Radboud University Nijmegen

Faculty of Social Sciences

Estimating and controlling dynamical systems through coordinated spikes

Spike Coding Networks for Kalman filtering and LQR control

Thesis MSc Artificial Intelligence

Author:
Filip S. Slijkhuis

Supervisor:
Pablo Lanillos

Supervisor:
Sander W. Keemink

2nd reader:
Bodo J. Rückauer

3-5-2022

Estimating and controlling dynamical systems through coordinated spikes

Filip S. Slijkhuis, Sander W. Keemink[?], and Pablo Lanillos[?]

Donders Institute for Brain, Cognition and Behavior
Department of AI, Radboud University, Nijmegen, the Netherlands
filipslijkhuis@outlook.com, {sander.keemink,pablo.lanillos}@donders.ru.nl

Abstract. Biological neural networks are able to control animal bodies robustly and efficiently, through highly sparse and irregular spiking patterns. In contrast, current implementations of control via artificial spiking neural networks often do not adhere to the spiking patterns found in the brain. In the neuroscience theory of Spike Coding Networks (SCNs), irregularity, sparsity, and robustness naturally arise as a consequence of coordinated spiking across a neural network, but it is not clear how to use SCNs for control. Here, we build upon the SCN framework and describe an analytical solution for estimation and control of dynamical systems with spiking networks, without the need for neural learning and optimization. The resulting networks work as spiking equivalents to a state estimator (Kalman filter) and an optimal controller (Linear–quadratic regulator). Results on numerical experiments, using the spring-mass-damper and cartpole systems, show that the networks retain their irregular and sparse spiking patterns, whilst being robust against neural death. The framework offers a powerful perspective on how precise control can be achieved through biological and artificial neural networks, and opens the way for deploying fast and efficient neuromorphic controllers.

Keywords: Spiking Neural Networks · State Estimation · Optimal Control · Dynamical Systems.

1 Introduction

One of the main functions of the brain is to control the body, and it does so very effectively. Remarkably, the brain achieves this through sparse and irregular spiking behavior of networks of neurons [1, 14] making them highly energy-efficient. Additionally, the brain is robust against perturbations, such as neuron death [11, 18, 20]. Inspired by these spiking patterns, together with recent advances in surrogate gradient learning [22], artificial Spiking Neural Networks (SNNs) have started being investigated to achieve low-energy control in both machine learning and neuromorphics [10]. However, the resulting architectures often produce highly dense and regular spiking activity | see e.g. the activity patterns in [12, 23]; and although some papers did investigate enforcing more efficient spiking [17,

* Equal contribution

24], proper irregular spiking patterns are still not achieved. Finally and crucially, the majority of existing methods rely on learning and optimizing network parameters, while for known systems an analytical solution can be more desirable. *Can we find a way to control known dynamical systems using SNNs, without a need for training, while also reproducing the sparse and irregular spiking patterns found in the brain?*

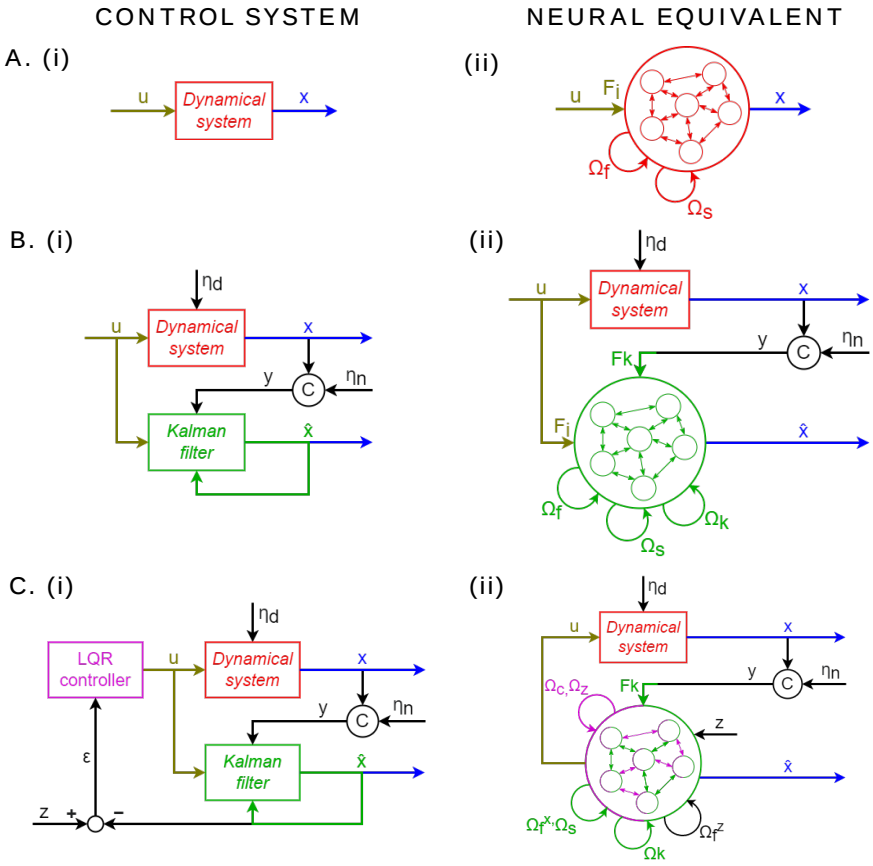

Fig. 1: Schematics for representation, estimation and control of dynamical systems. (left column) Control system description and (right column) neural equivalent. Recurrent connections in the neural representations are illustrative | see Sec 3. **(A)** Dynamical system representation. **(B)** Dynamical system estimation. **(C)** System estimation and control.

Spike Coding Networks (SCNs) [4] appeared as an alternative method to solve optimization problems without the need to learn and optimize the parameters. These networks follow a similar principle to the theory of predictive coding, as neurons only fire when the network's prediction error exceeds a threshold

value. As a result, the neural spiking activity is `coordinated' across the neuron population, producing sparse and irregular patterns similar to those observed in the brain [4, 8], and high robustness against biological perturbations such as neuron death [3, 8]. Importantly, all this is achieved through a closed-form solution for the recurrent connectivity. This includes being able to implement any dynamical system in an SCN analytically, [4, 21], or through learning [2, 5, 25] (Fig. 1A). This has an enormous potential in engineering, as in principle we could perform system identification, and implement the identified system in an SCN. However, it remains an open question how well-suited SCNs are for actual control of those systems, for which both optimal state estimation and control are necessary.

In this work, we mathematically formalize how to extend the spike-coding framework for estimation and control of dynamical systems by implementing both a Kalman filter (for estimation) and a Linear{quadratic regulator (LQR) conroller in the spiking dynamics (Fig. 1B and C). We evaluated the proposed SCN model on the partially observable Spring-Mass-Damper (SMD) and non-linear cartpole systems. We show that the networks can effectively estimate and control these systems, whilst preserving the irregular and sparse spiking patterns and robustness to neuron death. Our main contribution is a unified and clear mathematical framework for analytically deriving SNNs for state-estimation and optimal control.

1.1 Previous work on SCN and control

Our work builds on previous research on SCNs for dynamical system representation. In [4], the authors originally showed how any linear dynamical system can be analytically implemented in an SCN. While there have been several follow-up papers on extending the framework for general computations [2, 19, 21], control has been less well-studied. To the best of the authors' knowledge there are two exceptions.

First, in Thalmeier et al. [25], spike coding networks are extended by adding learning rules for connectivity weights. They use the resulting networks for non-linear control through forward modelling. In order to control a pendulum, the network first learns an internal world model of the dynamical system. Control is then achieved by simulating multiple future trajectories of the pendulum with different control policies, but the control is then not performed within a single network, and they do not consider explicit state estimation (like Kalman filtering).

Second, in a series of papers [15, 16], Huang et al. derive their own framework for controlling linear and nonlinear dynamical systems with a new loss function focused on control. While similar to the SCN framework, they also do not consider state estimation and do not investigate the robustness of the implementation in detail.

Overall, our approach takes inspiration from the above papers, but we consider robustness to cell death more extensively, and integrate both state estimation and control within the SCN framework.

2 Spike coding network foundations

Here, we will brie y summarize the SCN framework (as originally proposed in [4]), and how it can be used to implement any linear dynamical system in an SNN. For more extensive derivations we refer the reader to the method sections of [3, 21]. Consider an SNN consisting of N neurons. Each neuron emits a spike train of the form $s_i(t) = \sum_k (t - t_k^i)$, where t_k^i is the timing of the k^{th} spike of neuron i and (x) is the Dirac delta function. The vector $\mathbf{s}(t) = [s_1(t); :::; s_N(t)]^>$ contains the spike trains of the entire population of neurons. For ease of reading we will usually leave out the time dependency (t) moving forward.

Representation Before implementing a dynamical system, let us rst consider how to represent an external K-dimensional signal $\mathbf{x}(t) \in \mathbb{R}^K$, by using spikes to constrain the coding error $\|\mathbf{x} - \mathbf{D}\mathbf{r}\|_2^2$. The SCN framework achieves this by relying on two core assumptions [4]: (1) The input signal estimate can be linearly decoded from the spiking activity ($\hat{\mathbf{x}} = \mathbf{D}\mathbf{r}$), where $\mathbf{r}(t) = [r_1(t); :::; r_N(t)]^>$ is the vector of ltered spike trains of the neuron population (according to $\dot{\mathbf{r}} = -\mathbf{r} + \mathbf{s}$), and $\mathbf{D} \in \mathbb{R}^{K \times N}$ are random decoding weights (see Sec. 4.3). (2) Spikes should only be emitted when this improves the coding error, yielding a greedy spiking rule, i.e. neuron i should spike if $\|\mathbf{x} - \mathbf{D}\mathbf{r}\|_2^2 > \|\mathbf{x} - \mathbf{D}\mathbf{r} - \mathbf{D}_i\|_2^2$. Here $\mathbf{D}_i$ is the i'th column of $\mathbf{D}$, and shows the change in the error if neuron i would spike. These two assumptions can be fully implemented in an SNN in which the neural voltages track the projected coding error for neuron i

$$V_i = \mathbf{D}_i^>(\mathbf{x} - \hat{\mathbf{x}}), \tag{1}$$

and in which neuron i should spike when a speci c threshold is crossed ($T_i = \frac{\mathbf{D}_i^\top \mathbf{D}_i}{2}$).

A recurrent network of leaky-integrate-and- re neurons with the following dynamics will track the voltage in Eq. 1 **exactly**:

$$\dot{\mathbf{v}} = -\mathbf{v} + \mathbf{D}^>(\dot{\mathbf{x}} + \mathbf{x}) - \mathbf{D}^>\mathbf{D}\mathbf{s}, \tag{2}$$

where $\mathbf{v} = [V_1 ::: V_N]^>$ is the population vector of voltages. In this network, the input $\mathbf{x}$ is encoded through forward weights $\mathbf{D}^>$, with the derivative term $\dot{\mathbf{x}}$ ensuring that quick changes in $\mathbf{x}$ are adequately tracked. E ectively, the network takes in both the current state and its dynamics as inputs. The recurrent connections are given by $\Omega_f = -\mathbf{D}^>\mathbf{D}$. These connections, which we will refer to as the \fast" recurrent connections, make sure that the spiking in the network is coordinated across the neurons, such that there are no super uous spikes. The neural `self-reset' is implicitly included by Ω_f's diagonal.

Implementing a dynamical system The framework can be extended such that the neural dynamics implement a linear dynamical system (Fig. 1A) [4]. Let us now consider a linear dynamical system of the form $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$, where $\mathbf{u}$ is

some external input which is transformed by an input matrix B. Above we have an SCN with an internal signal tracking an input such that $\hat{x} \approx x$, instead, we now want a purely internal estimate which follows the dynamics $\dot{\hat{x}} = A\hat{x} + Bu$. We can replace $\underline{x}$ (in Eq. 2) by Ax, and then the external state x by the internal estimate $\hat{x}(= Dr)$. On top of that, we encode Bu into the network, and in order to investigate robustness against neural voltage, add voltage noise η_V, after which we arrive at

$$\underline{v} = v + D^{>}(A\hat{x} + \dot{\hat{x}}) - D^{>}Ds + D^{>}Bu + \eta_V \quad (3)$$

$$\underline{v} = v + \Omega_s r + \Omega_f s + F_i u + \eta_V, \quad (4)$$

where we have introduced "fast" connectivity, $\Omega_f = D^{>}D$ (since it acts through instantaneous spikes), "slow" connectivity, $\Omega_s = D^{>}(A + I)D$ (since it acts through the filtered spike trains r), and "input" connectivity, $F_i = D^{>}B$. The network now keeps track of its own estimate on a fast time scale through the fast connections, and drifts this estimate according to the dynamics (A) and the input (Bu), through the slow recurrent connections and feed-forward input connections. In essence, the slow connectivity takes the internal rates of the network and decodes them using D. It then applies the dynamics of the linear dynamical system through A, and encodes the result back into the network using $D^{>}$.

3 SCNs for prediction and control

In the previous section, we showed how SCNs can implement dynamical systems, but how well could they work for control? Here we describe how an SCN can function simultaneously as a state-estimator (Kalman filter) and a controller (LQR) for an external dynamical system.

3.1 Control problem definition

Consider a linear dynamical system in state-space representation (Fig. 1A.i):

$$\underline{x}(t) = Ax(t) + Bu(t) + \eta_d \quad (5)$$

$$y(t) = Cx(t) + \eta_n, \quad (6)$$

where $x(t) \in \mathbb{R}^K$ is the state vector, $A \in \mathbb{R}^{K \times K}$ is the system matrix conveying the dynamics of the system, $u(t) \in \mathbb{R}^P$ is the input vector, and $B \in \mathbb{R}^{K \times P}$ is the input matrix. η_d reflects internal disturbance, given by a zero-mean Gaussian process with co-variance Σ_d. $y(t) \in \mathbb{R}^Q$ are observations about $x(t)$, where $C \in \mathbb{R}^{Q \times K}$ is an observability matrix. η_n is sensor noise, given by a zero-mean Gaussian process with co-variance Σ_n.

Our goal is to control x to some target state $z \in \mathbb{R}^K$. To do this we must be able to estimate the state x from the observations y (*estimation problem*), and find the best control signal u to do so (*optimal control problem*). For the estimation problem, we will assume as known the output vector y, the system

matrix A, input matrix B, the input vector u, the observability matrix C, and co-variances $_d$ and $_n$. For the control problem we assume known y, A, B, C, $_d$, and $_n$. Given all of the above, how can we solve both the estimation and the control problem with an SCN?

3.2 SCN as a state estimator

Our goal is to gain a full-state estimate $\hat{x}$ given a noisy and incomplete measurement. The Kalman lter provides such full state estimates [6], and it optimally balances an internal (predicted) state estimate to a noisy and/or partially observable external measurement (Fig. 1B.i). The Kalman lter can be represented through a dynamical system of the form

$$\dot{\hat{x}} = A\hat{x} + Bu + K_f(\hat{y} \quad y). \tag{7}$$

The Kalman lter gain matrix, K_f, is applied to an error between the observations of the dynamical system and the Kalman lter's own internal estimate, $\hat{y} = C\hat{x}$. K_f can be found by solving an Algebraic Riccati equation [6]. Let us now assume that the Kalman lter gain matrix, K_f, is known beforehand. Since the Kalman lter in this form is a linear dynamical system, we can directly represent it in an SCN from Section 2. This results in the voltage update rule

$$\underline{v} = \quad v + D^{>}(A\hat{x} + \quad \hat{x}) \quad D^{>}Ds + D^{>}Bu + D^{>}K_f(\hat{y} \quad y) + \quad _V \tag{8}$$

$$\underline{v} = \quad v + \quad _s r + \quad _f s + F_i u + \quad _k r + F_k y + \quad _V, \tag{9}$$

where, on top of the previously introduced slow, fast, and input connectivity, we now also have recurrent \Kalman lter" connections, $_k = D^{>}K_f CD$, and feed-forward \Kalman lter" connections, $F_k = D^{>}K_f$. The former takes the internal state of the system and decodes it, applies both the observability matrix and the Kalman lter gain matrix, and encodes it back into the network. The latter takes in the external partially observable state, y, and applies the Kalman lter gain matrix. Now, the entire SCN, including all of its connections, represent a Kalman lter, as can be seen in Fig. 1B.ii. The network internally simulates a linear dynamical system of the form $\dot{\hat{x}} = A\hat{x} + Bu$, but also corrects its own estimate according to the external input y, like a regular Kalman lter.

3.3 SCN as a controller

Now that we can use SCNs for state-estimation, how can we extend it for control? We will consider the Linear-Quadratic Regulator (LQR) controller. Given a known system of the form in Eq. (5), this controller produces the optimal u signal to minimize the squared error between the state x and a target z. This gives rise to a linear control law of the form

$$u = \quad K_c(x \quad z). \tag{10}$$

K_c can again be found by solving an algebraic Riccati equation, with parameters Q and R, which weigh the cost of deviations of the state from zero and the cost

of actuation, respectively [6]. In order to control a partially observable and/or noisy dynamical system, we can combine an LQR controller with a Kalman filter, since it outputs a full state estimate (see Fig. 1C.i).

In order to perform both the estimation and LQR control within the same SCN, we can now extend the dynamical system which the SCN represents. Assuming we know our gain matrix K_c, we can swap out $\mathbf{u}$ for the LQR control law (Eq. (10) in Eq. (7)), which gives us a dynamical system of the form $\dot{\hat{\mathbf{x}}} = A\hat{\mathbf{x}} + BK_c(\hat{\mathbf{x}} - \mathbf{z}) + K_f(\hat{\mathbf{y}} - \mathbf{y})$. We encode $\mathbf{z}$ into the network with a new set of fast connections (similar to Eq. (2)), which allows us to read out $\mathbf{u}$ from the internal estimates $\hat{\mathbf{x}}$ and $\hat{\mathbf{z}}$ of the network. This causes us to have two sets of decoding weights: $D_{\mathbf{x}}$ for $\hat{\mathbf{x}}$, and $D_{\mathbf{z}}$ for $\hat{\mathbf{z}}$. Extending the SCN defined in Eq. (8) with LQR control and an internal encoding of $\mathbf{z}$, we get the new voltage update rule

$$\begin{aligned}\underline{\mathbf{v}} = \; & \mathbf{v} + {}_{s}\mathbf{r} - D_{\mathbf{x}}^{>} D_{\mathbf{x}}\mathbf{s} + D_{\mathbf{x}}^{>} BK_c(\hat{\mathbf{x}} - \hat{\mathbf{z}}) + D_{\mathbf{x}}^{>} K_f(\hat{\mathbf{y}} - \mathbf{y}) \\ & + D_{\mathbf{z}}^{>}(\underline{\mathbf{z}} + \mathbf{z}) - D_{\mathbf{z}}^{>} D_{\mathbf{z}}\mathbf{s} + {}_{V}\end{aligned} \tag{11}$$

$$\begin{aligned}\underline{\mathbf{v}} = \; & \mathbf{v} + \underbrace{{}_{s}\mathbf{r} + {}_{f}^{\mathbf{x}}\mathbf{s}}_{\text{System estimate}} + \underbrace{{}_{c}\mathbf{r} + {}_{z}\mathbf{r}}_{\text{Control estimate}} + \underbrace{{}_{k}\mathbf{r} + F_k\mathbf{y}}_{\text{Kalman update}} \\ & + D_{\mathbf{z}}^{>}(\underline{\mathbf{z}} + \mathbf{z}) + \underbrace{{}_{f}^{\mathbf{z}}\mathbf{s}}_{\text{Encoding of } \mathbf{z}} + {}_{V},\end{aligned} \tag{12}$$

where we have indicated the parts of the connectivity that are tracking the external system, the effect of the control on the system (of which $\mathbf{z}$ is internally encoded), and the Kalman updates. The system tracks both $\hat{\mathbf{x}}$ and $\hat{\mathbf{z}}$, which are separately encoded using their own sets of decoding weights. Consequently, both variables can also be read out from the state of the SCN using the corresponding weights. For the internal estimate $\hat{\mathbf{x}}$, we now have two sets of recurrent "control" and "target" connections, which represent the LQR controller within the Kalman filter SCN (see Fig. 1C.ii). Here, the recurrent control connectivity, ${}_{c} = -D_{\mathbf{x}}^{>} BK_c D_{\mathbf{x}}$, decodes $\hat{\mathbf{x}}$ from the internal state of the SCN using $D_{\mathbf{x}}$, and applies the LQR gain matrix K_c. The result is transformed using B, and encoded back into the network. The recurrent target connectivity, ${}_{z} = D_{\mathbf{x}}^{>} BK_c D_{\mathbf{z}}$, does something similar, but with $\hat{\mathbf{z}}$, which is decoded from the internal state of the SCN using $D_{\mathbf{z}}$. Because of the polarity of their respective connections, $\hat{\mathbf{x}}$ and $\hat{\mathbf{z}}$ are subtracted from each other. The reference signal, $\mathbf{z}$, is encoded into the network using the same method as defined in Eq. (8). We now have an SCN which represents the entirety of $\dot{\hat{\mathbf{x}}} = A\hat{\mathbf{x}} + BK_c(\hat{\mathbf{x}} - \hat{\mathbf{z}}) + K_f(\hat{\mathbf{y}} - \mathbf{y})$, but also internally keeps an estimate of $\mathbf{z}$. We can obtain the output of the internal LQR controller in the SCN by defining the following new set of decoding weights: $D_{\mathbf{u}} = -K_c(D_{\mathbf{x}} - D_{\mathbf{z}})$. We then apply $D_{\mathbf{u}}$ to the internal state of the SCN: $\mathbf{u} = D_{\mathbf{u}}\mathbf{r}$. This $\mathbf{u}$ can be applied to an external dynamical system to control it, all whilst the SCN keeps an internal estimate of the external dynamical system.

4 Results

Here, we examine the performance of the proposed mathematical framework (Sec. 3) through numerical experiments[1]. We evaluated the networks produced by the spike-coding framework for estimation, which we will refer to as \Kalman lter SCNs", and of networks produced by the framework for estimation and control, \Control SCNs", on two dynamical systems, the linear Spring-Mass-Damper (SMD) system and the nonlinear cartpole system. Furthermore, we studied the robustness of the Control SCNs to neuron death.

4.1 Spring-Mass-Damper system

Throughout this section, we shall look at the performance of both the Kalman lter SCN and the Control SCN in relation to the spring-mass-damper system. The SMD system has two dynamical variables: position x (= x_1) and velocity v (= $\underline{x}_1$). Fig. 2A shows an illustration of the spring-mass-damper system. Our instance of the SMD system has an internal disturbance $_d$, and it outputs a partially observable state with only the position x with sensor noise $_n$.

We rst studied the performance of the Kalman lter SCN on the previously described SMD system. For the estimation, we set $u = 0$, so there is no external force acting on our dynamical system. In Fig. 2B, we compared the estimation outputted by the spike coding network (green) to the real dynamical system (red). Even though the estimation starts in a di erent initial state and there is both substantial measurement noise and internal disturbance in the real dynamical system, it quickly converges. Overall, the SCN estimates both dynamical variables of the real dynamical system with very high accuracy.

Next, we studied the performance of the Control SCN on the SMD system. The SCN produces a control signal, u, such that the state of the real dynamical system converges towards the reference signal z. In Fig. 2C, we show the real dynamical system (red) and the estimation of the spike coding network (green). Once again, the estimation from the SCN is very similar to the real dynamical system. On top of that, we show the reference signal (purple, dashed), which is a stair-wise increase of the rst dynamical variable, x. We can see that both the estimation from the SCN and the real dynamical system follow the reference signal almost perfectly, which is an indication of the correct behavior of the internal controller of the SCN.

In previous work, SCNs have shown to be very tolerant against neuron death [8], which is retained in the SCNs generated through our proposed frameworks. Fig. 2D shows the performance of a Control during the removal of neurons at certain timesteps (red, vertical dashed lines). Once again, the SCN very accurately controls the dynamical system. The performance of the SCN stays accurate up until about 8 neurons, where the amount of neurons is not su cient enough for

[1] The code used to generate the results is available in the following Github repository: https://github.com/FSSlijkhuis/SCN_estimation_and_control

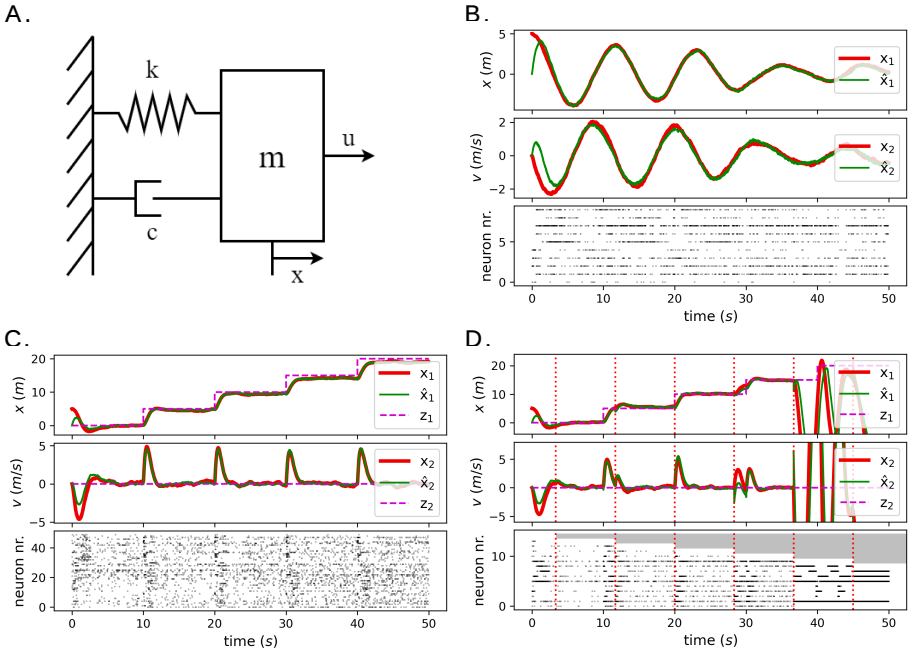

Fig. 2: Estimation and control of a linear system. (A) Spring-Mass-Damper system. (B) SCN estimation *(green)* of SMD system *(red)*. (C) Control of SMD system *(red)* using SCN controller and estimator *(green)*, with reference signal *(purple, dashed)*. (D) Robustness to neuron death of SCN controller and estimator *(green)* controlling SMD system *(red)*. At every red, dashed vertical line, a neuron is \killed" (indicated by the gray bars in the spike plot).

an accurate enough estimation, which causes an inaccurate control signal to be provided by the SCN.

In all of the spike plots in Fig. 2, we can see only moderately sparse and irregular spiking patterns, which is caused by the small amount of neurons used in the networks, which has been done for illustrative purposes. The spiking patterns, however, can be made arbitrarily sparse by adding more neurons (as then the problem is coordinated across more neurons). In Fig.2D, we can observe that when a neuron is killed o , other neurons compensate for its spiking, to make sure that the estimation stays as accurate as possible. This is a clear indication of the coordinated spiking of the neurons in the network.

4.2 Cartpole system

Here, we evaluate the Control SCN on the nonlinear cartpole system (see Fig. 3A). The cartpole has four dynamical variables: the position of the cart, x ($= x_1$), the velocity of the cart v ($= x_2 = \underline{x}_1$), the position of the pole, ($= x_3$), and the angular velocity of the pole, ! ($= x_4 = \underline{x}_3$). Just like the SMD, our instance of

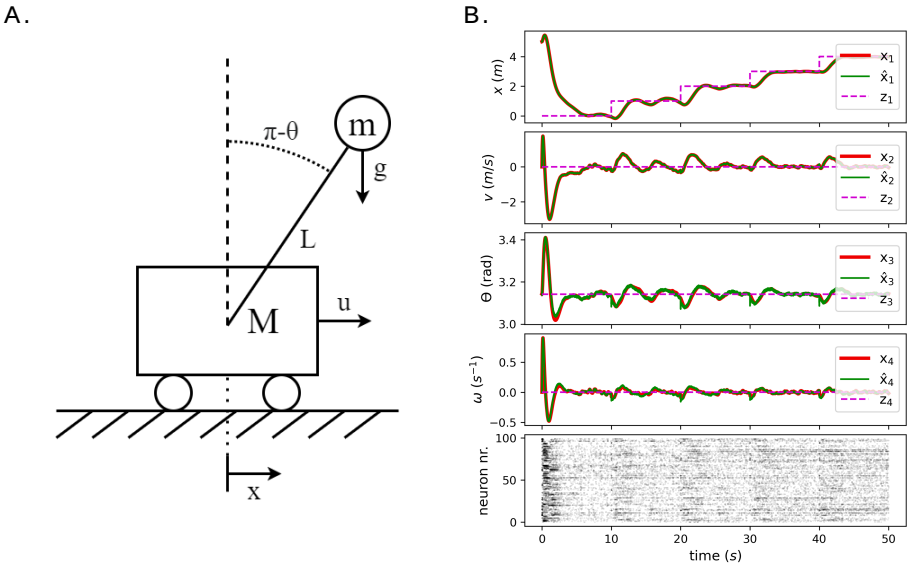

Fig. 3: Control of nonlinear Cartpole system. (A) Cartpole system. (B) Control of Cartpole system *(red)* using SCN controller and estimator *(green)* linearized around up-position of pole, with reference signal *(purple, dashed)*.

the cartpole has an internal disturbance $_d$, and it outputs a partially observable state consisting of only the position of the cart, x, with sensor noise $_n$. Please note that the cartpole system is a nonlinear dynamical system, but our Control SCN assumes linear dynamics. In order for the SCNs to produce a meaningful control signal, we use linearized dynamics for the internal estimate, which have been linearized around the point of control. In our case, we linearized around the up-position of the pole (=).

Fig. 3B compares the estimation from a Control SCN (green) to the real dynamical system (red). On top of that, we show the reference signal (purple, dashed), which is a stair-wise increase of the rst dynamical variable, x. We see that the dynamical system follows the reference signal almost perfectly, and most importantly, the pole remains in the up-position.

The spike plots show that especially with a large amount of neurons, the spiking is highly irregular and sparse, which is in line with SCN frameworks from previous work. Increased spiking activity is observed when the reference signal demands a change of the state of the dynamical system.

4.3 Parameter settings

For each network, the decoding weights (D) are sampled from a normal distribution, with each column normalized to have norm 0.1, except for norm 0.01 in Fig. 3B. 10, 50 and 14 neurons were used in Fig. 2B, C, and D respectively, and

100 neurons in Fig. 3B. The SMD parameters were $m = 20$, $k = 6$ and $c = 2$, with $_d = _n = 10$. The cartpole system parameters were $m = 1$, $M = 5$, $L = 2$, $g = 10$, and $d = 1$, with $_d = _n =$ 1e-4. All networks used $_V =$ 1e-5 and $= 0{:}1$. Forward Euler was used throughout, with a timestep of 1e-3s, except for 1e-4s in Fig. 3B. For control, $R =$ 1e-2. For control of the SMD system, the matrix $Q \; 2 \; R^{K \; K}$ prioritizes x_1 with weight 10, and x_2 with weight 1. For cartpole control, x_3 is weighted with 10, the rest with 1.

5 Conclusion

We extended the spike-coding framework for estimation and control. The resulting spike coding networks work as spiking equivalents to a Kalman lter and an LQR controller. We showed that the sparse and irregular spiking patterns found in the original framework, and the robustness against neural death, are retained. Given these properties, our proposed mathematical framework provides a new perspective for precise control using spiking neural networks. Importantly, since these SCNs are analytically derived, this opens up the prospect for deploying fast and e cient neuromorphic controllers. However, some challenges have to be addressed beforehand, such as modelling synaptic delays (which are assumed to be instantaneous in the SCN framework, although solutions exist [7, 9]). Future work could involve the learning of LQR and Kalman Filter gains (similarly to [13]), combining our analytical implementation with internal learning, as well as the implementation of nonlinear dynamical systems inside the network (similar to [21]).

6 Acknowledgements

We thank Marcel van Gerven, Bodo Rueckauer, Justus Hubotter, Christian Machens, William Podlaski, and Michele Nardin for helpful discussions on control and networks.

References

1. L. F. Abbott, Brian DePasquale, and Raoul-Martin Memmesheimer. Building functional networks of spiking model neurons. *Nature Neuroscience*, 19(3):350{355, March 2016. Number: 3 Publisher: Nature Publishing Group.
2. Alireza Alemi, Christian Machens, Sophie Deneve, and Jean-Jacques Slotine. Learning Nonlinear Dynamics in E cient, Balanced Spiking Networks Using Local Plasticity Rules. *Proceedings of the AAAI Conference on Arti cial Intelligence*, 32(1), April 2018.
3. G. T. Barrett David, Deneve Sophie, and Christian K. Machens. Optimal compensation for neuron loss. *eLife*, 5, 2016.
4. Martin Boerlin, Christian K. Machens, and Sophie Deneve. Predictive Coding of Dynamical Variables in Balanced Spiking Networks. *PLOS Computational Biology*, 9(11):1{16, November 2013. Publisher: Public Library of Science.

5. Wieland Brendel, Ralph Bourdoukan, Pietro Vertechi, Christian K. Machens, and Sophie Deneve. Learning to represent signals spike by spike. *PLOS Computational Biology*, 16(3):e1007692, 2020. Publisher: Public Library of Science.
6. Steven L. Brunton and J. Nathan Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, USA, 1st edition, 2019.
7. Camille E. Rullan Buxo and Jonathan W. Pillow. Poisson balanced spiking networks. *PLOS Computational Biology*, 16(11):e1008261, November 2020. Publisher: Public Library of Science.
8. Nuno Calaim, Florian Alexander Dehmelt, Pedro J. Goncalves, and Christian K. Machens. Robust coding with spiking networks: a geometric perspective. *bioRxiv*, 2020. Publisher: Cold Spring Harbor Laboratory _eprint: https://www.biorxiv.org/content/early/2020/06/15/2020.06.15.148338.full.pdf.
9. Matthew Chalk, Boris Gutkin, and Sophie Deneve. Neural oscillations as a signature of e cient coding in the presence of synaptic delays. *eLife*, 5:e13824, July 2016. Publisher: eLife Sciences Publications, Ltd.
10. Mike Davies, Andreas Wild, Garrick Orchard, Yulia Sandamirskaya, Gabriel A. Fonseca Guerra, Prasad Joshi, Philipp Plank, and Sumedh R. Risbud. Advancing Neuromorphic Computing With Loihi: A Survey of Results and Outlook. *Proceedings of the IEEE*, 109(5):911{934, May 2021. Conference Name: Proceedings of the IEEE.
11. Sophie Deneve, Alireza Alemi, and Ralph Bourdoukan. The Brain as an E cient and Robust Adaptive Learner. *Neuron*, 94(5):969{977, June 2017.
12. Chris Eliasmith, Terrence C. Stewart, Xuan Choo, Trevor Bekolay, Travis DeWolf, Yichuan Tang, and Daniel Rasmussen. A Large-Scale Model of the Functioning Brain. *Science*, 338(6111):1202{1205, November 2012. Publisher: American Association for the Advancement of Science.
13. Johannes Friedrich, Siavash Golkar, Shiva Farashahi, Alexander Genkin, Anirvan Sengupta, and Dmitri Chklovskii. Neural optimal feedback control with local learning rules. In *Advances in Neural Information Processing Systems*, volume 34, pages 16358{16370. Curran Associates, Inc., 2021.
14. Wulfram Gerstner, Werner M. Kistler, Richard Naud, and Liam Paninski. *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press, July 2014. Google-Books-ID: D4j2AwAAQBAJ.
15. Fuqiang Huang and ShiNung Ching. Dynamical Spiking Networks for Distributed Control of Nonlinear Systems. In *2018 Annual American Control Conference (ACC)*, pages 1190{1195, June 2018. ISSN: 2378-5861.
16. Fuqiang Huang, James Riehl, and ShiNung Ching. Optimizing the dynamics of spiking networks for decoding and control. In *2017 American Control Conference (ACC)*, pages 2792{2798, May 2017. ISSN: 2378-5861.
17. Justus F. Hubotter, Pablo Lanillos, and Jakub M. Tomczak. Training Deep Spiking Auto-encoders without Bursting or Dying Neurons through Regularization. *arXiv:2109.11045 [cs]*, September 2021. arXiv: 2109.11045.
18. Megan C. Leary and Je rey L. Saver. Annual Incidence of First Silent Stroke in the United States: A Preliminary Estimate. *Cerebrovascular Diseases*, 16(3):280{285, 2003. Publisher: Karger Publishers.
19. Allan Mancoo, Sander Keemink, and Christian K Machens. Understanding spiking networks through convex optimization. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 8824{8835. Curran Associates, Inc., 2020.

20. John H. Morrison and Patrick R. Hof. Life and Death of Neurons in the Aging Brain. *Science*, 278(5337):412{419, October 1997. Publisher: American Association for the Advancement of Science.
21. Michele Nardin, James W. Phillips, William F. Podlaski, and Sander W. Keemink. Nonlinear computations in spiking neural networks through multiplicative synapses. *Peer Community In Circuit Neuroscience*, November 2021. Publisher: Peer Community In.
22. Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-Based Optimization to Spiking Neural Networks. *IEEE Signal Processing Magazine*, 36(6):51{63, November 2019. Conference Name: IEEE Signal Processing Magazine.
23. Darjan Salaj, Anand Subramoney, Ceca Kraisnikovic, Guillaume Bellec, Robert Legenstein, and Wolfgang Maass. Spike frequency adaptation supports network computations on temporally dispersed information. *eLife*, 10:e65459, July 2021. Publisher: eLife Sciences Publications, Ltd.
24. Martino Sorbaro, Qian Liu, Massimo Bortone, and Sadique Sheik. Optimizing the Energy Consumption of Spiking Neural Networks for Neuromorphic Applications. *Frontiers in Neuroscience*, 14, 2020.
25. Dominik Thalmeier, Marvin Uhlmann, Hilbert J. Kappen, and Raoul-Martin Memmesheimer. Learning Universal Computations with Spikes. *PLOS Computational Biology*, 12(6):1{29, June 2016. Publisher: Public Library of Science.

A Open-loop control with an SCN

In Sec. 3.3, we provided a method for extending Kalman- lter SCNs with control. However, we do not necessarily need state estimation for control. In this section, we show that control can also internally be applied to SCNs simulating dynamical systems (Eq. (3)). In order to show the precision of the internal estimation of a dynamical system with control, we extract the generated control policy, and apply this to an external dynamical system, yielding a form of open-loop control (see also Fig. 4A).

Let us consider an SCN simulating a dynamical system, as de ned by Eq. (3). The dynamical system has a form of $\underline{x} = Ax + Bu$. In Sec. 3.3, we have seen that LQR is implemented using a linear control law of the form $u = K_c(x \quad z)$. Therefore, to enable internal control of the dynamical system, we need the SCN to represent a dynamical system of the form $\underline{\hat{x}} = A\hat{x} + BK_c(\hat{x} \quad z)$. Just like in Sec. 3.3, we use an internal representation of the reference signal z. Extending the SCN de ned in Eq. (3) with LQR control and internal estimation of z, we get the voltage update rule

$$\begin{aligned} \underline{v} = \quad & v + \quad _s r \quad D_x^{>} D_x s + D_x^{>} B K_c(\hat{x} \quad \hat{z}) \\ & + D_z^{>}(\underline{z} + \quad z) \quad D_z^{>} D_z s + \quad _V \end{aligned} \tag{13}$$

$$\begin{aligned} \underline{v} = \quad & v + \quad _s r + \quad _f^x s + \quad _c r + \quad _z r \\ & + D_z^{>}(\underline{z} + \quad z) + \quad _f^z s + \quad _V , \end{aligned} \tag{14}$$

where, for the estimation of $\hat{x}$, we have added the recurrent control connectivity, $\quad _c = \quad D_x^{>} B K_c D_x$, which decodes the internal state of the SCN using