Encoding 3D Leg Kinematics Using Spatially-Distributed, Population Coded Network Model

Bohdan Zadokha(✉) and Nicholas S. Szczecinski

West Virginia University, Morgantown, WV 26506, USA
nss00001@mail.wvu.edu

Abstract. Controlling limbs in robotics is a nonlinear, complicated calculation that animals can do without significant effort. In this work we present a dynamical neural model with bioinspired sensory receptive fields which is capable of encoding the forward kinematics of a robotic leg. The model is implemented using the SNS-Toolbox. Synaptic conductance values are tuned using the Functional Subnetwork Approach. No optimization or machine learning is required. To understand how network construction affects encoding accuracy, we systematically varied the sensory neuron receptor functions, the number of sensory neurons, and neuron time constants. We use the root-mean-squared error to check the accuracy of the designed model. Finally, we show that our model with multiple outputs is more efficient than multiple networks with one output each.

Keywords: Dynamical Neural Network · Population Coding · Synthetic Nervous System · Functional Subnetwork Approach

1 Introduction

The control of limbs, whether for an animal or a robot, is a complicated task. For example, to accurately compute the position of the endpoint of the limb (e.g., hand, foot), sensory signals from across the limb must be integrated in a highly nonlinear way [1]. Furthermore, measurements taken along the limb may need to be transformed into the body's or head's frame of reference to be intelligible and useful for control, e.g., targeting placement of a hand or foot [2]. Despite the complexity of such calculations, animals solve problems of limb placement gracefully and without much apparent effort, suggesting that the nervous system has evolved mechanisms for computing (or approximating) such calculations. In this study, we create a dynamical neural model inspired by sensory receptive fields in insect limbs that may perform computations of this type.

In robotics, kinematics problems are solved by formulating vector and matrix equations [1]. For an open chain manipulator (e.g., a leg), the "forward kinematics" are computed by expressing the position and orientation of the end-effector in space as a function of the joint angles. Forward kinematics problems have a unique solution and can be expressed as a cumulative multiplication of transformation matrices [1]. Despite these

N. S. Szczecinski et al. (Eds.): Living Machines 2024, LNAI 14930, pp. 317–332, 2025.
https://doi.org/10.1007/978-3-031-72597-5_22

properties, their computation can be expensive and thus difficult to calculate as part of a real-time robot controller or biomechanical model. This challenge has motivated the utilization of artificial neural networks that reproduce forward kinematic solutions while effectively side-stepping the actual kinematics calculations. Researchers have trained and deployed such networks as part of robot manipulator controllers, enabling real-time control and, thus, the implementation of more sophisticated controllers [3]. Researchers have also used such networks to compute kinematic relationships within biomechanical models, e.g., the length of a muscle as a joint rotates, accelerating simulation and facilitating more rapid model improvement [4]. Such methods represent an advance in computation, but they require a lot of data and time for tuning, and performance cannot be guaranteed. Instead, we seek a model with structural and functional meaning that can be tuned rapidly and directly.

We also seek a model with biological relevance. In insects and other animals, sensory feedback is encoded by "range fractionated" sensory neurons [5–7], in which many neurons respond to the same mechanical state (e.g., joint angle) but with sensitivity to different "ranges". For example, some neurons are only active when the joint is extended; others are active only when the joint is flexed. This is believed to provide redundancy and, therefore resiliency in case of noisy sensory feedback or damaged sensory organs or neurons. We hypothesize that it also has computational advantages by enabling downstream networks to operate like a decoder rather than an analog circuit, which is prone to noise and miscalibration [8].

In this study, we apply tuning methods from our Functional Subnetwork Approach [9] to extend and analyze our previously developed spatially distributed population-coded network model [10]. The present study 1) demonstrates that this method can be applied to accurately encode 3-dimensional forward kinematics problems with no optimization or machine learning; 2) applies this method to compute multiple functions of the inputs using the same network, resulting in efficient computation of many nonlinear functions, simultaneously; and 3) quantifies network accuracy as the number of sensory neurons, their activation functions, and time constants throughout the network are altered. Furthermore, this study implements these models using the SNS-Toolbox [11].

2 Materials and Methods

2.1 Neural and Synaptic Models

The neural network was designed and simulated using the SNS-Toolbox library [11]. Each trial simulates 2 s of neural responses using Forward Euler integration with a timestep of 1 ms. All simulations were performed through JetBrains PyCharm 2022.3.3 (Professional Edition).

All neurons are modeled as leaky integrators whose dynamics follow the equation:

$$C\, {}^{dU}\!/_{dt} = -G(U - E_r) + I_{bias} + I_{syn} + I_{ext}, \tag{1}$$

where C is the membrane capacitance, G is the membrane conductance, U is the compartment's voltage, E_r is the resting potential of the compartment, I_{bias} is the constant bias current, I_{syn} is the synaptic current, and I_{ext} is an external current, for example,

the mechanotransductive current in a sensory neuron. In our network, all neurons have $E_r = 0$ and $G = 1$. Setting $E_r = 0$ simply shifts all compartment voltages such that the equilibrium voltage is 0 mV, instead of a physiological resting potential (e.g., $E_r \approx -60$ mV), which complicates tuning and analysis. Setting $G = 1$ effectively normalizes all synaptic conductance values to the leak conductance of that compartment.

Connections between neurons are non-spiking conductance-based chemical synapses, where the conductance is:

$$G_{syn} = G_{max} \cdot \max\big(0, \min\big(1, U_{pre}/R\big)\big), \tag{2}$$

where G_{max} is the maximum synaptic conductance, U_{pre} is the presynaptic neuron's voltage, and R is the operating range of each neuron's voltage.

Synaptic current has the following dynamics:

$$I_{syn} = G_{syn} \cdot \big(\Delta E_s - U_{post}\big), \tag{3}$$

where ΔE_s is the reversal potential relative to the postsynaptic rest potential (set to 20 mV for all synapses), and U_{post} is the postsynaptic neuron's voltage.

The maximum synaptic conductance can be related to the functional "gain" of the synapse, that is, $\mathrm{k_{syn}} = \mathrm{U_{post}/U_{pre}} \mathrm{k_{syn}} = \mathrm{U_{post}/U_{pre}}$, by the approximation:

$$G_{max} = \frac{k_{syn} R}{\Delta E_s - k_{syn} R}. \tag{4}$$

The derivation of this expression can be found in [9].

2.2 Network Architecture

For the neural network design, we incorporated and simplified several key features of insect thoracic sensorimotor networks. First, joint angles are encoded by range fractionated sensory neurons that form receptive fields (Fig. 1, yellow and orange circles 1–6) [5–7]. In our model, each sensory neuron has a Gaussian receptive field with a unique mean value (Fig. 2), which is not the most common receptive field in leg joint proprioceptor sensory neurons but can be found [5]. We chose Gaussian receptive fields, which are similar to visual sensory neurons [12, 13], because they facilitate network tuning (see discussion).

The interneuron compartments (Fig. 1, cyan circles 7–15) represent the dendrite of a single nonspiking interneuron (NSI). NSIs are large, branching neurons in insect thoracic sensorimotor networks that integrate sensory feedback from dozens or hundreds of sensory neurons and contribute to adaptive motor output [14–17]. We hypothesize that the way information is integrated may support the calculation of limb kinematics. To simplify the tuning and simulation of our model, compartments are not connected through resistive pathways (see discussion).

Finally, the output neurons (Fig. 1, red, green, and blue circles) represent motor neurons onto which the NSIs synapse. The synaptic connections to them (Fig. 1, red, green, and blue lines) ensure that the output neuron response reflects the quantity of interest in a graded way. Such a scheme could form part of a feedback pathway, in which the motor neurons are excited by sensory pathways and inhibited by other connections that specify the intended leg posture. This network architecture is inspired by motor control network in insects but may be extended to model many different neural structures.

All connections between neurons represent tuned conductance-based excitatory synapses. For a particular angle of θ_2, one of the yellow sensory neurons will be most active. In the same way, for a particular angle of θ_3, one of the orange sensory neurons will be most active. The interneuron compartment that is excited by both these neurons simultaneously will be the most active, so the synapse from that compartment to the output neuron will primarily determine the output neuron's activity level. We tuned each synapse conductance such that the output neuron activity encodes the actual position of the foot.

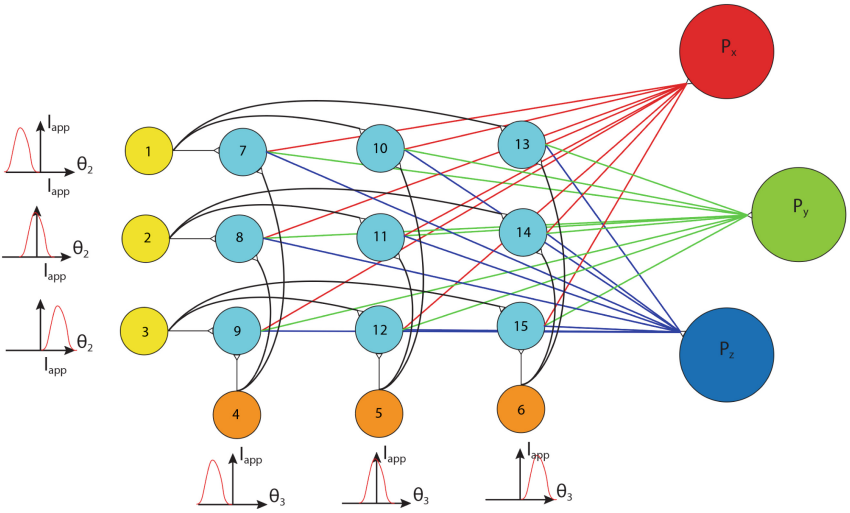

Fig. 1. Example of the designed neural network architecture with 3 sensory neurons per joint. Sensory neurons are represented in yellow and orange, interneuron compartments in cyan, and output neurons for the leg's x-, y-, and z-position are red, green, and blue, respectively. For the synapses between sensory neurons and interneuron compartments, we set Eq. (4)'s gain values for excitatory synapses equal to 1 (black lines). Synapses from interneuron compartments to output neurons were tuned using normalized x-position (red lines), y-position (green lines), and z-position (blue lines) of the foot from the forward kinematics calculation in Eq. (7). All figures are color-coded, please use an online version of the paper for a better experience.

To check the accuracy of the network prediction, we used the root-mean-squared error (RMSE) metric:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x}_i)^2}{N}}, \tag{5}$$

where x_i is the actual time series, $\bar{x}_i$ is the predicted time series, and N is the number of data points.

2.3 Population Coding of an Input

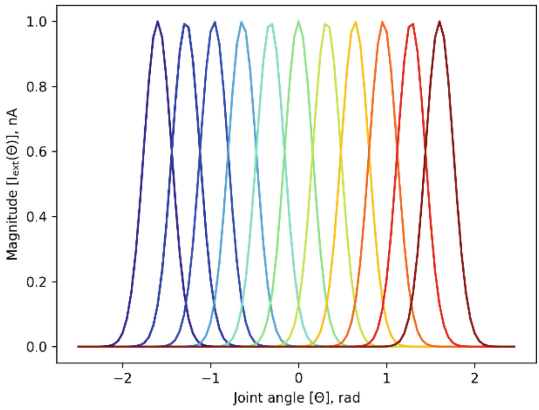

Fig. 2. An example of bell curve sensory encoding is used in the network. Each curve has magnitude A = 1 and width c = 20, but each curve's "mean" value b varies by 0.32 radians from the others. In this way, the activity profile of the neuron signals represents the rotation of the joint.

For every joint, we defined sensory neurons to get an encoded implementation of an input. As an activation function, we used the Gaussian bell curve equation:

$$I_{ext}(\theta) = A \cdot \exp\left(-c \cdot (\theta - b)^2\right), \tag{6}$$

where A is the magnitude, b controls the mean value, and c controls the width of the curve.

2.4 Synaptic Conductance Tuning

To calculate how joint angles map to the position of the end effector, we solved a forward kinematics problem in 3D space with joint angles ranging from −1.6 to 1.6 radians using the product of exponentials formula [1]:

$$T = e^{S_{n-1}\theta_{n-1}}\left(e^{S_n\theta_n}M\right), \tag{7}$$

where T is a 4x4 matrix that represents the position and orientation of the end effector, S is the screw axis matrix, θ is the joint rotation, and M is the position and orientation of the end effector in the "zero configuration", when all joint angles equal 0 (Fig. 3).

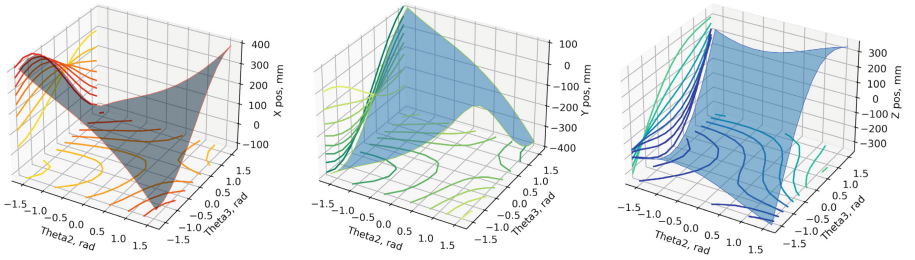

Fig. 3. X-, y-, and z- coordinate position surfaces for θ_2 and θ_3 joint angles

To find all exponential matrices, we used Rodrigues' rotation formula:

$$e^{S\theta} = I + \sin\theta[\widehat{S}] + (1 - \cos\theta)[\widehat{S}]^2, \tag{8}$$

where $[\widehat{S}]$ is a skew-symmetric representation of a screw axis matrix S.

Since every synaptic connection is excitatory, synaptic reversal potential values must be positive. Furthermore, synaptic conductance must always be positive. To comply with these conditions, we normalized all data from the forward kinematics to set gain values for the synapses between interneuron compartments and output neurons:

$$k_{synX} = \frac{P_X - min(P_X)}{max(P_X) - min(P_X)}, \tag{9}$$

where P_X is a calculated x-position. This was repeated for P_y and P_z. Note that no training dataset is required because the parameters are tuned based on the function of the network.

3 Results

To test our approach, we used a robotic leg from the Trossen Robotics PhantomX Hexapod MK-IV (Fig. 4). We used the front left leg, which is rotated 45 degrees about the Z-axis such that rotating either θ_2 or θ_3 changes the X, Y, and Z coordinate of the foot.

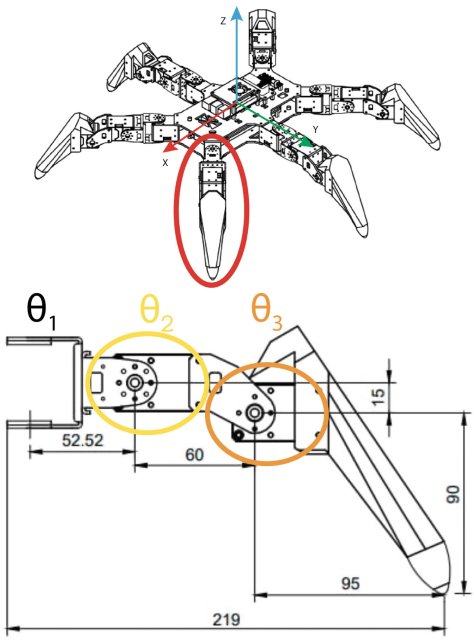

Fig. 4. The drawing of the PhantomX Hexapod MK-IV with the origin at the center of a robot's body (I) and a leg configuration (II). θ_1 servo was fixed, and only θ_2 and θ_3 were used as inputs to the network. Measurements of the leg were used for the forward kinematics calculation.

The sensory neurons accurately encode the joint angles using the Gaussian activation function (Eq. 6) into neuron voltages (Fig. 5). In this experiment, there are 11 sensory neurons that encode one joint, and each Gaussian "bell curve" function has a parameter value c = 20 and magnitude equal to 1. As θ_2 changes, which neuron is most active changes in a corresponding way.

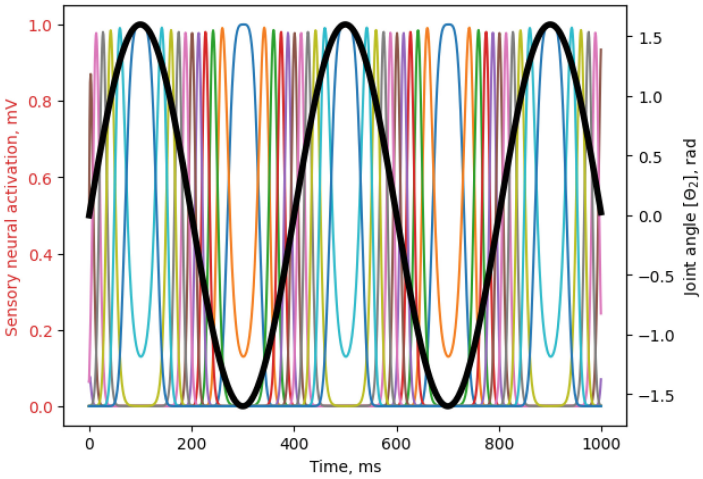

Fig. 5. Bell curve encoded responses (colored lines) of the θ_2 input (black line). Each color represents a sensory neuron response for the specific joint angle.

Each interneuron compartment represents one of the possible combinations of the joint angles for θ_2 and θ_3. It implies that interneuron compartments can be active only if both inputs from θ_2 and θ_3 are active at the same time or if activation of one of those occurs right after another (Fig. 6). The threshold for synaptic activation for each interneuron compartment is 1 mV, which is equal to the magnitude of the Gaussian activation function. Using this threshold, we minimize situations when two or more

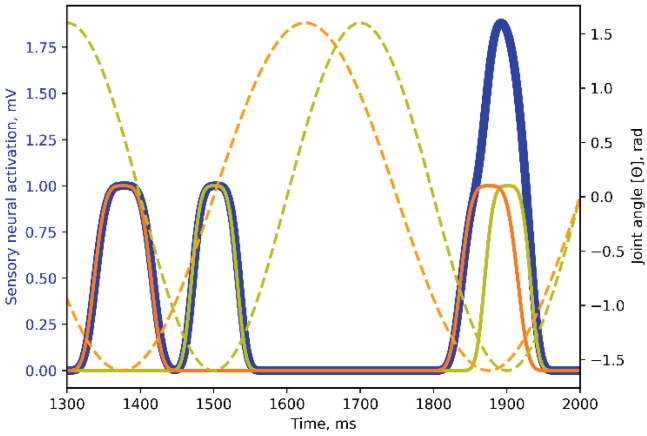

Fig. 6. The interneuron compartment (blue) activation with response to the $\theta_2 = \theta_3 = -1.6$ rad input values (solid yellow and orange lines respectively). Dashed yellow and orange lines represent inputs of θ_2 and θ_3 joint angles.

interneuron compartments are active simultaneously in response to the same sensory inputs.

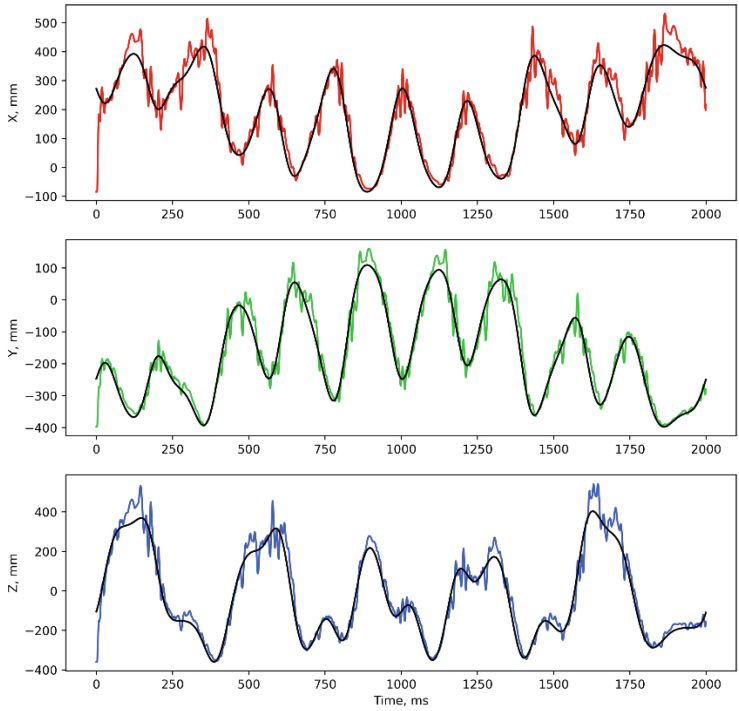

Fig. 7. The network's predicted x-position (red), y-position (green), and z-position (blue) compared to the forward kinematics actual result (black). RMSE for x-, y-, and z-position are 41.76, 31.74, and 51.98 respectively.

By using optimal parameters for the number of sensory neurons per joint and width of the activation function, the network activity tracks P_x, P_y, and P_z for the robot foot (Fig. 7). The neural activity exhibits some fluctuation over time but clearly follows the position of the foot on average. To understand the impact of different network parameters, we tested combinations of bell curve width (c) and the number of sensory neurons per joint (N). We found that for every activation function, there is a specific number of sensory neurons that produces the smallest error (Fig. 8), which we call an "optimal range." For narrower activation functions (higher value of the width parameter c) the RMSE is less sensitive to the number of sensory neurons per joint (N). The best parameters for all x-, y-, and z-positions are $N = 11$ sensory neurons per joint, with the bell curve width $c = 20$ (Fig. 8, black line). We refer to these as the "optimal parameters" throughout the rest of the results section. This model contains 146 neurons/neuron compartments in total (22 sensory neurons, 121 interneuron compartments, and 3 output neurons).

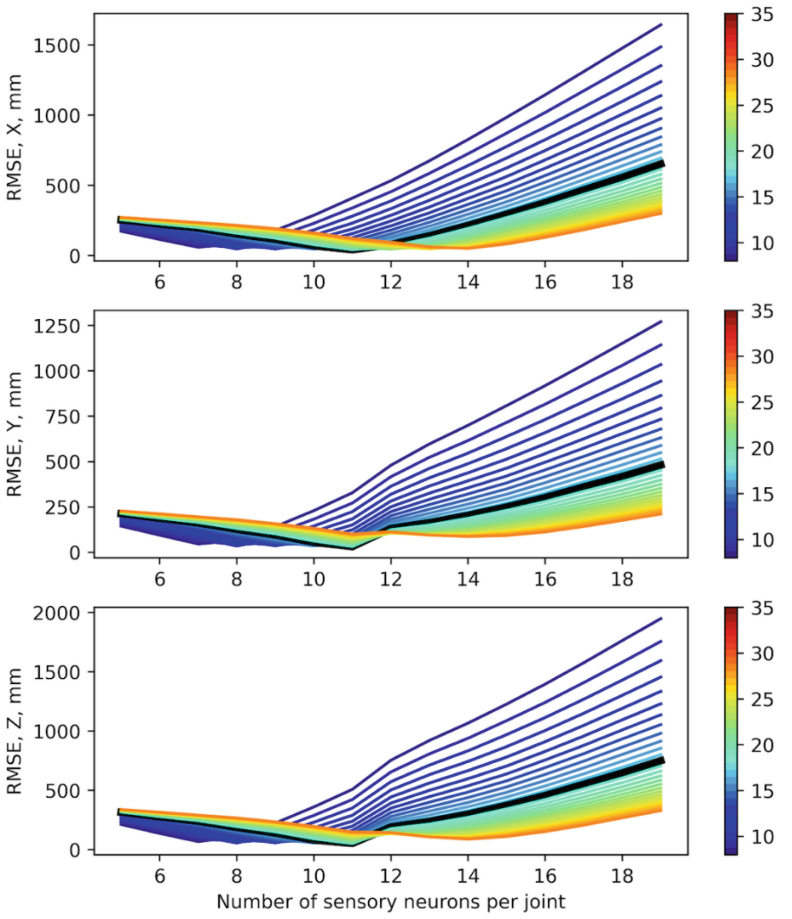

Fig. 8. RMSE of x-, y-, and z-positions for different combinations of bell curve's width (color lines) and number of sensory neurons. Bell curve width (c) varies from 8 to 34 (a smaller number of the bell curve width means a wider activation function), and the number of neurons – varies from 5 to 19. Optimal parameter values for the activation function's width are shown in black lines.

Another parameter that affects output is the neuron's membrane capacitance. We calculated RMSE by ranging values from 2 to 40 nF and found that increasing membrane capacitance increases the error, presumably due to increased lag in the response (Fig. 9). Past 40 nF, the RMSE hardly increases.

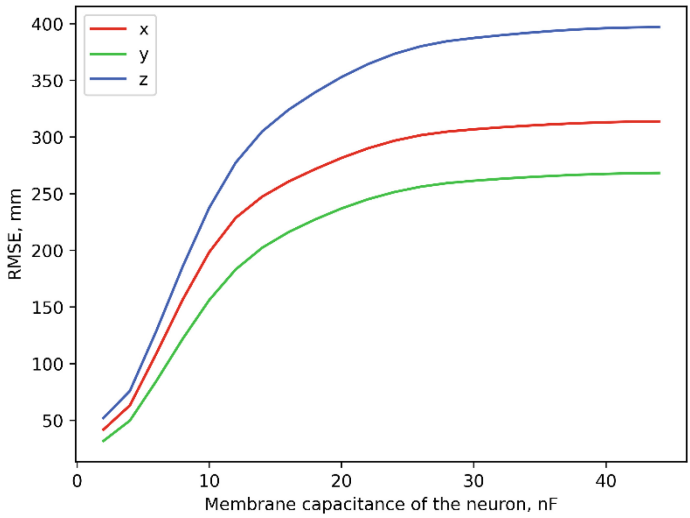

Fig. 9. The RMSE of x-, y-, and z-positions with different values for membrane capacitance of neurons.

To investigate the statistical nature of the network's accuracy, we calculated the difference between the predicted output and the actual calculation at each instant in time and plotted histograms with error distributions (Fig. 10). All three outputs have a bell-curve-like distribution with mean and median values for x-position: −9.4; −10.3, y-position: −7.1; −8.5, and z-position: −15.6; −17.3 respectively. The negative sign indicates peak overshoot (Fig. 7).

In addition, the linear regression slope between actual and predicted signals at each instant in time was found (Fig. 11). This parameter characterizes the correlation between signals: if the slope is equal to 1, it indicates a positive, 1:1 correlation between signals and, thus, accurate network output. Using optimal parameters, slopes for the regression are 1.02, 1.00, and 1.04 for x-, y-, and z-coordinates, respectively. This parameter can also be combined with the RMSE to find optimal network parameters.

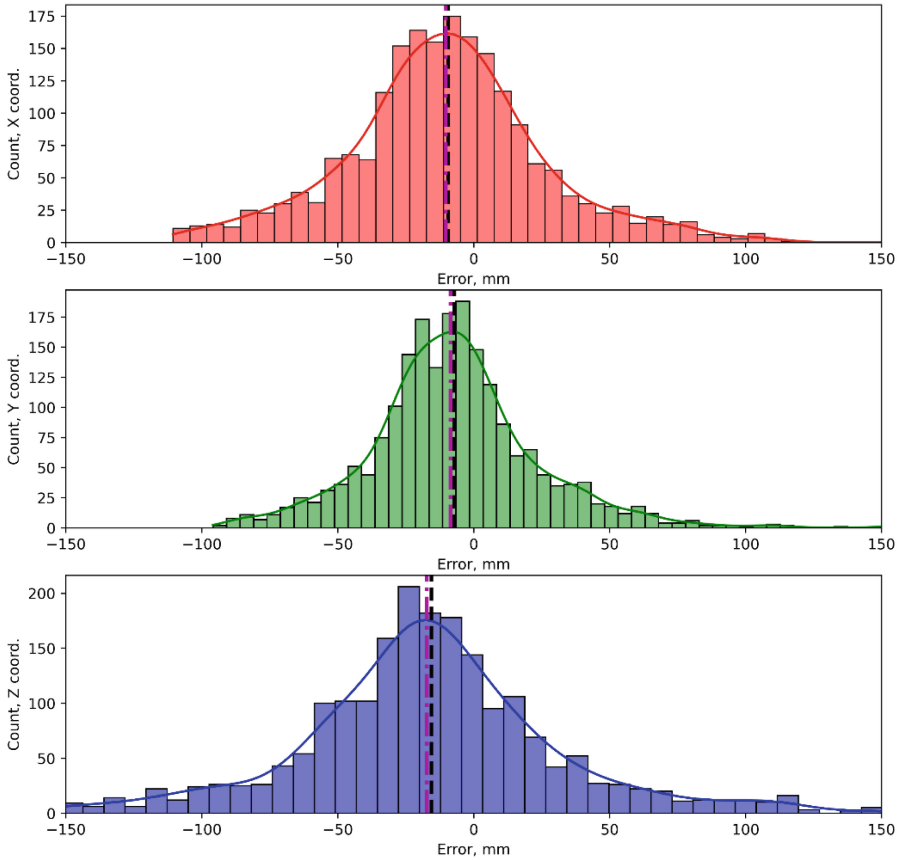

Fig. 10. Error distribution for x- (red), y- (green), and z-coordinate (blue) of the leg. Dashed black lines indicate mean values; dash-dotted magenta lines indicate median

To check the efficiency of the simultaneous multidimensional computation, we compared the running time of the designed network, which has 3 outputs for x-, y-, and z-positions with 3 separate networks that have the same architecture but only 1 output: for x-, y-, and z-positions respectively. Running simulations on the Apple M1 SoC, 3.2 GHz, 7 cores GPU, 16 GB unified memory, we found that the network with 3 output neurons performed calculations on average of 0.6 s for 10 trials, and the 3 networks with one output neuron required 2 s for 10 trials. This suggests that using the same nonspiking compartments to compute multiple quantities would be more efficient than building a separate network for each quantity.

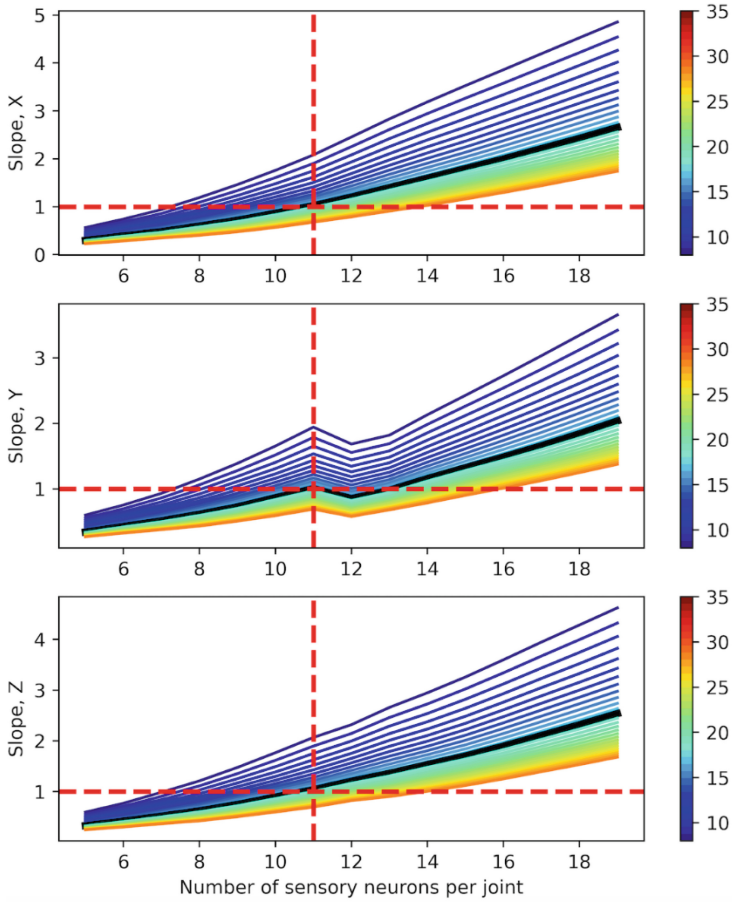

Fig. 11. Slopes of the linear regression between actual and predicted signals for different combinations of activation function's width (color lines) and number of sensory neurons. A dashed horizontal line shows the ideal positive correlation between signals, and a dashed vertical – the number of sensory neurons used in the paper. The black line shows an optimal width parameter.

4 Discussion

We designed a non-spiking neural network using SNS-Toolbox to calculate forward kinematics for the 2 DoF robotic leg in 3D space. Gaussian bell curve was used as an activation function to encode mechanical signals into sensory neuron activation. We used normalized position values from the product of exponentials to tune synaptic connections. No machine learning or optimization was required. By varying the number of sensory neurons and bell curve width, we found parameters that give the best result in forward kinematics computation. We found that the same network could be used to compute multiple functions of the inputs, in this case, separate representations of the x-, y-, and z-coordinate of the robotic leg's foot. We discovered that there is an "optimal

range" for the different sets of parameters, but it is unclear why it changes. We suggest that one reason is the overlapping of activation functions. In the designed network, only two neighboring functions may be active at each instant in time. Investigating an optimal range, how it can be found, and its behavior will be useful for future work, such as implementing a designed network into a robot. Since minimum error values were achieved in all three output signals using the same parameters, we think that finding conditions for an optimal range for one activation function will speed up the tunning process for the whole network regardless of the number of inputs and outputs. As a statistical analysis, we calculated a correlation between the predicted and actual signal using linear regression. Alongside RMSE, the slope of the regression may be used in tunning parameters of the network. We compared the running time for the 3-dimensional neural network with that for 3 1-dimensional networks and found that simulation time largely does not depend on a number of outputs once the bulk of the network is constructed. We believe this method can be used in other high-dimensional nonlinear computations, e.g., altering descending commands to change the robot's walking direction [18]. It also does not require large datasets for training which reduces the running time for simulation.

Despite the success of our approach, there are limitations to this study. First, it may be possible to achieve higher accuracy if machine learning methods were used to fine-tune the network we constructed here. Recent work has shown that SNS models are generalized representations of several different artificial neural network models, and has successfully applied backpropagation to tune synaptic weights over time [19]. We anticipate that the synaptic weights we calculated here would greatly accelerate learning more precise values.

Another limitation is the computational complexity of this network. A model with n input states and m sensory neurons per input would require m^n interneuron compartments. This may result in unwieldy networks for limbs with many joints or in cases where many sensory neurons are required for each joint. In the future, we will investigate ways to decrease the number of sensory neurons required. One possibility is that by increasing the width of sensory encoding functions (i.e., decreasing c) or by using different encoding functions (e.g., sigmoids), the network may achieve "hyperacuity" as has been described in other sensory systems (e.g., visual system [12, 13]). Changing the encoding function in this way would require us to adapt the tuning method, but would likely have additional benefits, e.g., the incorporation of resiliency against broken sensors or noisy inputs.

Having demonstrated that our approach works for three different, highly-nonlinear functions, we believe this approach can be applied to other functions, for example, computing Jacobians for transforming sensory data from one reference frame to another. Another example is recent experimental work suggesting that an insect's central brain can transform "vectors" of sensory information to convert egocentric exteroceptive cues into allocentric navigational signals [20]. We may be able to model such a transformation using our approach.

Acknowledgments. This study was funded by NSF EFRI BRAID 2223793 to NSS.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Lynch, K., Park, F.C.: Modern Robotics: Mechanics, Planning, and Control. Cambridge University Press, Cambridge, UK (2017)
2. Burnod, Y., Grandguillaume, P., Otto, I., Ferraina, S., Johnson, P., Caminiti, R.: Visuomotor transformations underlying arm movements toward visual targets: a neural network model of cerebral cortical operations. J. Neurosci. **12**, 1435–1453 (1992). https://doi.org/10.1523/JNEUROSCI.12-04-01435.1992
3. Zubizarreta, A., Larrea, M., Irigoyen, E., Cabanes, I., Portillo, E.: Real time direct kinematic problem computation of the 3PRS robot using neural networks. Neurocomputing **271**, 104–114 (2018). https://doi.org/10.1016/j.neucom.2017.02.098
4. Smirnov, Y., Smirnov, D., Popov, A., Yakovenko, S.: Solving musculoskeletal biomechanics with machine learning. PeerJ Comput. Sci. **7**, e663 (2021). https://doi.org/10.7717/peerj-cs.663
5. Matheson, T.: Range fractionation in the locust metathoracic femoral chordotonal organ. J. Comp. Physiol. A. (1992). https://doi.org/10.1007/BF00191466
6. Mamiya, A., et al.: Biomechanical origins of proprioceptor feature selectivity and topographic maps in the Drosophila leg. Neuron (2023). https://doi.org/10.1016/j.neuron.2023.07.009
7. Delcomyn, F., Nelson, M.E., Cocatre-Zilgien, J.H.: Sense organs of insect legs and the selection of sensors for agile walking robots. Int. J. Robot. Res. **15**, 113–127 (1996). https://doi.org/10.1177/027836499601500201
8. Pouget, A., Dayan, P., Zemel, R.: Information processing with population codes. Nat. Rev. Neurosci. **1**, 125–132 (2000). https://doi.org/10.1038/35039062
9. Szczecinski, N.S., Hunt, A.J., Quinn, R.D.: A functional subnetwork approach to designing synthetic nervous systems that control legged robot locomotion. Front. Neurorobot. (2017). https://doi.org/10.3389/fnbot.2017.00037
10. Guie, C.K., Szczecinski, N.S.: Direct assembly and tuning of dynamical neural networks for kinematics. In: Conference on Biomimetic and Biohybrid Systems, pp. 321–331 (2022). https://doi.org/10.1007/978-3-031-20470-8_32
11. Nourse, W.R.P., Jackson, C., Szczecinski, N.S., Quinn, R.D.: SNS-toolbox: an open source tool for designing synthetic nervous systems and interfacing them with cyber-physical systems. Biomimetics. **8**, 247 (2023). https://doi.org/10.3390/biomimetics8020247
12. Westheimer, G., McKee, S.P.: Integration regions for visual hyperacuity. Vision. Res. **17**, 89–93 (1977). https://doi.org/10.1016/0042-6989(77)90206-1
13. Westheimer, G., McKee, S.P.: Spatial configurations for visual hyperacuity. Vision. Res. **17**, 941–947 (1977). https://doi.org/10.1016/0042-6989(77)90069-4
14. Siegler, M.V.S., Burrows, M.: The morphology of local non-spiking interneurones in the metathoracic ganglion of the locust. J. Comp. Neurol. **183**, 121–147 (1979). https://doi.org/10.1002/cne.901830110
15. Burrows, M., Newland, P.L.: Correlation between the receptive fields of locust interneurons, their dendritic morphology, and the central projections of mechanosensory neurons. J. Comp. Neurol. **329**, 412–426 (1993). https://doi.org/10.1002/cne.903290311
16. Bueschges, A., Kittmann, R., Schmitz, J.: Identified nonspiking interneurons in leg reflexes and during walking in the stick insect. J. Comp. Physiol. A. **174**, 685–700 (1994). https://doi.org/10.1007/BF00192718

17. Gebehart, C., Büschges, A.: Temporal differences between load and movement signal integration in the sensorimotor network of an insect leg. J. Neurophysiol. **126**, 1875–1890 (2021). https://doi.org/10.1152/jn.00399.2021
18. Szczecinski, N.S., Quinn, R.D.: Template for the neural control of directed stepping generalized to all legs of MantisBot. Bioinspir. Biomim. **12**, 045001 (2017). https://doi.org/10.1088/1748-3190/aa6dd9
19. Li, Y., Sukhnandan, R., Gill, J.P., Chiel, H.J., Webster-Wood, V., Quinn, R.D.: A bioinspired synthetic nervous system controller for pick-and-place manipulation. In: 2023 IEEE International Conference on Robotics and Automation (ICRA), pp. 8047–8053. IEEE, London, United Kingdom (2023). https://doi.org/10.1109/ICRA48891.2023.10161198
20. Lyu, C., Abbott, L.F., Maimon, G.: Building an allocentric travelling direction signal via vector computation. Nature **601**, 92–97 (2022). https://doi.org/10.1038/s41586-021-04067-0