_{pre} < E_{hi}, \\ g_{s,i}, & \text{if } V_{pre} > E_{hi}. \end{cases} \tag{4}$$

The parameters $g_{s,i}$, E_{lo}, and E_{hi} are constants representing the synapse's maximum conductance, its lower threshold, and its upper threshold, respectively. The relationship between the presynaptic neuron voltage, synaptic conductance, and postsynaptic neuron voltage is illustrated in **Figure 1A**.

We prefer this piecewise-linear representation better than a sigmoidal function for several reasons. First, the thresholds ensure that for low activations, synapses conduct exactly 0 current. This could represent a reduced model of a spiking neuron, which transmits no information while it is not spiking. Second, equation (4) contains no transcendental terms, facilitating analytical manipulation of the equations. A discontinuous system does complicate traditional gradient-based optimization methods, but this structure can be exploited to make these methods unnecessary. In the following sections, we show how networks of three or four neurons with synapses between them can be constructed and analytically

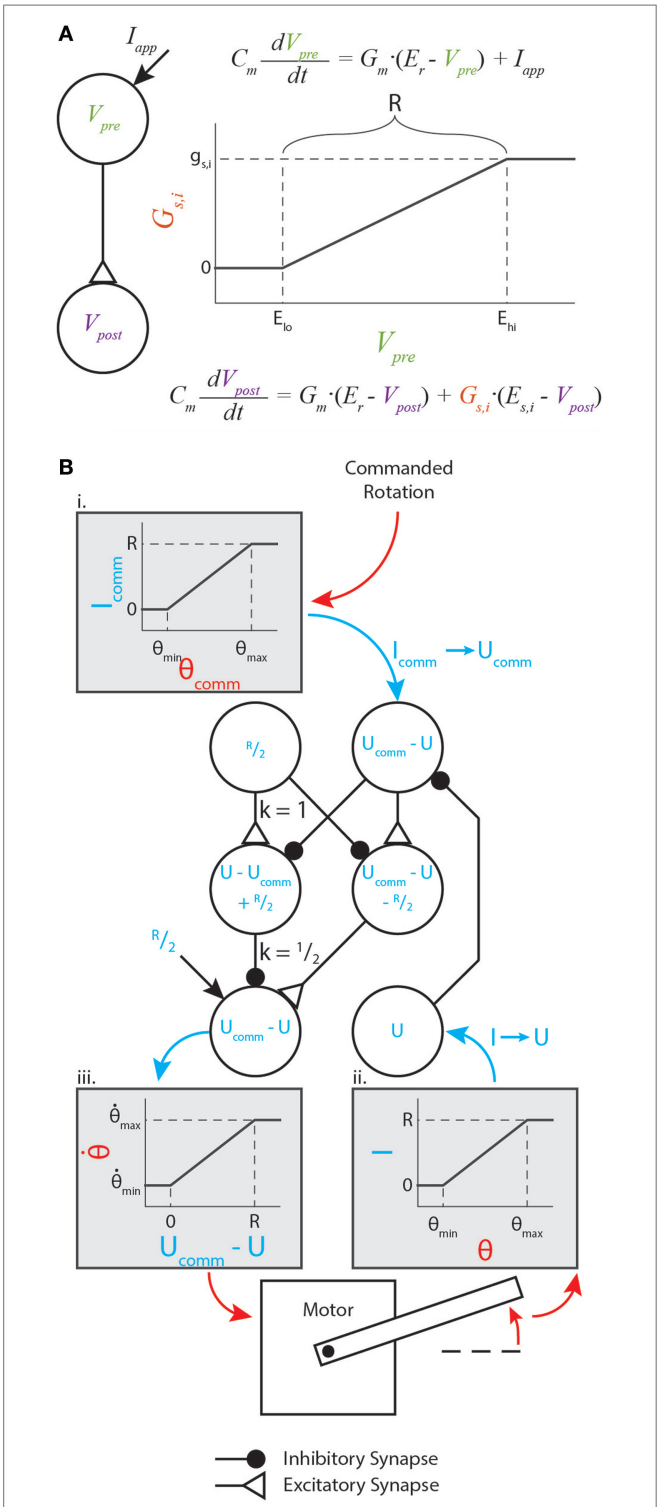

FIGURE 1 | Graphical representation of synaptic dynamics and mapping between mechanical and neural values. **(A)** Graphical representation of how synapses couple neural dynamics. Note that R is marked on the plot. **(B)** Enhanced version of the motor control network from Szczecinski et al. (2017b), showing how R relates mechanical and neural values. Mechanical values are drawn in red, and neural values are drawn in blue. Gray shaded boxes map from mechanical to neural values (i. and ii.), or from neural to mechanical values (iii.).

tuned to perform mathematical operations on the input signals, such as addition or differentiating with respect to time.

Instead of analyzing V when designing these networks, we shift the neural activity to simplify analysis. For each neuron, we substitute $U = V - E_r$, the activation level above the resting voltage. A typical value is $E_r = -60$ mV, but using U for analysis rather than V lets us apply the same analysis no matter what E_r is. We also set $G_m = 1\ \mu$S, which is a typical value (Daun-Gruhn et al., 2009; Daun-Gruhn, 2010).

For the synapses, we set $E_{lo} = E_r$ of the presynaptic neuron, and introduce a new parameter $R = E_{hi} - E_{lo}$. Thus, a synapse's conductivity rises as the presynaptic neuron's voltage rises above its resting potential, and exhibits an "operating range" of R mV. The constraints we apply ensure that $U_{pre} \in [0, R]$, meaning that the synapse is always active, but never saturates. Thus, we can replace G_s with the second line of equation (4). Applying the substitutions described so far,

$$G_s = g_s \cdot \frac{V_{pre} - E_{lo}}{E_{hi} - E_{lo}} = g_s \cdot \frac{U_{pre}}{R} = \frac{g_s}{R} \cdot U_{pre}. \tag{5}$$

For each synapse, we also introduce the parameter $\Delta E_{s,i} = E_{s,i} - E_{r,post}$, where $E_{r,post}$ is the resting potential of the postsynaptic, or receiving neuron.

Making all of these substitutions in equations (1)–(3) gives the response

$$C_m \frac{dU}{dt} = -U + \sum_{i=1}^{n} \frac{g_{s,i}}{R} \cdot U_{pre,i} \cdot (\Delta E_{s,i} - U) + I_{app}. \tag{6}$$

When $U = R$, the neuron is fully active, and when $U = 0$, the neuron is inactive. We can use this knowledge to categorize synapses as excitatory or inhibitory, depending on the sign of $\Delta E_{s,i}$. If $\Delta E_{s,i} \geq R$, then the ith synapse will always transmit positive current, no matter the instantaneous value of U. Thus, this synapse will cause U to increase and is, therefore, excitatory. Similarly, if $\Delta E_{s,i} \leq 0$, then the ith synapse will always transmit negative current, no matter the instantaneous value of U. Thus, this synapse will cause U to decrease and is, therefore, inhibitory.

2.1. Mapping between Neural and Mechanical Values

The nervous system encodes physical quantities as neural activity. In insects, the firing rate of sensory neurons encode the stretch of chordotonal organs (Field and Matheson, 1998) and the strain of campaniform sensilla (Zill et al., 2004), among other physical quantities. Typical robot controllers perform operations on these signals to provide meaningful information for control actions. These operations may include the subtraction of measured and reference values, differentiation or integration of error values, or gain adjustments. Neural systems perform these same operations, but in a transformed space. The exact transformation that nervous systems use is not known, but for reliable behavior, it is necessary that sensory information is mapped to neural activity in apredictable way. Thus, we map any sensory input, θ, to an applied current

$$I_{app} = R \cdot \frac{\theta - \theta_{\min}}{\theta_{\max} - \theta_{\min}}, \tag{7}$$

where R is the "operating range" specified in the previous section. **Figure 1B** illustrates such a transformation within a diagram of a neural feedback loop that controls the position of a motor. The purpose of this paper is not to analyze how this particular network functions; for a detailed analysis of this network and its function, see Szczecinski et al. (2017b). Instead, the purpose is to show how R and other functional values (time constants, gains, etc.) may be used to constrain neural and synaptic parameter values.

Figure 1B (i,ii) graphically illustrate the mapping in equation (7), and **Figure 1B** (iii) graphically illustrates the inverse relationship (i.e., neural value to mechanical value). If a sensory neuron has only this applied current and leak current, equation (6) shows that

$$C_m \frac{dU}{dt} + U = R \cdot \frac{\theta - \theta_{\min}}{\theta_{\max} - \theta_{\min}}. \tag{8}$$

This means that the sensory neuron acts as a low-pass filter with time constant $\tau = C_m$. It is trivial to show that, when the neuron is at equilibrium (i.e., $dU/dt = 0$),

$$U^* = R \cdot \frac{\theta - \theta_{\min}}{\theta_{\max} - \theta_{\min}}, \tag{9}$$

where the superscript "$*$" specifies the equilibrium value. (Throughout this manuscript, the equilibrium activation of neuron U will be referred to as U^*, and the neuron itself will be referred to as U.) Equation (9) means that the neuron's activation above its rest potential encodes the sensory signal. In addition to perceiving sensory information, commands must be issued in the same transformation. Thus, we map the commanded sensory quantity, θ_{comm}, to the commanded neural activation, U_{comm}, with the inverse function of equation (9),

$$\theta_{\text{comm}} = \theta_{\min} + \frac{U_{\text{comm}}}{R} \cdot (\theta_{\max} - \theta_{\min}). \tag{10}$$

In this way, the nervous system may specify an intended motion, such as the rotation of a joint, encoded in neural activity. In our synthetic nervous systems, R specifies how mechanical quantities and neural activation are related. Thus, the tuning of every functional subnetwork described in this work relies on R, which the designer specifies before tuning the rest of the network. Two other parameters are critical for tuning these subnetworks: the amplification of synaptic transmission, k_{syn} (discussed in Sec. 3.1), and the synaptic reversal potential, ΔE_s. From these values, biological parameters such as synaptic conductance and neural tonic drive can be directly calculated. This makes network design intuitive, enabling the designer to select biological parameter values based on functional ones.

3. METHODS: ARITHMETIC SUBNETWORKS

This section describes how to use typical engineering quantities to design neural and synaptic pathways. We can understand how these pathways work by manipulating their equilibria, something that naive optimization does not leverage. The steady-state activation U^* is calculated by solving for U when $dU/dt = 0$

$$0 = -U^* + \sum_{i=1}^{n} \frac{g_{s,i}}{R} \cdot U_{pre,i} \cdot \left(\Delta E_{s,i} - U^*\right) + I_{app}. \tag{11}$$

Moving all U^* terms to the left hand side

$$U^* \cdot \left(1 + \sum_{i=1}^{n} \frac{g_{s,i}}{R} \cdot U_{pre,i}\right) = \sum_{i=1}^{n} \frac{g_{s,i}}{R} \cdot U_{pre,i} \cdot \Delta E_{s,i} + I_{app}. \quad (12)$$

Solving for U^*,

$$U^* = \frac{\sum_{i=1}^{n} \frac{g_{s,i}}{R} \cdot U_{pre,i} \cdot \Delta E_{s,i} + I_{app}}{1 + \sum_{i=1}^{n} \frac{g_{s,i}}{R} \cdot U_{pre,i}}. \quad (13)$$

This solution is the basis for the remainder of Sec. 3.

3.1. Signal Transmission Pathways

The goal of a signal transmission pathway is to cause the postsynaptic neuron's voltage to be some ratio of the presynaptic neuron's voltage. We call this ratio k_{syn}. The $U_{pre,i}$ terms in the denominator of the right hand side of equation (13) mean that k_{syn} changes as U_{pre} changes, so we approximate k_{syn} as U_{post}/U_{pre} when the presynaptic neuron is fully activated (i.e., $U_{pre} = R$). The steady-state response of a neuron with a single synaptic input and no applied current can be written based on equation (13), as below:

$$U^*_{post} = \frac{\frac{g_s}{R} \cdot U_{pre} \cdot \Delta E_s}{1 + \frac{g_s}{R} \cdot U_{pre}}. \quad (14)$$

To find k_{syn} for this synapse, we first divide both sides of equation (14) by U_{pre},

$$\frac{U^*_{post}}{U_{pre}} = \frac{\frac{g_s}{R} \cdot U_{pre} \cdot \Delta E_s}{U_{pre} \cdot \left(1 + \frac{g_s}{R} \cdot U_{pre}\right)}. \quad (15)$$

Next, we want to find k_{syn}, which can be calculated for any value of U_{pre}. To simplify analysis and improve the clarity of this derivation, we set find k_{syn} when $U_{pre} = R$. Then, we show how to set parameter values to keep k_{syn} nearly constant, even as U_{pre} changes. Making this substitution,

$$\frac{U^*_{post}}{R} = k_{syn} = \frac{\frac{g_s}{R} \cdot R \cdot \Delta E_s}{R \cdot \left(1 + \frac{g_s}{R} \cdot R\right)}. \quad (16)$$

Finally, reducing R/R terms reveals

$$k_{syn} = \frac{g_s \cdot \Delta E_s}{R \cdot (1 + g_s)}. \quad (17)$$

Rearranging to solve for g_s,

$$g_s = \frac{k_{syn} \cdot R}{\Delta E_s - k_{syn} \cdot R}. \quad (18)$$

Because g_s must be positive, and the numerator of equation (18) is always positive, equation (18) is also subject to the constraint

$$\Delta E_s > k_{syn} \cdot R. \quad (19)$$

Equation (18) will be used to tune addition, subtraction, multiplication, and division networks (Secs. 3.3 through 3.6).

3.2. Signal Modulation Pathways

We may also use synapses to modulate a neuron's sensitivity to other inputs. Based on equation (13), the steady-state response of a neuron with only an applied current I_{app} is simply

$$U^*_{post} = I_{app}, \quad (20)$$

if we set $G_m = 1$. For example, this is the case for a sensory neuron that receives applied current proportional to a sensor's state, such as a joint angle (**Figure 1B**), muscle stretch, or touch sensor. However, the nervous system may need to actively increase or reduce the sensitivity of the sensory neuron depending on context. Hyperpolarizing or depolarizing the neuron, however, would cause sensory information to be truncated (i.e., $V_{pre} < E_{lo}$). We can change the sensitivity of this neuron without losing sensory information by adding a synaptic input to the response from equation (20):

$$U^*_{post} = \frac{\frac{g_s}{R} \cdot U_{pre} \cdot \Delta E_s + I_{app}}{1 + \frac{g_s}{R} \cdot U_{pre}}. \quad (21)$$

To quantify how U_{pre} modulates U^*_{post} for a given I_{app}, we introduce the parameter c_{syn}, which quantifies this degree of modulation. We define c_{syn} as U^*_{post}/U_{pre}, the same as k_{syn}, but with the understanding that U_{pre} will decrease U^*_{post} in this case. Dividing both sides of equation (21) by U_{pre} and using the definition of c_{syn},

$$\frac{U^*_{post}}{U_{pre}} = c_{syn} = \frac{\frac{g_s}{R} \cdot U_{pre} \cdot \Delta E_s + I_{app}}{U_{pre} \cdot \left(1 + \frac{g_s}{R} \cdot U_{pre}\right)}. \quad (22)$$

As in the previous section, we will solve for c_{syn} when $U_{pre} = R$ to simplify analysis. Making this substitution and reducing R/R terms,

$$c_{syn} \cdot R = \frac{g_s \cdot \Delta E_s + R}{1 + g_s}. \quad (23)$$

Multiplying both sides by the denominator of the right hand side and expanding,

$$c_{syn} \cdot R + c_{syn} \cdot R \cdot g_s = g_s \cdot \Delta E_s + R. \quad (24)$$

Collecting g_s terms on the left hand side,

$$c_{syn} \cdot R \cdot g_s - g_s \cdot \Delta E_s = R - c_{syn} \cdot R. \quad (25)$$

Solving equation (25) for g_s,

$$g_s = \frac{c_{syn} \cdot R - R}{\Delta E_s - c_{syn} \cdot R}. \quad (26)$$

Just as in Sec. 3.1, $g_s > 0$ depends only on R, which the designer specifies beforehand, ΔE_s, which is limited by biological constraints, and c_{syn}, which the designer picks based on network function. ΔE_s should be negative, and as close to 0 as possible to minimize hyperpolarization of the postsynaptic neuron. Equation (26) will be used to tune division and multiplication networks (Secs. 3.5 and 3.6).

3.3. Addition

A subnetwork that approximates linear addition of the form $U^*_{post} = k_{syn} \cdot (U_{pre,1} + U_{pre,2})$ may underlie positive feedback mechanisms, which increase motor neuron activation proportional to sensory inputs such as force sensing organs (Zill et al., 2004), or used to sum sensory signals from different body segments (Mittelstaedt, 1957). We construct such a network by using two Signal Transmission pathways as presented in Sec. 3.1.

Let us rewrite equation (13) here, for clarity:

$$U^* = \frac{\sum_{i=1}^{n} \frac{g_{s,i}}{R} \cdot U_{pre,i} \cdot \Delta E_{s,i} + I_{app}}{1 + \sum_{i=1}^{n} \frac{g_{s,i}}{R} \cdot U_{pre,i}}. \tag{27}$$

This equation shows $U_{pre,i}$ in both the numerator and denominator. To capture addition, we wish to minimize the impact of $U_{pre,i}$ on the denominator. This is accomplished by minimizing g_s. However, if $g_s = 0$, then the network will not function at all. Therefore, we instead maximize ΔE_s, which yields a small g_s (equation (18)). Mathematically, there is no limit on ΔE_s, but synaptic potentials are limited in biological systems. In our work, we choose the reversal potential of calcium ($E_s = 134$ mV), which yields $\Delta E_s = E_s - E_r = 134 - (-60) = 194$ mV, and specify $R = 20$ mV. To design a pathway where $k_{syn} = 1$, for example, we plug these values into equation (18), which gives $g_s = 115$ nS. The contour plots in **Figure 2A** show that the network matches the ideal behavior very closely over the operating range $U_{sum} \in [0, R]$. These design constraints are summarized in **Table 1**.

TABLE 1 | This table assumed that the designer has already selected a value of R for the subnetwork.

Operation	Component pathways	Constraint equations	Free parameters
Addition	Syn. 1, transmission	$g_{s,1} = \frac{k_{syn,1} \cdot R}{\Delta E_{s,1} - k_{syn,1} \cdot R}$ $\Delta E_{s,1} - k_{syn,1} \cdot R > 0$	$k_{syn,1}$ $\Delta E_{s,1}$, maximize
	Syn. 2, transmission	$g_{s,2} = \frac{k_{syn,2} \cdot R}{\Delta E_{s,2} - k_{syn,2} \cdot R}$ $\Delta E_{s,2} - k_{syn,2} \cdot R > 0$	$k_{syn,2}$ $\Delta E_{s,2}$, maximize
Subtraction	Syn. 1, transmission	$g_{s,1} = \frac{k_{syn} \cdot R}{\Delta E_{s,1} - k_{syn} \cdot R}$ $\Delta E_{s,1} - k_{syn} \cdot R > 0$	k_{syn} $\Delta E_{s,1}$, maximize
	Syn. 2, transmission	$g_{s,2} = \frac{\Delta E_{s,1}}{\Delta E_{s,2}} \cdot \frac{-k_{syn} \cdot R}{\Delta E_{s,1} - k_{syn} \cdot R}$	$\Delta E_{s,2}$, minimize
Division	Syn. 1, transmission	$g_{s,1} = \frac{k_{syn} \cdot R}{\Delta E_{s,1} - k_{syn} \cdot R}$ $k_{syn} = 1$ $\Delta E_{s,1} - k_{syn} \cdot R > 0$	c_{syn} $\Delta E_{s,1}$, maximize
	Syn. 2, modulation	$g_{s,2} = \frac{c_{syn} \cdot R - R}{\Delta E_{s,2} - c_{syn} \cdot R}$ $\Delta E_{s,2} = 0$ $0 < c_{syn} < 1$	
Multiplication	Syn. 1, transmission	$g_{s,1} = \frac{k_{syn} \cdot R}{\Delta E_{s,1} - k_{syn} \cdot R}$ $k_{syn} = 1$ $\Delta E_{s,1} - k_{syn} \cdot R > 0$	 $\Delta E_{s,1}$, maximize
	Syn. 2, modulation	$g_{s,2} = \frac{-R}{\Delta E_{s,2}}$ $\Delta E_{s,2} < 0$	 $\Delta E_{s,2}$, maximize
	Syn. 3, modulation	$g_{s,3} = g_{s,2}$ $\Delta E_{s,3} = \Delta E_{s,2}$	

In this table, "minimize" refers to making a value as negative as possible and "maximize" refers to making a value as positive as possible.

3.4. Subtraction

A subnetwork that approximates linear subtraction of the form $U^*_{post} = k_{syn} \cdot (U_{pre,1} - U_{pre,2})$ may underlie negative feedback mechanisms, which are important for controlling many parameters in locomotion (Pearson, 1993; Peterka, 2003; Buschmann et al., 2015). Just as in the previous section, equation (18) is used to find g_s for each pathway.

Designing a subtraction network requires that we pay attention to how the two synapses affect one another. Since the reversal potentials of hyperpolarizing ion channels are not much more negative than typical resting potentials, larger $g_{s,2}$ values are required to transmit information than for depolarizing ion channels. This makes it harder to minimize g_s like we did in the previous section. Equation (13) enables us to constrain $g_{s,2}$ such that when $U_{pre,1} = R$ and $U_{pre,2} = R$, $U^*_{post} = 0$. Starting with the neuron response in equation (13) for two synaptic currents and no applied current,

$$U^*_{post} = \frac{g_{s,1}/R \cdot U_{pre,1} \cdot \Delta E_{s,1} + g_{s,2}/R \cdot U_{pre,2} \cdot \Delta E_{s,2}}{1 + g_{s,1}/R \cdot U_{pre,1} + g_{s,2}/R \cdot U_{pre,2}}. \tag{28}$$

Substituting in $U_{pre,1} = R$, $U_{pre,2} = R$, and $U^*_{post} = 0$,

$$0 = \frac{g_{s,1}/R \cdot R \cdot \Delta E_{s,1} + g_{s,2}/R \cdot R \cdot \Delta E_{s,2}}{1 + g_{s,1}/R \cdot R + g_{s,2}/R \cdot R} \tag{29}$$

$$0 = \frac{g_{s,1} \cdot \Delta E_{s,1} + g_{s,2} \cdot \Delta E_{s,2}}{1 + g_{s,1} + g_{s,2}} \tag{30}$$

$$0 = g_{s,1} \cdot \Delta E_{s,1} + g_{s,2} \cdot \Delta E_{s,2} \tag{31}$$

$$g_{s,2} = \frac{\Delta E_{s,1}}{\Delta E_{s,2}} \cdot -g_{s,1}. \tag{32}$$

Substituting equation (18) for $g_{s,1}$,

$$g_{s,2} = \frac{\Delta E_{s,1}}{\Delta E_{s,2}} \cdot \frac{-k_{syn} \cdot R}{\Delta E_{s,1} - k_{syn} \cdot R}. \tag{33}$$

To be physically realizable, $g_{s,2} > 0$. Because $g_{s,1} > 0$ and $\Delta E_{s,1} > 0$, $g_{s,2} > 0$ if and only if $\Delta E_{s,2} < 0$. Thus, it is critical that $\Delta E_{s,2} < 0$.

Just as for the addition network, we minimize $g_{s,1}$ by maximizing $\Delta E_{s,1}$. If $R = 20$ mV and $k_{syn} = 1$, then $g_{s,1} = 115$ nS and $\Delta E_{s,1} = 194$ mV. To tune $g_{s,2}$, we first select $\Delta E_{s,2} = -40$ mV, then we solve equation (33) to find $g_{s,2} = 558$ nS. These design constraints are summarized in **Table 1**, and **Figure 2B** graphically shows the accuracy of the subtraction network.

3.5. Division

A subnetwork that approximates division of the form

$$U^*_{post} = \frac{U_{pre,1}}{1 + \frac{1 - c_{syn}}{c_{syn} \cdot R} \cdot U_{pre,2}} \tag{34}$$

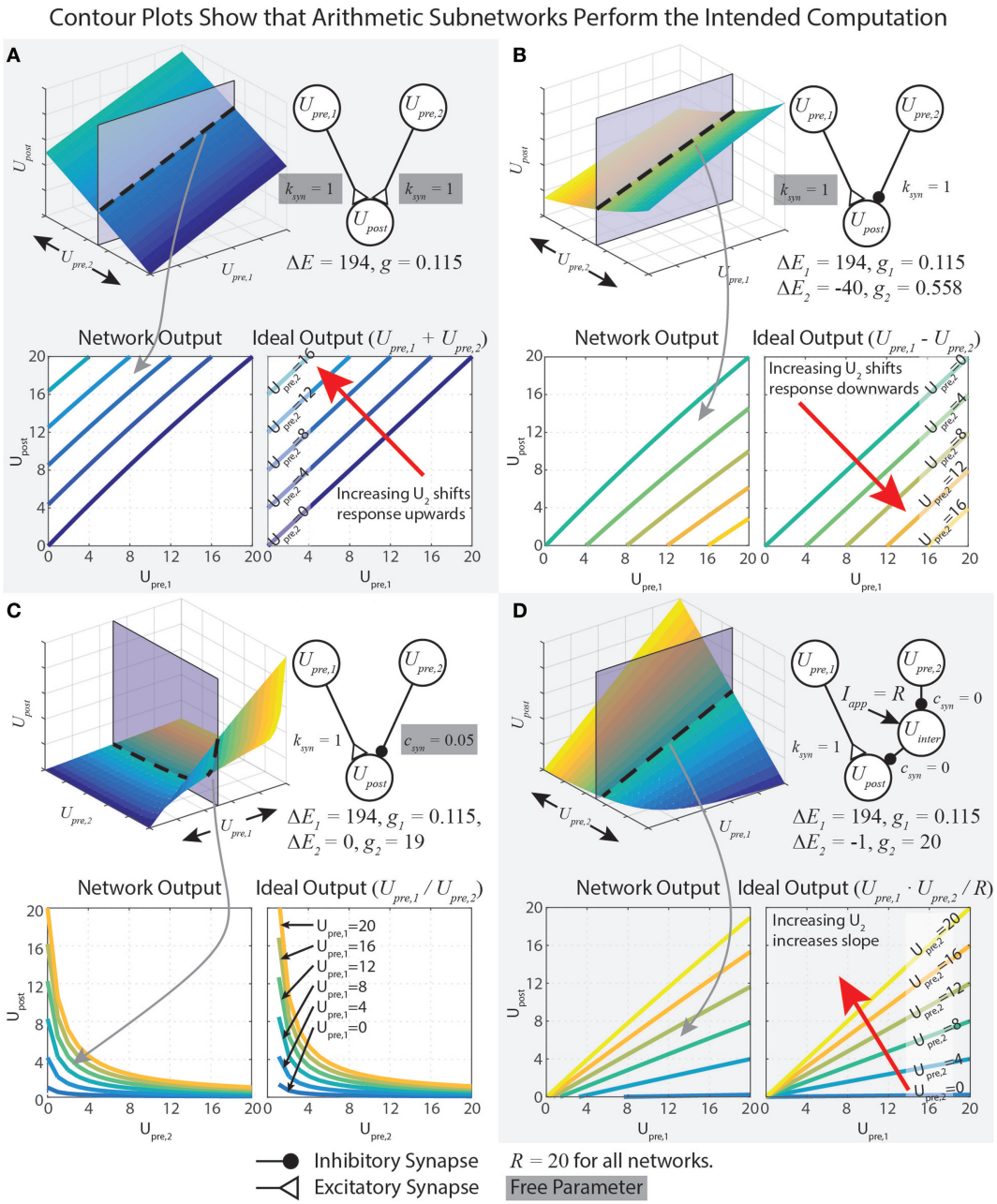

FIGURE 2 | Data demonstrating the function of arithmetic networks. Each contour plot represents cross sections of the response surface, as depicted at the top. The network diagram, relevant parameters, and data are shown for addition **(A)**, subtraction **(B)**, division **(C)**, and multiplication **(D)**. Triangular synaptic terminations stand for excitatory inputs, and filled round terminations stand for inhibitory inputs. For each operation, the contour on the right is the ideal output, and the contour on the left is the actual operation for the parameter values listed. Free parameters from **Table 1** are highlighted in gray.

replicates the function of GABA synapses that regulate activity in the brain. A key reason for this behavior is that the reversal potential of GABA-ergic synapses is about equal to the resting potential of the postsynaptic neuron (Trappenberg, 2009). Equation (26) is used to find g_s for the division pathway.

The synapse from $U_{pre,1}$ to U_{post} is tuned as an excitatory Signal Transmission pathway with $k = 1$, as in Sec. 3.1. In our work, $R = 20$ mV, $\Delta E_{s,1} = 194$ mV, and equation (18) tells us that $g_{s,1} = 115$ nS. Such a small g_s ensures that the signal from $U_{pre,1}$ to U_{post} is transmitted without greatly affecting the sensitivity of U_{post} to inputs. That is, the effect of $U_{pre,1}$ on the denominator of U^*_{post} is very nearly 0.

The synapse from $U_{pre,2}$ to U_{post} is tuned as a Signal Modulation pathway, as analyzed in Sec. 3.2. Setting $\Delta E_{s,2} = 0$ will eliminate $U_{pre,2}$'s influence on the numerator of U^*_{post}. Substituting this case into equation (26) and reducing,

$$g_{s,2} = \frac{1 - c_{syn}}{c_{syn}}, \tag{35}$$

where $U^*_{post} = c_{syn} \cdot R$ when $U_{pre,1} = U_{pre,2} = R$, their maximal value. Equation (35) also reveals that since $g_{s,2} > 0$, $0 < c_{syn} < 1$.

The steady-state response of the network is the result of these two synaptic inputs, as written in equation (28). Substituting equation (35), and specifying that $k_{syn,1} = 1$, U^*_{post} simplifies to

$$U^*_{post} = \frac{\cancel{g_{s,1}/R \cdot \Delta E_{s,1}}^{\,1} \cdot U_{pre,1} + g_{s,2}/R \cdot \cancel{\Delta E_{s,2}}^{\,0} \cdot U_{pre,2}}{1 + \cancel{g_{s,1}}^{\,0} \cdot U_{pre,1}/R + \frac{1-c_{syn}}{c_{syn}} \cdot U_{pre,2}/R} \approx \frac{U_{pre,1}}{1 + \frac{1-c_{syn}}{c_{syn} \cdot R} \cdot U_{pre,2}} \tag{36}$$

In our network, we wished $U^*_{post} = 1$ when $U_{pre,2} = R$, so we set $c_{syn} = 1/R = 0.05$, which makes $g_{s,2} = 19\,\mu$S. When c_{syn} is close to 0, $U_{pre,2}$ can strongly reduce U_{post}'s sensitivity to inputs. When c_{syn} is close to 1, $U_{pre,2}$ can only weakly reduce U_{post}'s sensitivity to inputs. **Figure 2C** shows that this network performs the intended division of the signals. **Table 1** summarizes these design constraints.

3.6. Multiplication

A subnetwork that approximates multiplication of the form $U^*_{post} = U_{pre,1} \cdot U_{pre,2}/R$ can be used to control the gain of a sensory feedback loop, a frequently observed characteristic of neural systems that control locomotion (Cruse, 1981; Gabriel and Büschges, 2007) and posture (Peterka and Loughlin, 2004).

A multiplication network can be assembled by replacing the Modulatory Pathway in the division network with two identical Modulatory Pathways in series, connected into a disinhibitory network (see **Figure 2D**). This works because the product of two numbers, $a \cdot b = a/(1/b)$. However, tuning the Modulatory Pathway for the multiplication network differs from tuning the division network. This is because the right-side pathway of the network in **Figure 2D** must make $U^*_{post} = 0$, no matter how active $U_{pre,1}$ becomes (because $a \cdot 0 = 0$, no matter the value of a). Thus, according to equation (22), $c_{syn} = 0$, unlike the division network, for which $0 < c_{syn} < 1$. Solving equation (26) when $c_{syn} = 0$ reveals that

$$g_{s,2} = -R/\Delta E_{s,2}. \tag{37}$$

To solve for $g_{s,2}$, we must first select $\Delta E_{s,2}$. If $\Delta E_{s,2} = 0$ like for the division network, then equation (37) divides by 0. If $\Delta E_{s,2} > 0$, then $g_{s,2} < 0$, which is physically not realizable. Therefore, we must choose a value $\Delta E_{s,2} < 0$. The more negative $\Delta E_{s,2}$ is, the more small-amplitude signals are clipped; however, the less negative it is, the larger $g_{s,2}$ must be. Therefore, $g_{s,2}$ is the limiting factor to maintain biological realism. We have chosen $g_{s,2} = 20\,\mu$S and $R = 20$ mV, making $\Delta E_{s,2} = -1$.

Now that we have designed one of the Modulatory synapses, we can calculate the response of the complete multiplication network seen in **Figure 2D**, which includes two identical Modulatory Pathways in series. When $U_{pre,2}$ is inactive, then it does not inhibit U_{inter}, which is tonically active. In this case, U_{inter}'s activity completely desensitizes U_{post} to inputs. When $U_{pre,2}$ is active, then it inhibits U_{inter}. In this case, U_{inter} is hyperpolarized, and cannot desensitize U_{post} to inputs. To show that this is the case, let us find the full response of the system. We first calculate U^*_{inter}, which has one Modulatory Pathway input and a tonic applied current $I_{app} = R$. Its response is the same as in equation (21), with the constraint from equation (37), which causes terms to cancel:

$$U^*_{inter} = \frac{\frac{g_{s,2}}{R} \cdot U_{pre,2} \cdot -\frac{R}{g_{s,2}} + R}{1 - \frac{U_{pre,2}}{\Delta E_{s,2}}} = \frac{R - U_{pre,2}}{1 - \frac{U_{pre,2}}{\Delta E_{s,2}}}. \tag{38}$$

U_{post} has two presynaptic neurons, $U_{pre,1}$ and U_{inter}. The synapse from $U_{pre,1}$ is a Signal Transmission synapse, and the synapse from U_{inter} is a Signal Modulation synapse. Its response is found via equation (13),

$$U^*_{post} = \frac{\frac{g_{s,3}}{R} \cdot U_{inter} \cdot \Delta E_{s,3} + \frac{g_{s,1}}{R} \cdot U_{pre,1} \cdot \Delta E_{s,1}}{1 + \frac{g_{s,3}}{R} \cdot U_{inter} + \frac{g_{s,1}}{R} \cdot U_{pre,1}}. \tag{39}$$

We showed in Sec. 3.3 that equation (18) can be used to design a synapse that transmits the presynaptic neuron's activity to the postsynaptic neuron, while minimizing its impact on the denominator of the postsynaptic neuron's steady-state response, U^*_{post}. This enables us to approximate $U_{pre,1}$'s effect on U^*_{post} as an applied current $I_{app} \approx U_{pre,1}$. Making this substitution in equation (39),

$$U^*_{post} \approx \frac{\frac{g_{s,3}}{R} \cdot U_{inter} \cdot \Delta E_{s,3} + U_{pre,1}}{1 + \frac{g_{s,3}}{R} \cdot U_{inter}}. \tag{40}$$

Because we previously specified that the Modulatory Pathways are identical, we can apply the constraint from equation (37),

$$U^*_{post} = \frac{U_{pre,1} - U_{inter}}{1 - \frac{U_{inter}}{\Delta E_{s,3}}}. \tag{41}$$

We can now substitute equation (38) for U_{inter},

$$U^*_{post} = \frac{U_{pre,1} - \frac{R - U_{pre,2}}{1 - \frac{U_{pre,2}}{\Delta E_{s,2}}}}{1 - \frac{1}{\Delta E_{s,3}} \cdot \frac{R - U_{pre,2}}{1 - \frac{U_{pre,2}}{\Delta E_{s,2}}}}. \tag{42}$$

This expression can be simplified. First, as noted previously, synapses 2 and 3 are identical, so $\Delta E_{s,2} = \Delta E_{s,3} = \Delta E_s$. Second, we can multiply the first term in both the numerator and denominator by the factor $(1 - U_{pre,2}/\Delta E_s)$, which enables us to combine terms. Performing these simplifications,

$$U^*_{post} = \frac{U_{pre,1} - U_{pre,1} \cdot U_{pre,2}/\Delta E_s - R + U_{pre,2}}{1 - U_{pre,2}/\Delta E_s - R/\Delta E_s + U_{pre,2}/\Delta E_s}, \tag{43}$$

$$U^*_{post} = \frac{-U_{pre,1} \cdot U_{pre,2}/\Delta E_s + U_{pre,1} + U_{pre,2} - R}{1 - R/\Delta E_s}. \tag{44}$$

Equation (44) contains a lot of information about how the multiplication network functions. First, U_{post}'s response indeed contains a term that multiplies $U_{pre,1}$ and $U_{pre,2}$. When $\Delta E_s = -1$, then U^*_{post} scales with $U_{pre,1} \cdot U_{pre,2}$ in a 1:1 fashion. Second, the numerator will be ≤ 0 if either $U_{pre,1} = 0$ or $U_{pre,2} = 0$, $U^*_{post} \leq 0$. This is because $U_{pre,1}$ and $U_{pre,2}$ must each be less than or equal to R. If either input is greater than R, then their synaptic inputs to U_{post} will saturate (see equation (4)), preventing this condition

from being violated. Third, the denominator does not depend on the input values. Technically, because of the approximation made in equation (40), the denominator does change slightly with $U_{pre,1}$. However, with our chosen values of R (20), ΔE_s (-1), and $g_{s,1}$ (0.115), this change is less than 1%, justifying this approximation. **Figure 2D** demonstrates that this network multiplies the two inputs.

Table 1 summarizes the function, component pathways, constraint equations, and free parameters of each network from this section. This analysis enables direct construction and parameter selection for functional subnetworks that can be assembled into more complex networks capable of performing real-time robotic control (e.g., Szczecinski and Quinn (2017) and Szczecinski et al. (2017a)). Additionally, one of the key advantages to using dynamic neural systems for motor control is the handling of time varying signals. The next section examines how the dynamics of these neurons can be exploited to perform calculus on signals.

4. METHODS: DYNAMIC NETWORKS

The differential equation for a single neuron's response (equation (1)) can be solved analytically. Solving an equation $dx/dt = f(x)$ is simplified if the equilibrium state is $x^* = 0$, so as in Sec. 3, the substitution $U = V - E_r$ is made. Additionally, the membrane conductance G_m and capacitance C_m can be combined into a new parameter $\tau = C_m/G_m$, which is a more intuitive parameter when discussing dynamic networks. This section uses analysis from the previous section, plus additional analysis, to derive design constraints for networks that differentiate or integrate input signals over time.

4.1. Differentiation

One dynamic response neural systems are known to utilize is differentiation of signals. Early examination of neural networks led to the discovery of the Reichardt detector network (Reichardt, 1961), an autocorrelation network with delays that approximates the differential of an incoming signal. Other examples include human balance, which relies on feedback proportional to the position, velocity, and acceleration of the center of mass (Peterka, 2003; Safavynia and Ting, 2012). Also, positive velocity feedback plays an important role in insect muscle control (Cruse, 1981).

We have developed differentiation networks based on the Reichardt detector network, shown in **Figure 3A**. We can understand its function by examining a neuron's response to a ramp input, $I_{app} = A{\cdot}t$, where A is an arbitrary slope of the ramp. The response of the network should be a step with a magnitude proportional to A, as shown in **Figure 3B**. Inserting this applied current into equation (6), a single neuron's response is

$$C_m \cdot \frac{dU}{dt} = -U + A \cdot t \tag{45}$$

$$C_m \cdot \frac{dU}{dt} + U = A \cdot t. \tag{46}$$

The response of the neuron, $U(t)$, is the sum of the particular and homogeneous solutions to equation (46), $U_p(t)$ and $U_h(t)$, respectively. Simulating the dynamics of equation (46) suggests

TABLE 2 | This table assumed that the designer has already selected a value of R for the subnetwork.

Operation	Components	Constraints and useful relations	Free parameters
Differentiation	Neuron 1	$C_{m,1} < C_{m,2}$	τ_d
	Neuron 2	$C_{m,2} = \tau_d$	k_d
		$C_{m,1} = C_{m,2} - k_d$	
	Syn. 1, transmission	$k_{syn} = 1/k_d$	$\Delta E_{s,1}$, maximize
		$g_{s,1} = \frac{k_{syn}\cdot R}{\Delta E_{s,1} - k_{syn}\cdot R}$	
		$\Delta E_{s,1} - k_{syn}{\cdot}R > 0$	
	Syn. 2, transmission	$g_{s,2} = \frac{\Delta E_{s,1}}{\Delta E_{s,2}} \cdot \frac{-k_{syn}\cdot R}{\Delta E_{s,1} - k_{syn}\cdot R}$	$\Delta E_{s,2}$, minimize
Integration	Neuron 1	$I_{app,1} = R$	$k_{i,\text{mean}}$
		$C_{m,1} = \frac{1}{2\cdot k_{i,\text{mean}}}$	
	Neuron 2	$I_{app,2} = R$	
		$C_{m,1} = C_{m,2}$	
	Syn. 1, transmission	$\Delta E_{s,1} = \frac{-R}{g_{s,1}}$	$k_{i,\text{range}}$
		$g_{s,1} = \frac{2\cdot C_{m,1}}{1/k_{i,\text{range}} - C_{m,1}}$	
	Syn. 2, transmission	$g_{s,2} = g_{s,1}$	
		$\Delta E_{s,2} = \Delta E_{s,1}$	

In this table, "minimize" refers to making a value as negative as possible and "maximize" refers to making a value as positive as possible.

that the particular solution is a ramp of slope A, which lags behind the input with a time constant C_m. To confirm this, we can substitute a candidate solution and its derivative into equation (46), and check for equality. The result is the particular (i.e., steady-state) response,

$$U_p(t) = A \cdot (t - C_m) \tag{47}$$

This means that if the same I_{app} were injected into neurons with different C_m values, and then their outputs were subtracted from one another with a network from Sec. 3.4, the network would perform a finite-difference approximation of the derivative of I_{app}, once the transient response decays (illustrated in **Figures 3A,B**).

Calculating the homogeneous solution, $U_h(t)$, informs us how quickly the transient response decays. The homogeneous solution to first-order linear equation like equation (46) is well-known, $U_h(t) = b{\cdot}\exp(-t/C_m)$. The constant b is found by plugging the initial condition into the full response, $U(t) = U_p(t) + U_h(t)$,

$$b = A \cdot C_m. \tag{48}$$

To tune this network, the response of U_{post} is written as the difference between neuron $U_{pre,1}$ with $C_{m,1}$ and neuron $U_{pre,2}$ with $C_{m,2} > C_{m,1}$,

$$\begin{aligned} U_{post}(t) &= U_{pre,1}(t) - U_{pre,2}(t) \\ &= A \cdot t - A \cdot C_{m,1} \cdot \big(1 - \exp(-t/C_{m,1})\big) \\ &\quad - \Big(A \cdot t - A \cdot C_{m,2} \cdot \big(1 - \exp(-t/C_{m,2})\big)\Big). \end{aligned} \tag{49}$$

Canceling the terms that are linear in t and expanding,

$$\begin{aligned} U_{post}(t) &= A \cdot (C_{m,2} - C_{m,1}) \\ &\quad + A \cdot \big(C_{m,1} \cdot \exp(-t/C_{m,1}) - C_{m,2} \cdot \exp(-t/C_{m,2})\big). \end{aligned} \tag{50}$$

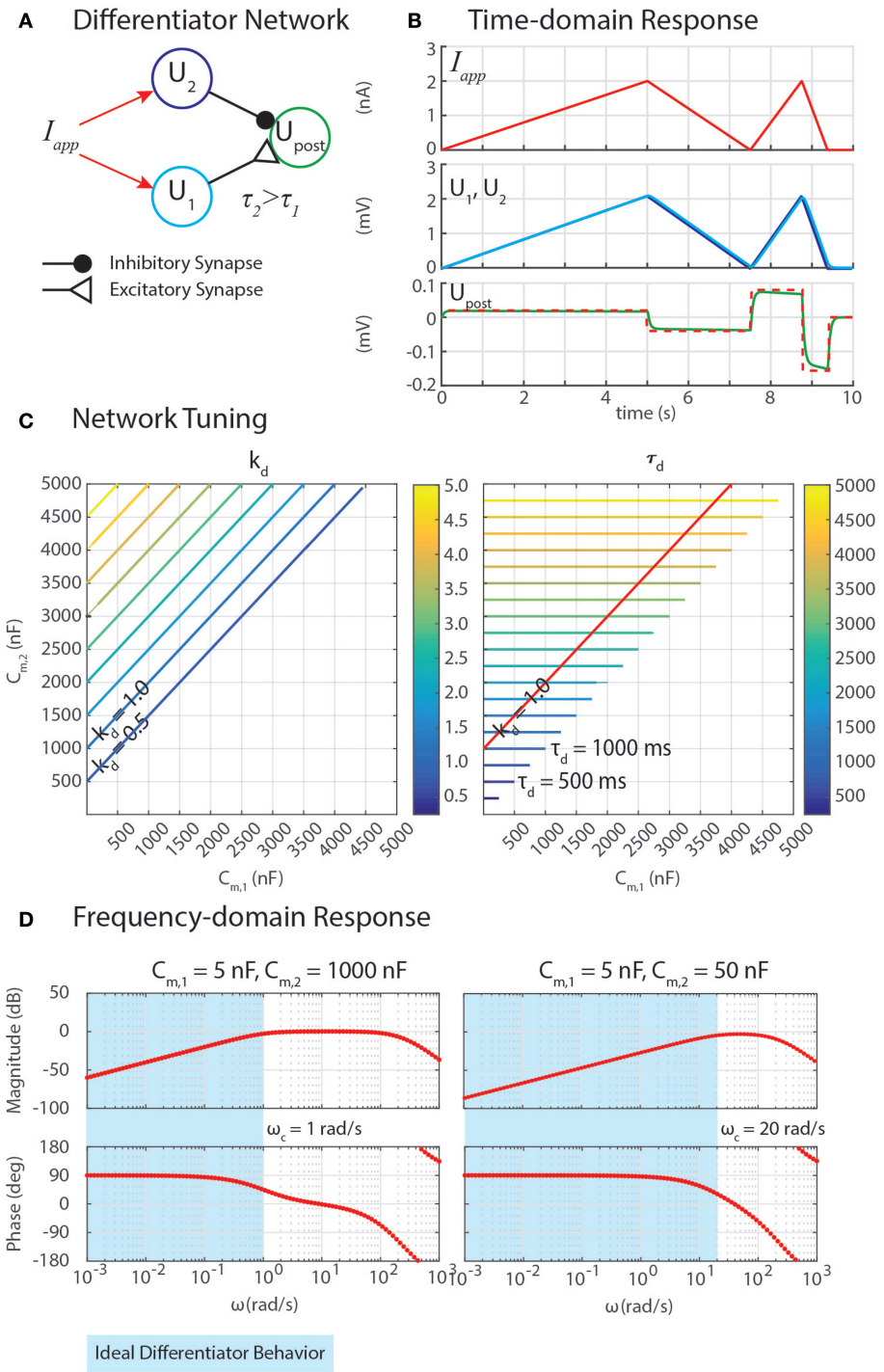

FIGURE 3 | **(A)** A network can exploit neural dynamics to compute the differential of an incoming signal. **(B)** When given an applied current in the form of ramps, the network returns steps whose heights are proportional to the slopes of the ramps. **(C)** The amplification of the differential, k_d, and the time constant of the network, τ_d, depend on the capacitance of the neurons, $C_{m,1}$ and $C_{m,2}$. **(D)** Frequency domain analysis enables the identification of the cutoff frequency ω_c, enabling the network to naturally filter out high-frequency noise.

Properly tuning a differentiator network requires tuning $C_{m,1}$ and $C_{m,2}$ to obtain the intended gain of the network, k_d, and an appropriately high cutoff frequency, ω_c. Equation (50) reveals how these may be tuned. First, the steady-state response of this network to a ramp input defines $k_d = (C_{m,2} - C_{m,1})$. Second, the cutoff frequency $\omega_c = 1/\tau_d$ quantifies the frequency of incoming signals (i.e., $I_{app} = A \cdot \sin(\omega \cdot t)$) above which the network's response has less than half the energy of a lower-frequency signal.

This is especially useful because although differential calculations amplify high-frequency noise, this network filters out noise with a frequency $\omega > \omega_c$. Because $C_{m,2} > C_{m,1}$, the time constant $\tau_d = C_{m,2}$.

Figure 3C shows contours of k_d and τ_d as $C_{m,1}$ and $C_{m,2}$ change. The plots show that increasing $C_{m,2}$ relative to $C_{m,1}$ increases k_d, which may be valuable for amplifying signals. However, this also increases τ_d, making ω_c impractically low, which will cause the network's output to lag behind the input substantially. The contour for $k_d = 1$ is drawn on the contour of τ_d, showing that the smallest τ_d achievable for this gain value is 1,000 ms, which would filter out all incoming signals for which $\omega > \omega_c = 1/(1\,s) = 1$ rad/s (0.159 Hz).

We can gain further insight into tuning τ_d using our FeedbackDesign tool (Szczecinski et al., 2017b). **Figure 3D** shows Bode plots for this network's response, given two different values for $C_{m,2}$. When $C_{m,2} = 1{,}000$ nF, like in **Figure 3B**, the network functions properly for inputs with $\omega < 1$ rad/s, as predicted in the previous paragraph. Lowering $C_{m,2}$ to 50 nF increases ω_c to 20 rad/s (3.18 Hz). Lowering $C_{m,2}$ also lowers the magnitude response as a function of ω, that is, it decreases k_d. To regain this lost gain, we may increase k_{syn} in the subtraction network. **Figure 4** shows simulation data that explores this tradeoff. **Table 2** lists how to use τ_d and k_d to tune the entire differentiation network.

4.2. Integration

Our neuron model is a leaky integrator, which means that the membrane voltage will integrate an applied current, but "leak" current to return to its resting potential. As a result, data cannot be stored in individual neurons, because neurons only have one stable equilibrium point. A network that is constructed to have a marginally stable equilibrium curve (or subspace) will not leak. A network will have this property if the determinant of the Jacobian matrix is 0, or in other words, if it is not full rank (Khalil, 2002). Instead of leaking, it will maintain its activation when no external currents are applied; when currents are applied, the state of the system will change continuously. This is analogous to the position of a box on a table with friction; it will remain wherever it is placed indefinitely, unless an external force is applied. In this section, we expand on previous work (Szczecinski et al., 2017a) to show how to construct a network that is marginally stable by applying constraints to reduce the rank of its Jacobian matrix; demonstrate that such a network can be used to integrate signals over time; and relate the integration rate, k_i, to the parameter values of the network, such that $\dot{U}_1 = k_i \cdot I_{app}$.

Marginally stable networks are hypothesized to play an important role in navigation (Haferlach et al., 2007) and the regulation of muscle forces in posture (Lévy and Cruse, 2008). Some memory models use carefully tuned self-excitation to cancel the leak current with excitatory synaptic current (Seung et al., 2000). In a similar vein, our network uses self-disinhibition (**Figure 5A**) to produce a line-attractor network in which a continuum of marginally stable equilibrium states exist. Simulation data in **Figure 5B** shows that stimulating U_1 with an applied constant current u causes U_1 to increase at an apparently constant rate, and when u is removed, neither U_1 nor U_2 leak to their rest potentials.

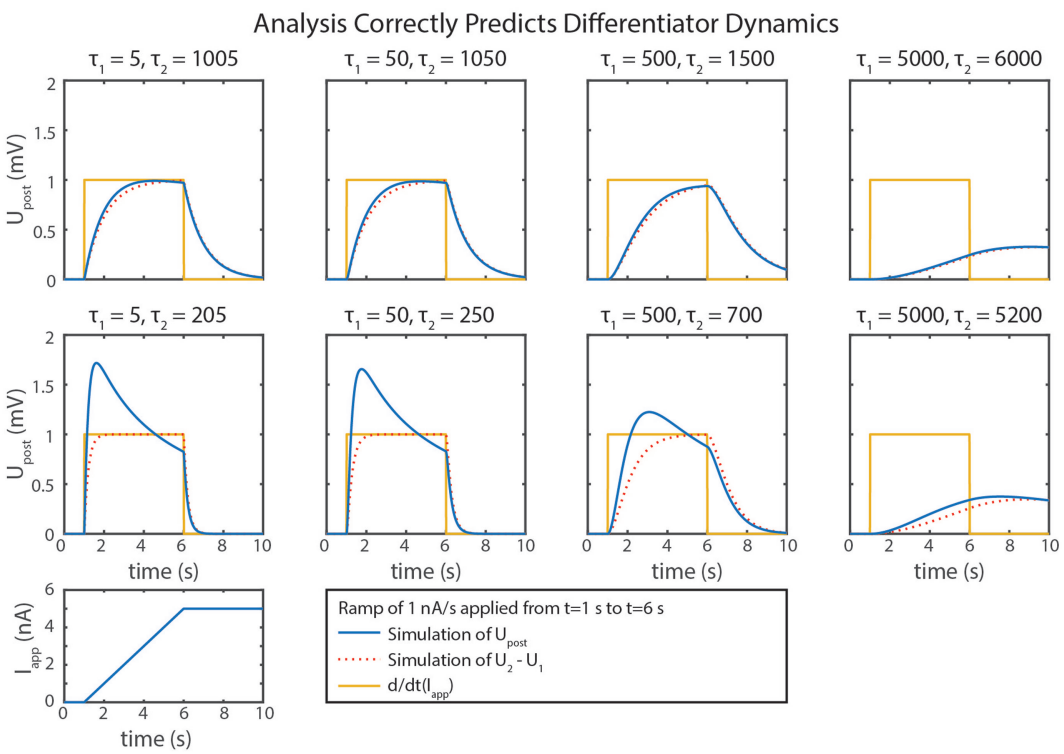

FIGURE 4 | Simulation data from eight trials with the differentiator network are shown. Different values of $C_{m,1}$ and $C_{m,2}$ were used in each. U^*_{post} is plotted in blue, $U_2 - U_1$ is plotted in dotted red, and the actual rate of change of the input, $d/dt(I_{app})$, is plotted in gold.

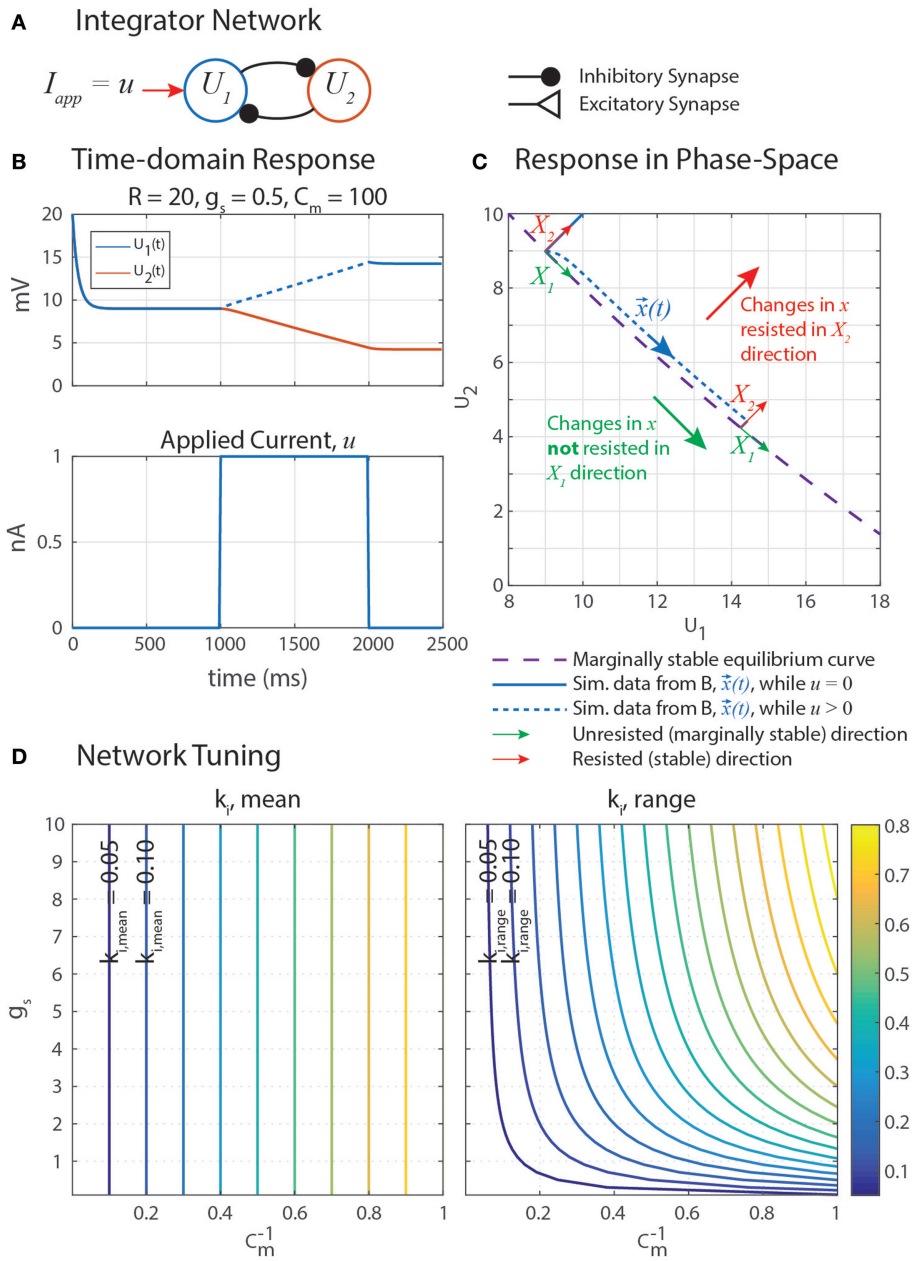

FIGURE 5 | **(A)** A disinhibitory network can exploit neural dynamics to compute the integral of an incoming signal. **(B)** When given an applied current in the form of a step, the network response is a ramp whose slope is proportional to the amplitude of the step. **(C)** A plot of this data in the (U_1, U_2) phase space shows that when stimulated by applied current u, the system state, $x(t) = [U_1(t), U_2(t)]^T$ (blue), moves in the X_1 direction (green) while maintaining a constant distance from the equilibrium subspace (dashed violet) in the X_2 direction (red). This difference in behavior in each direction is because the eigenvalue associated with eigenvector X_1, $\lambda_1 = 0$, and the eigenvalue associated with eigenvector X_2, $\lambda_2 < 0$. X_1 and X_2 are drawn in multiple places because they depend on $x(t)$, as shown in Appendix. **(D)** The mean rate of integration, $k_{i,\text{mean}}$ (left), and the range of the rate of integration, $k_{i,\text{range}}$ (right), depend on the synaptic conductance of mutual inhibition, g_s, and the membrane capacitance of the neurons, C_m. Note that the x-axis of these plots are $1/C_m$, to better space the contour lines.

This is the behavior of an integrator, as described in the previous paragraph.

Let us write the response of the integrator network as shown in **Figure 5A** to find its equilibrium states. Each neuron has leak current, synaptic current, and a constant applied current. Let all parameter values be symmetrical between the two neurons. We make the same substitutions as before; $U = V - E_r$, $E_r = E_{lo}$, $\Delta E_s = E_s - E_r$, and $R = E_{hi} - E_{lo}$. If $I_{app} = R$,

$$C_m \cdot \frac{dU_1}{dt} = -U_1 + g_s \cdot \frac{U_2}{R} \cdot (\Delta E_s - U_1) + R \qquad (51)$$

$$C_m \cdot \frac{dU_2}{dt} = -U_2 + g_s \cdot \frac{U_1}{R} \cdot (\Delta E_s - U_2) + R. \qquad (52)$$

Moving dynamical terms to the left hand side, and applied current to the right hand side,

$$\frac{dU_1}{dt} + \frac{1}{C_m}\left(U_1 - g_s \cdot \frac{U_2}{R} \cdot (\Delta E_s - U_1)\right) = \frac{R}{C_m} \tag{53}$$

$$\frac{dU_2}{dt} + \frac{1}{C_m}\left(U_2 - g_s \cdot \frac{U_1}{R} \cdot (\Delta E_s - U_2)\right) = \frac{R}{C_m} \tag{54}$$

Solving equation (53) when $dU_1/dt = 0$ reveals the equilibrium curve

$$U_2 = \frac{R \cdot (U_1 - R)}{g_s \cdot (\Delta E_s - U_1)}. \tag{55}$$

Solving equation (54) when $dU_2/dt = 0$ reveals the equilibrium curve

$$U_1 = \frac{R \cdot (U_2 - R)}{g_s \cdot (\Delta E_s - U_2)}, \tag{56}$$

which can be algebraically rearranged to be the same as equation (55) as long as g_s and ΔE_s are constrained such that

$$g_s \cdot \Delta E_s = -R. \tag{57}$$

Multiplying both sides of equation (56) by the denominator of the right hand side, and expanding,

$$g_s \cdot \Delta E_s \cdot U_1 - g \cdot U_1 \cdot U_2 = R \cdot U_2 - R^2. \tag{58}$$

Collecting multiples of U_2 and applying equation (57),

$$U_2 = \frac{R \cdot (U_1 - R)}{g_s \cdot (\Delta E_s - U_1)}. \tag{59}$$

Thus, equations (55) and (56) are the same equilibrium curve if g_s and ΔE_s satisfy equation (57). This curve, drawn on the phase-space diagram in **Figure 5C**, describes every equilibrium state that this network can have. In other words, a $[U_1, U_2]$ pair is an equilibrium state of the system if and only if it satisfies equation (55). In the coming paragraph, we will use eigenvalue analysis to show that this network always functions as an integrator, as long as equation (57) is satisfied.

To find the system's eigenvalues, let us write equations (53) and (54) together in matrix form,

$$\begin{bmatrix}\dot{U}_1\\ \dot{U}_2\end{bmatrix} + \frac{1}{C_m}\cdot\begin{bmatrix}1 + U_2 \cdot \frac{g_s}{R} & \frac{-g_s}{R} \cdot (\Delta E_s - U_1)\\ \frac{-g_s}{R} \cdot (\Delta E_s - U_2) & 1 + U_1 \cdot \frac{g_s}{R}\end{bmatrix}\cdot\begin{bmatrix}U_1\\ U_2\end{bmatrix} = \frac{1}{C_m}\cdot\begin{bmatrix}R\\ R\end{bmatrix}, \tag{60}$$

in which the square matrix is J, the system Jacobian. Because J contains U_1 and U_2 terms, it is not constant, but still describes the stability of the system, given specific values of U_1 and U_2. To construct a marginally stable equilibrium subspace for the network, we must show that J has insufficient rank (i.e., the rows are identical) when U_1 and U_2 are at equilibrium (i.e., equation (55) is satisfied). However, the rows are identical, no matter the values of U_1 and U_2, if we apply the constraint from equations (57) to (60),

$$\begin{bmatrix}\dot{U}_1\\ \dot{U}_2\end{bmatrix} + \frac{1}{C_m}\cdot\begin{bmatrix}1 + \frac{g_s}{R} \cdot U_2 & 1 + \frac{g_s}{R} \cdot U_1\\ 1 + \frac{g_s}{R} \cdot U_2 & 1 + \frac{g_s}{R} \cdot U_1\end{bmatrix}\cdot\begin{bmatrix}U_1\\ U_2\end{bmatrix} = \frac{1}{C_m}\cdot\begin{bmatrix}R\\ R\end{bmatrix}. \tag{61}$$

Thus, the system will always have one null direction, and we do not need to calculate J for specific equilibrium conditions to determine the system's stability. To make notation more compact, let us define

$$a = 1 + g_s/R \cdot U_1 \tag{62}$$

$$b = 1 + g_s/R \cdot U_2. \tag{63}$$

These expressions let us write equation (61) as simply

$$\begin{bmatrix}\dot{U}_1\\ \dot{U}_2\end{bmatrix} + \begin{bmatrix}b/C_m & a/C_m\\ b/C_m & a/C_m\end{bmatrix}\cdot\begin{bmatrix}U_1\\ U_2\end{bmatrix} = \begin{bmatrix}R/C_m\\ R/C_m\end{bmatrix}. \tag{64}$$

Plotting the simulation data of the network's forced response from **Figure 5B** on a phase-space diagram (**Figure 5C**) suggests that u causes U_1 and U_2 to change in such a way that the state of the system ($\vec{x}(t)$, blue) moves *tangent to* the equilibrium curve (dashed violet), with some constant distance *away* from it. These curves do not overlap because the forced response is not the same as the equilibrium condition while the external current u is applied. Motion in the X_2 direction is resisted by the neural dynamics, much how a spring resists the translation of an object with an applied force.

Nonetheless, these direction-dependent responses suggest that the state can be generalized into two decoupled degrees of freedom in the phase-space: unresisted, marginally stable motion parallel to the equilibrium curve (X_1, green in **Figure 5C**); and resisted, stable motion away from the equilibrium curve (X_2, red). The natural coordinates, $\vec{x} = [U_1, U_2]^T$, are transformed into generalized coordinates, $\vec{q} = [q_1, q_2]^T$, by a matrix X comprised of the eigenvectors of J. This same transformation matrix is used to transform J into the generalized coordinate system, yielding J_q. J_q is diagonal, decoupling the dynamics of the generalized coordinates and enabling us to quantify how quickly $\vec{x}$ moves parallel to the equilibrium curve.

Appendix shows the calculation of X, with q_1 representing the marginally stable mode and q_2 representing the stable mode. Using X, we can transform the system into generalized coordinates. First, we write the dynamics from equation (64) in a compact format.

$$\dot{\vec{x}} + J\vec{x} = \vec{F}, \tag{65}$$

where J is the square matrix in equation (64) and

$$\vec{F} = \begin{bmatrix}R/C_m + u/C_m\\ R/C_m\end{bmatrix}. \tag{66}$$

The generalized coordinates, $\vec{q}$, are defined as

$$\vec{x} = X\vec{q}. \tag{67}$$

To transform equation (65) into generalized coordinates, premultiply both sides of equation (65) by X^{-1},

$$\dot{\vec{q}} + J_q\vec{q} = \vec{Q}, \tag{68}$$

where $J_q = X^{-1}JX$ and $\vec{Q} = X^{-1}\vec{F}$. The top and bottom rows of equation (68) are decoupled because J_q is a diagonal matrix.

Furthermore, $J_q^{i,i} = \lambda_i$, meaning that $J_q^{1,1} = 0$, so the system simplifies even further.

To find the particular solution of this system, we can guess the form of $q_{p,1}$ and $q_{p,2}$, and substitute those in to equation (68). We observe that $\dot{q}_1(t) = B \cdot u$ in steady state, where B is a constant that relates $\dot{q}_1(t)$ and u. $q_1(t)$ would be the integral of $\dot{q}_1(t)$, but because the top row of J_q is zeros, it will not appear in the particular solution, and thus need not be explicitly included. We also observe that $\dot{q}_2(t) = 0$ in steady state, so $q_2(t) = D$, a constant. We can calculate $\vec{Q} = X^{-1}F$ using X^{-1}, which is calculated in Appendix (equation (A9)). Solving for the particular solution of this system, $\dot{\vec{q}}_p(t)$,

$$\dot{\vec{q}}_p(t) + J_q\vec{q}_p = \begin{bmatrix} B \cdot u \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & \frac{a+b}{C_m} \end{bmatrix} \cdot \begin{bmatrix} q_{p,1} \\ D \end{bmatrix}$$

$$= \begin{bmatrix} \frac{a \cdot d}{C_m \cdot (a+b)} \cdot u \\ \frac{R\sqrt{2}}{C_m} + \frac{b\sqrt{2}}{C_m \cdot (a+b)} \cdot u. \end{bmatrix} \quad (69)$$

$$B = \frac{a \cdot d}{C_m \cdot (a+b)}, \quad (70)$$

where d is defined in equation (A5). B describes how quickly $q_{p,1}$ varies with u, but we want to know how quickly U_1 varies with u. Therefore, we use equation (67) to transform $\vec{q}_p = [B \cdot u, 0]^T$ into natural coordinates to find $\dot{\vec{x}}$,

$$\dot{\vec{x}_p} = X\dot{\vec{q}_p} \quad (71)$$

$$\begin{bmatrix} \dot{U}_{1,p}(t) \\ \dot{U}_{2,p}(t) \end{bmatrix} = \begin{bmatrix} 1/d & 1/\sqrt{2} \\ -b/(ad) & 1/\sqrt{2} \end{bmatrix} \cdot \begin{bmatrix} \frac{a \cdot d}{C_m \cdot (a+b)} \cdot u \\ 0 \end{bmatrix} \quad (72)$$

$$\dot{U}_{1,p}(t) = \frac{a}{C_m \cdot (a+b)} \cdot u \quad (73)$$

$$k_i = \frac{a}{C_m \cdot (a+b)}. \quad (74)$$

Recall that a and b are functions of U_1 and U_2, respectively. This means that k_i, the integral gain of the network, is not a constant. To place bounds on k_i, let us substitute equations (62) and (63) into equation (74),

$$k_i = \frac{1 + g_s/R \cdot U_1}{C_m \cdot (2 + g_s/R \cdot (U_1 + U_2))}. \quad (75)$$

We can now plug in different values of U_1 and U_2 to see how k_i varies. Using equations (55) and (56), we find that the most extreme cases are when $[U_1, U_2] = [0, R]$ and $[U_1, U_2] = [R, 0]$. We can plug these cases into equation (75) to find the minimum and maximum values for k_i,

$$k_{i,\min} = \frac{1 + g_s/R \cdot 0}{C_m \cdot (2 + g_s/R \cdot (0 + R))} = \frac{1}{C_m \cdot (2 + g_s)} \quad (76)$$

and

$$k_{i,\max} = \frac{1 + g_s/R \cdot R}{C_m \cdot (2 + g_s/R \cdot (R + 0))} = \frac{1 + g_s}{C_m \cdot (2 + g_s)}. \quad (77)$$

The difference between $k_{i,\min}$ and $k_{i,\max}$:

$$k_{i,\text{range}} = \frac{1 + g_s}{C_m \cdot (2 + g_s)} - \frac{1}{C_m \cdot (2 + g_s)} = \frac{g_s}{C_m \cdot (2 + g_s)}. \quad (78)$$

To find the mean rate of integration, we can calculate $k_{i,\text{mean}} = (k_{i,\min} + k_{i,\max})/2$,

$$k_{i,\text{mean}} = \frac{1}{2 \cdot C_m}. \quad (79)$$

This is the same value of k_i obtained from computing k_i when $U_1 = U_2$. This simple expression is a useful relationship for tuning the integrator network. One may select C_m to obtain the intended mean integration rate, and then minimize the variation of the integration rate by minimizing g_s, as long as equation (57) is satisfied.

Figure 5D graphically demonstrates how $k_{i,\text{mean}}$ and $k_{i,\text{range}}$ determine C_m and g_s. Just as in equation (79), $k_{i,\text{mean}}$ is a function only of C_m. Therefore, the contour only shows vertical lines. The value of $k_{i,\text{range}}$ is minimized by decreasing either g_s or C_m^{-1} (i.e., increasing C_m). **Figure 6** shows simulation data of the integrator's response to a step input with eight different parameter value combinations. In every case, the change in U_1 is bounded by the values of $k_{i,\text{mean}}$ and $k_{i,\text{range}}$. As shown in **Figure 5D**, increasing C_m decreases the integration rate, and increasing g_s increases the variation in the integration rate.

Table 2 summarizes the design approach for this integrator network. The mean and range of the integration rate are free parameters that are determined by the intended network performance. Using these values and the constraint in equation (57), the neurons' C_m value and the synapses' g_s and ΔE_s values can be fully specified.

5. APPLICATION TO A ROBOT CONTROLLER

We have used the methods in this paper to tune (i.e., select parameter values for) several different networks that control robotic stepping (Szczecinski and Quinn, 2017; Szczecinski et al., 2017a) and visual tracking (Szczecinski et al., 2017a). Once a network layout is determined, whether hypothetical or based on neurobiological findings, individual subnetworks can be identified and tuned to work together. **Figure 7** shows a simplified joint-control network in which different functional pathways are color-coded. This illustrates how these functional subnetworks enable the direct assembly of control networks based on neurobiology. The neurobiological inspiration for these networks and the results of robotic experiments are presented in Szczecinski and Quinn (2017) and Szczecinski et al. (2017a), and so are omitted here.

The joint network in **Figure 7** uses three simple descending commands (body heading, stride length, and reference leg load) to control the walking motion of one joint of a leg. The descending commands modulate the output of a central pattern generator (CPG) to control the speed of the motion, and sensory feedback is used to adjust both the timing and amplitude of motor output. Addition pathways are drawn in red. These include the

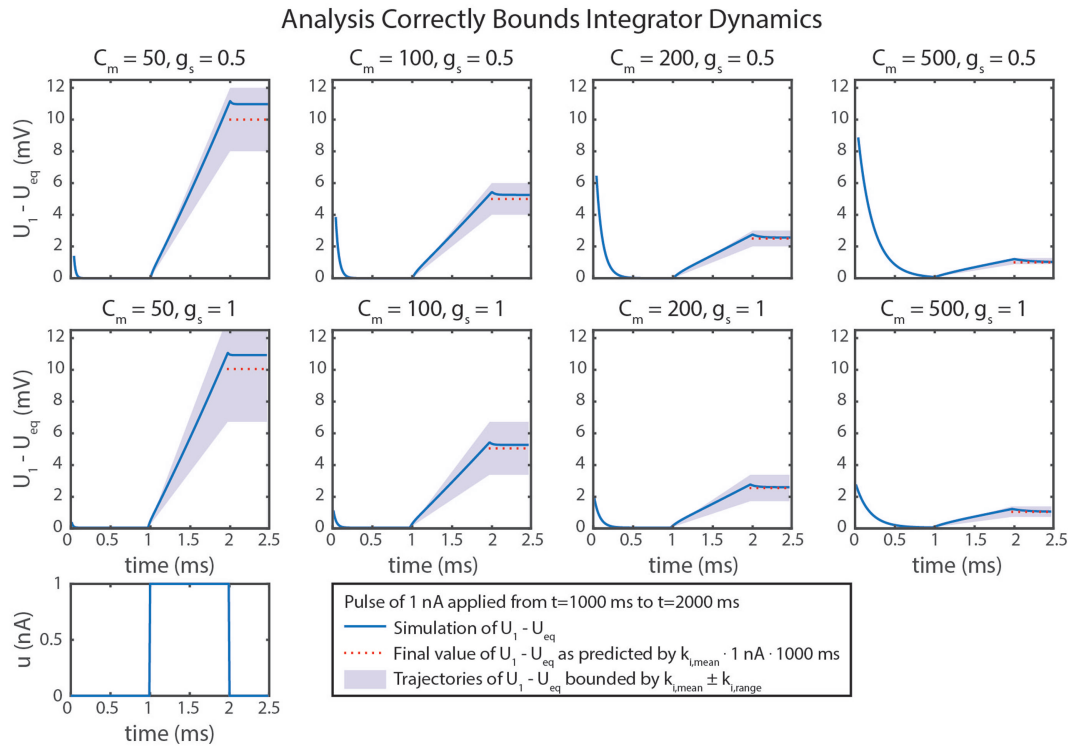

FIGURE 6 | Simulation data from eight trials are shown. Different values of C_m and g_s were used in each. Neural dynamics are plotted as blue lines. The expected final values of the simulations are plotted in dotted red lines. Regions bounded by $k_{i,\text{mean}} \pm k_{i,\text{range}}$ are shaded in violet. In every case, the actual outcome is correctly bounded. As demonstrated mathematically in the text, $k_{i,\text{mean}}$ only depends on C_m. In addition, $k_{i,\text{range}}$ depends on g_s, leading to more variation in k_i, as indicated by larger shaded areas.

mapping between body heading and stride length (i.e., descending commands, drawn in gray) to the PEP and AEP (Szczecinski and Quinn, 2017). The PEP can also be modulated by force feedback, which compares the load on the leg to a reference value (Szczecinski and Quinn, 2017, in review). This requires a subtraction network, drawn in orange, to compute if there is too much or too little load on the leg. The difference is used to adjust the PEP Memory network, which is an integration network, drawn in blue. This network adjusts the PEP over time, and remembers the motor command that produces the intended force.

The output of the CPG, drawn in purple, excites the motor neurons. Tuning CPG dynamics is discussed in our previous work (Szczecinski et al., 2017b). The PEP and AEP neurons adjust the motor output via multiplication pathways, drawn in green, which scale CPG output to the motor neurons based on the intended range of motion. Motor neuron activity controls the motor velocity, and the θ neuron receives position feedback from the motor via the mappings in **Figure 1B**. The motor velocity, computed by the cyan differentiation pathway, reinforces ongoing CPG behavior through the $\dot{\theta}$ neuron (Szczecinski et al., 2017b). A division pathway (not shown) can be used to normalize the velocity feedback to the joint's commanded range of motion, simplifying the control of stepping frequency. The $\dot{\theta}$ neuron also receives some input from the Load neuron, ensuring that stance phase is stable (Szczecinski and Quinn, 2017, accepted). By using the functional subnetworks and the design constraints presented in this paper, we can rapidly and directly assemble models of neural systems that perform as intended without hand-tuning or optimization methods.

How are the "Free Parameters" in **Tables 1** and **2** chosen? The free parameters fall into two classes: reversal potentials (i.e., ΔE_s) and dynamical constants (e.g., k, τ, etc.). The reversal potentials are informed by biology. In this paper, we kept $-40 < \Delta E_s < 194\,\text{mV}$ (i.e., $-100 < E_s < 134\,\text{mV}$). The modeler could use reversal potentials from specific synapses if that data were available. The dynamical constants are informed by the function of the robot. For example, the k_{syn} of the subtraction network in **Figure 1B** controls the stiffness of the controller, and may destabilize the system if not tuned to match the mechanical properties of the robot (Szczecinski et al., 2017b).

As another example, τ_d and k_d of the differentiator network in **Figure 7** determines the robustness of CPG rhythms, and how well it entrains to sensory feedback (Szczecinski et al., 2017b). A slow, adaptively-walking robot may want a high k_d to regularize CPG oscillations, whereas a fast running robot may want a low k_d to be less sensitive to sensory feedback. Picking specific values for these free parameters ultimately depends on the intended behavior of the robot. The constraints in this paper enable the designer to think in terms of more traditional robotics quantities, and use these to set neural and synaptic parameter values, which may be less intuitive.

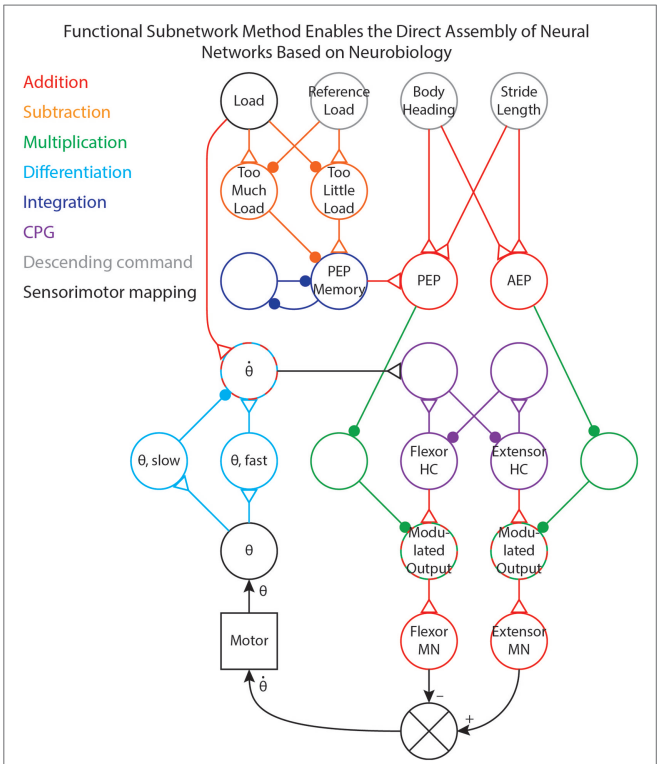

FIGURE 7 | A simplified joint-control network from our previous work (Szczecinski and Quinn, 2017; Szczecinski et al., 2017a), with pathways color-coded based on the functional subnetwork.

6. DISCUSSION

In this paper, we presented analytical methods for setting parameters in dynamical neural networks that can add, subtract, multiply, divide, differentiate, and integrate incoming signals. Such operations are at the core of control, and these techniques enable control networks to be assembled rapidly and tuned directly. This work primarily identifies constraint equations, not unique values, that govern how parameters should be tuned. Thus, many different networks may perform the same function with different parameter values, as observed in real neural circuits (Prinz et al., 2004). Since these results are analytical, not based on machine learning or optimization, there is no concern about these networks over- or under-fitting training data, and their behavior is provable. These techniques build on our previous analysis of synthetic nervous systems (Szczecinski et al., 2017b) and have been validated through several studies with our robot, MantisBot (Szczecinski and Quinn, 2017; Szczecinski et al., 2017a).

All of the results from this paper make it easier to tune neuromechanical models of animals, as well. Many such models have been created to study the principles underlying insect (Daun-Gruhn and Tóth, 2010; Szczecinski et al., 2014) and mammalian (Hunt et al., 2015a; Markin et al., 2016) locomotion alike. Oftentimes, parameters of these models are tuned by hand to obtain the intended motion, which is a painstaking, slow, and imprecise process. The analysis in this paper can make neuromechanical models come together more quickly, and have more predictable behavior, leading to more thorough scientific investigations. More precise tuning methods enable more thorough validation or invalidation of hypotheses. Faster tuning methods enable more rapid validation or invalidation of hypotheses. For example, these methods could be used to improve the coordination our previous cockroach model (Szczecinski et al., 2014). In the model, curve walking of varying radii was achieved by modulating muscle activations with broad descending commands. However, the coordination, reliability, and repeatability of such motion could be improved with the methods of this paper, enabling us to improve or reject the model.

6.1. Simplifications

Some of the calculations in this paper are based on approximations, which lead to inaccuracies in the calculations of the subnetworks. One example is that the subtraction network does not produce linear output. This non-linearity occurs because the reversal potentials of synapses are rarely much lower than the resting potentials of neurons, requiring large values of $g_{s,2}$ to build a subtractor where $k_{syn} = 1$. A large $g_{s,2}$ value increases $U_{pre,2}$'s effect on the denominator of U^*_{post}'s response, causing the synaptic input to reduce U_{post}'s sensitivity to inputs. This is particularly noticeable in the differentiator's response (**Figure 4**), especially as k_{syn} increases.

Another example of a simplification we made is that our calculation of k_i only used the particular solution of the system. This means that a transient response also exists, which we did not compute. In addition, k_i is a function of U_1 and U_2. This means that k_i is not a constant for this network. However, the impact of U_1 and U_2 on k_i can be minimized by minimizing g_s and maximizing R, as we showed in Sec. 4.2.

However, the developed networks are not intended to act as perfect analogs to their mathematical counterparts. These networks are intended to act as representations of real neural circuits, which likely do not act as perfect adders, multipliers, differentiators, etc. Dynamic and transient effects are a real part of biological control systems, and effective neural controllers have developed around these idiosyncrasies and have likely evolved to even exploit many of these aspects. In spite of these issues, the methods in this paper are valuable. Our recent robotics work (Szczecinski and Quinn, 2017; Szczecinski et al., 2017a), as well as related work in progress, is proof of the effectiveness of this approach.

6.2. Why Put Neurons in the Way?

The methods in this paper enable the direct construction of networks that perform arithmetic and dynamic calculations. Why bother building neural networks just to recreate mathematical operators? We believe there are several reasons to take this approach. From a neurobiology perspective, the constraints that we have identified may help explain why certain structures are common in the nervous system (David Friel, personal correspondence). For instance, mutually inhibitory parallel pathways are common in the thoracic control of insect locomotion (Büschges and Wolf, 1995), which may function as subtraction networks in negative feedback loops. As another example, networks in the retina of the rabbit are selectively sensitive to motion in one direction or the other (Barlow and Levick, 1965). Such a network could be constructed by using adjacent cells in the retina as

inputs to differentiator networks. This would be consistent with both the function of direction-sensitivity, as well as the laterally inhibitive structure. Even though such consistency does not guarantee that the animal's nervous system functions precisely this way, the design methods in this paper may aid in understanding the function of neural networks found in animals.

Additionally, the constraints that we identified may be used to constrain parameter values in large brain models. Rather than using global search techniques to understand the dynamics of a large pool of neurons, we believe it may be faster to begin with a number of functional subnetworks, and then use local search techniques to tune the connections between them. In this way, the designer is certain that parts of the network perform specific, useful computations, rather than naively optimizing a large network (Haferlach et al., 2007; Agmon and Beer, 2013; Izquierdo and Beer, 2013). The end result is something like a genetic program, but in a neuroscience context.

AUTHOR CONTRIBUTIONS

NS led research on functional subnetworks and led the preparation of the manuscript. AH aided in the preparation of the manuscript. RQ provided critical oversight of the research and aided in the preparation of the manuscript.

FUNDING

This work was supported by NASA Space Technology Research Fellowship NNX12AN24H, as well as a GAANN Fellowship.

REFERENCES

Agmon, E., and Beer, R. D. (2013). The evolution and analysis of action switching in embodied agents. *Adapt. Behav.* 22, 3–20. doi:10.1177/1059712313511649

Allen, T., Quinn, R., Bachmann, R., and Ritzmann, R. (2003). "Abstracted biological principles applied with reduced actuation improve mobility of legged vehicles," in *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 2 (Las Vegas, NV, USA: IEEE), 1370–1375.

Barlow, H. B., and Levick, W. R. (1965). The mechanism of directionally selective units in rabbit's retina. *J. Physiol.* 178, 477–504. doi:10.1113/jphysiol.1965.sp007638

Beer, R. D., and Gallagher, J. C. (1992). Evolving dynamical neural networks for adaptive behavior. *Adapt. Behav.* 1, 91–122. doi:10.1177/105971239200100105

Blickhan, R. (1989). The spring-mass model for running and hopping. *J. Biomed.* 22, 1217–1227. doi:10.1016/0021-9290(89)90224-8

Büschges, A., and Wolf, H. (1995). Nonspiking local interneurons in insect leg motor control. I. Common layout and species-specific response properties of femur-tibia joint control pathways in stick insect and locust. *J. Neurophysiol.* 73, 1843–1860.

Buschmann, T., Ewald, A., Twickel, A. V., and Büschges, A. (2015). Controlling legs for locomotion – insights from robotics and neurobiology. *Bioinspir. Biomim.* 10, 41001. doi:10.1088/1748-3190/10/4/041001

Cofer, D. W., Cymbalyuk, G., Reid, J., Zhu, Y., Heitler, W. J., and Edwards, D. H. (2010). AnimatLab: a 3D graphics environment for neuromechanical simulations. *J. Neurosci. Methods* 187, 280–288. doi:10.1016/j.jneumeth.2010.01.005

Cruse, H. (1981). Is the position of the femur-tibia joint under feedback control in the walking stick insect?: I. Force measurements. *J. Exp. Biol.* 92, 87–95.

Dasgupta, S., Goldschmidt, D., Wörgötter, F., and Manoonpong, P. (2015). Distributed recurrent neural forward models with synaptic adaptation and CPG-based control for complex behaviors of walking robots. *Front. Neurorobot.* 9:10. doi:10.3389/fnbot.2015.00010

Daun-Gruhn, S. (2010). A mathematical modeling study of inter-segmental coordination during stick insect walking. *J. Comput. Neurosci.* 30, 255–278. doi:10.1007/s10827-010-0254-3

Daun-Gruhn, S., Rubin, J. E., and Rybak, I. A. (2009). Control of oscillation periods and phase durations in half-center central pattern generators: a comparative mechanistic analysis. *J. Comput. Neurosci.* 27, 3–36. doi:10.1007/s10827-008-0124-4

Daun-Gruhn, S., and Tóth, T. I. (2010). An inter-segmental network model and its use in elucidating gait-switches in the stick insect. *J. Comput. Neurosci.* 31, 43–60. doi:10.1007/s10827-010-0300-1

Field, L. H., and Matheson, T. (1998). Chordotonal organs of insects. *Adv. In Insect Phys.* 27, 1–56, C1–C2, 57–228. doi:10.1016/S0065-2806(08)60013-2

Gabriel, J. P., and Büschges, A. (2007). Control of stepping velocity in a single insect leg during walking. *Philos. Trans. A Math. Phys. Eng. Sci.* 365, 251–271. doi:10.1098/rsta.2006.1912

Golowasch, J., Goldman, M. S., Abbott, L. F., and Marder, E. (2002). Failure of averaging in the construction of a conductance-based neuron model. *J. Neurophysiol.* 87, 1129–1131. doi:10.1152/jn.00412.2001

Haferlach, T., Wessnitzer, J., Mangan, M., and Webb, B. (2007). Evolving a neural model of insect path integration. *Adapt. Behav.* 15, 273–287. doi:10.1177/1059712307082080

Hunt, A., Szczecinski, N., and Quinn, R. (2017). Development and training of a neural controller for hind leg walking in a dog robot. *Front. Neurorobot.* 11:18. doi:10.3389/fnbot.2017.00018

Hunt, A. J., Schmidt, M., Fischer, M. S., and Quinn, R. D. (2015a). A biologically based neural system coordinates the joints and legs of a tetrapod. *Bioinspir. Biomim.* 10, 055004. doi:10.1088/1748-3190/10/5/055004

Hunt, A. J., Szczecinski, N. S., Andrada, E., Fischer, M. S., and Quinn, R. D. (2015b). "Using animal data and neural dynamics to reverse engineer a neuromechanical rat model," in *Biomimetic and Biohybrid Systems*, Vol. 9222 (Barcelona, Spain), 211–222.

Ijspeert, A. J. (2014). Biorobotics: using robots to emulate and investigate agile locomotion. *Science* 346, 196–203. doi:10.1126/science.1254486

Izquierdo, E. J., and Beer, R. D. (2013). Connecting a connectome to behavior: an ensemble of neuroanatomical models of *C. elegans* klinotaxis. *PLoS Comput. Biol.* 9:e1002890. doi:10.1371/journal.pcbi.1002890

Karakasiliotis, K., Thandiackal, R., Melo, K., Horvat, T., Mahabadi, N. K., Tsitkov, S., et al. (2016). From cineradiography to biorobots: an approach for designing robots to emulate and study animal locomotion. *J. R. Soc. Interface* 13, 1–15. doi:10.1098/rsif.2015.1089

Khalil, H. K. (2002). *Nonlinear Systems*, 3rd Edn. Upper Saddle River, NJ: Prentice Hall.

Lévy, J., and Cruse, H. (2008). Controlling a system with redundant degrees of freedom: II. Solution of the force distribution problem without a body model. *J. Comp. Physiol. A Neuroethol. Sens. Neural. Behav. Physiol.* 194, 735–750. doi:10.1007/s00359-008-0348-9

Marder, E., and Taylor, A. L. (2011). Multiple models to capture the variability in biological neurons and networks. *Nat. Neurosci.* 14, 133–138. doi:10.1038/nn.2735

Markin, S. N., Klishko, A. N., Shevtsova, N. A., Lemay, M. A. M. A., Prilutsky, B. I., and Rybak, I. A. (2016). "A neuromechanical model of spinal control of locomotion," in *Neuromechanical Modeling of Posture and Locomotion*, eds B. I. Prilutsky and D. H. Edwards (New York: Springer), 197–223.

Mittelstaedt, H. (1957). "Prey capture in mantids," in *Recent Advances in Invertebrate Physiology*, ed. B. T. Scheer (Eugene, Oregon: University of Oregon Publications), 51–72.

Pasemann, F., Steinmetz, U., Hülse, M., and Lara, B. (2001). Robot control and the evolution of modular neurodynamics. *Theory Biosci.* 120, 311–326. doi:10.1007/s12064-001-0025-9

Pearson, K. G. (1993). Common principles of invertebrates. *Annu. Rev. Neurosci.* 16, 265–297. doi:10.1146/annurev.ne.16.030193.001405

Peterka, R. J. (2003). Simplifying the complexities of maintaining balance. *IEEE Eng. Med. Biol. Mag.* 22, 63–68. doi:10.1109/MEMB.2003.1195698

Peterka, R. J., and Loughlin, P. J. (2004). Dynamic regulation of sensorimotor integration in human postural control. *J. Neurophysiol.* 91, 410–423. doi:10.1152/jn.00516.2003

Prinz, A. A., Billimoria, C. P., and Marder, E. (2003). Alternative to hand-tuning conductance-based models: construction and analysis of databases of model neurons. *J. Neurophysiol.* 90, 3998–4015. doi:10.1152/jn.00641.2003

Prinz, A. A., Bucher, D., and Marder, E. (2004). Similar network activity from disparate circuit parameters. *Nat. Neurosci.* 7, 1345–1352. doi:10.1038/nn1352

Reichardt, W. (1961). "Autocorrelation, a principle for the evaluation of sensory information by the central nervous system," in *Sensory Communication*, ed. W. A. Rosenblith (Cambridge: The MIT Press), 303–317.

Safavynia, S. A., and Ting, L. H. (2012). Task-level feedback can explain temporal recruitment of spatially fixed muscle synergies throughout postural perturbations. *J. Neurophysiol.* 107, 159–177. doi:10.1152/jn.00653.2011

Schilling, M., Hoinville, T., Schmitz, J., and Cruse, H. (2013). Walknet, a bio-inspired controller for hexapod walking. *Biol. Cybern.* 107, 397–419. doi:10.1007/s00422-013-0563-5

Schroer, R., Boggess, M., Bachmann, R., Quinn, R., and Ritzmann, R. (2004). "Comparing cockroach and Whegs robot body motions," in *IEEE International Conference on Robotics and Automation*, Vol. 4 (New Orleans, LA, USA: IEEE), 3288–3293.

Seung, H. S., Lee, D. D., Reis, B. Y., and Tank, D. W. (2000). The autapse: a simple illustration of short-term analog memory storage by tuned synaptic feedback. *J. Comput. Neurosci.* 9, 171–185. doi:10.1023/A:1008971908649

Szczecinski, N. S., Brown, A. E., Bender, J. A., Quinn, R. D., and Ritzmann, R. E. (2014). A neuromechanical simulation of insect walking and transition to turning of the cockroach *Blaberus discoidalis*. *Biol. Cybern.* 108, 1–21. doi:10.1007/s00422-013-0573-3

Szczecinski, N. S., Getsy, A. P., Martin, J. P., Ritzmann, R. E., and Quinn, R. D. (2017a). MantisBot is a robotic model of visually guided motion in the praying mantis. *Arthropod Struct. Dev.* doi:10.1016/j.asd.2017.03.001

Szczecinski, N. S., Hunt, A. J., and Quinn, R. D. (2017b). Design process and tools for dynamic neuromechanical models and robot controllers. *Biol. Cybern.* 111, 105–127. doi:10.1007/s00422-017-0711-4

Szczecinski, N. S., and Quinn, R. D. (2017). Template for the neural control of directed walking generalized to all legs of MantisBot. *Bioinspir. Biomim.* 12, 045001. doi:10.1088/1748-3190/aa6dd9

Trappenberg, T. (2009). *Fundamentals of Computational Neuroscience*, 2nd Edn. Oxford: Oxford University Press.

Zill, S. N., Schmitz, J., and Büschges, A. (2004). Load sensing and control of posture and locomotion. *Arthropod Struct. Dev.* 33, 273–286. doi:10.1016/j.asd.2004.05.005

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

APPENDIX

A. Derivation of Integrator Eigenvalues and Eigenvectors

We find the eigenvalues λ_1 and λ_2 and the associated eigenvectors X_1 and X_2 of the Jacobian matrix by the eigenvalue problem,

$$\det(J - \lambda_i \cdot I) = 0 \tag{A1}$$

$$J \cdot X_i = \lambda_i \cdot X_i, \tag{A2}$$

where i is the index of the eigenvalue (1 or 2), $I \in \mathbb{R}^{2\times 2}$ is an identity matrix, and J is the square matrix from equation (64). Solving for λ,

$$\lambda_1 = 0, \quad \lambda_2 = \frac{a+b}{C_m} > 0. \tag{A3}$$

Because J is on the same side of the equation as $\dot{\vec{x}}$ (see equation (65)), $\lambda_2 > 0$ indicates a stable system (e.g., as the stiffness matrix of a physical system). $\lambda_2 > 0 \forall \vec{x}$, because $a > 0$ and $b > 0$. The definition of a in equation (62) shows that $a > 0$ because $g_s > 0$ (it is a physical quantity) and $U_1/R \in [0, 1]$. The same reasoning applies to b.

We use the eigenvalues to find their associated eigenvectors,

$$X_1 = \begin{bmatrix} 1 \\ -b/a \end{bmatrix}. \tag{A4}$$

Normalizing X_1 to 1,

$$X_1 = \begin{bmatrix} \frac{1}{\sqrt{1^2+(-b/a)^2}} \\ \frac{-b/a}{\sqrt{1^2+(-b/a)^2}} \end{bmatrix} = \begin{bmatrix} 1/d \\ -b/(ad) \end{bmatrix}, \quad d = \sqrt{1^2 + (-b/a)^2}. \tag{A5}$$

Next, we calculate

$$X_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}. \tag{A6}$$

Normalizing X_2 to 1,

$$X_2 = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}. \tag{A7}$$

We now know the transformation matrix between the natural coordinates, $\vec{x} = [U_1, U_2]^T$, and the generalized coordinates, $\vec{q} = [q_1, q_2]^T$:

$$\vec{x} = X \cdot \vec{q}, \qquad X = [X_1, X_2] = \begin{bmatrix} 1/d & 1/\sqrt{2} \\ -b/(ad) & 1/\sqrt{2} \end{bmatrix}. \tag{A8}$$

We will also make use of X^{-1} when transforming between natural and generalized coordinates. We can analytically invert X from equation (A8),

$$X^{-1} = \frac{a \cdot d \cdot \sqrt{2}}{a+b} \cdot \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ b/(ad) & 1/d \end{bmatrix} = \begin{bmatrix} \frac{a\cdot d}{a+b} & \frac{-a\cdot d}{a+b} \\ \frac{b\cdot\sqrt{2}}{a+b} & \frac{a\cdot\sqrt{2}}{a+b}. \end{bmatrix} \tag{A9}$$