Spiking neural networks application to signal processing: observation of dynamical systems

Anna Bulanova[1], Olivier Temam[1] and Rodolphe Heliot[2]

***Abstract*— We investigated the possibility of using spiking neural networks for signal processing, specifically for observation of dynamical systems. For this task we applied Deneve's balanced spiking network framework. Simulations in Brian/Python were performed, in which the network was initialized using known information about the dynamical system, and taking in account spiking neural hardware limitations. In our tests, mean normalized root mean square error of the solution was under 2%, which makes such spiking neural networks very suitable for observation tasks on neuromorphic hardware.**

I. Introduction

Spiking neural networks (SNN) have been widely investigated in recent years [1]. Networks of spiking neurons were proven to have at least the same or greater computational power than networks of sigmoidal and McCulloch–Pitts neurons, while computation of certain functions with SNN requires significantly fewer neurons [2]. Many applications of SNNs are in emulating and studying biological neural networks, as they provide a more realistic representation of biological networks than classical artificial neural networks (see for example [3], [4], [5], [6]). Also, much research has been done in applying SNN to temporal pattern recognition [7], [8]. There are some projects investigating SNN applications in robotics, most of which rely on evolutionary algorithms to train the network [9], [10].

We address the problem of using SNNs in signal processing, particularly to the observation of dynamical systems. Indeed, the observation problem is a classic one in signal processing, and has many practical applications. Once the observer has been built (see details in section II-A), the problem comes down to the simulation of a dynamical system. To achieve such a simulation using SNNs, balanced spiking networks from [11] were adopted. In this approach, recurrent spiking network of leaky integrate and fire neurons tracks solution of a linear dynamical system by minimizing prediction error. The connection, encoding and decoding weights are set based on the knowledge about the dynamical system, so no training of the network is necessary.

We considered a "real life" problem of tracking heading of a ship using control input and compass measurement, and ran simulations of networks implementing dynamical system representing the ship and its observer.

The paper is organized as follows: observation problem and neural network design are introduced in section II, section III contains the details of software simulations and their results, section IV concludes this paper.

II. Methods

A. Observation problem

We consider a control theory framework in this section; dynamical systems are defined by a state equation and an output equation. In many systems, the state vector is not fully available for measurement at the output: certain elements of the system's state may be internal variables, or be too costly to measure physically. One possible solution to this problem is the use of a state observer, which allows estimation of the entire state of an observable system using measurements of its input and output [12]. State observer is itself a dynamical system,

[1]Anna Bulanova and Olivier Temam are with the By-Moore team, INRIA, Campus de l'Ecole Polytechnique, 91120 Palaiseau, France anna.bulanova@inria.fr, olivier.temam@inria.fr

[2]Rodolphe Heliot is with CEA – LETI, Minatec Campus, 38054 Grenoble Cedex, France rodolphe.heliot@gmail.com

whose inputs are inputs and outputs of the original system, and its state vector tracks the state of the original system (see Fig. 1). Since it is simulated, its entire state vector is available, thus giving access to previously unknown state variables.

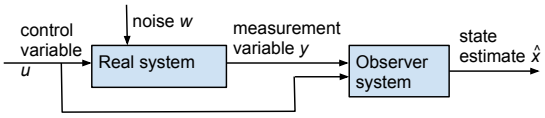

Fig. 1. Diagram of a state observer

B. Example system: inertial navigation

We considered the following model of the ship motion (1st-order Nomoto model) [13]:

$$\begin{aligned} \dot{\psi} &= r, \\ \dot{r} &= -\frac{1}{T} - \frac{K}{T}(\delta - b) + w_r, \\ \dot{b} &= -\frac{1}{T_b} b + w_b. \end{aligned} \tag{1}$$

Above, ψ is heading angle, r is yaw rate, b is rudder offset, δ is the control input. Wave response is modeled by a linear wave model:

$$\begin{aligned} \dot{\xi}_w &= \psi_w, \\ \dot{\psi}_w &= -\omega_0^2 \xi_w - 2\lambda\omega_0\psi_w + K_w w_w. \end{aligned} \tag{2}$$

The noise terms, w_r, w_b, and w_w are modeled as white noise processes. The output is a compass measurement:

$$y = \psi + \psi_w. \tag{3}$$

The system given by the equations (1) and (2) can be presented in matrix form:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}u + \mathbf{E}\mathbf{w}, \\ y &= \mathbf{C}\mathbf{x}, \end{aligned} \tag{4}$$

where

$$\mathbf{x} = \left[\xi_w, \psi_w, \psi, r, b\right]^T, \mathbf{w} = \left[w_w, w_r, w_b\right]^T, u = \delta,$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ -\omega_0^2 & -2\lambda\omega_0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\frac{1}{T} & -\frac{K}{T} \\ 0 & 0 & 0 & 0 & -\frac{1}{T_b} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{K}{T} \\ 0 \end{bmatrix},$$

$$\mathbf{E} = \begin{bmatrix} 0 & 0 & 0 \\ 2\lambda\omega_0\sigma & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \end{bmatrix}.$$

The system (4) is driven by control input u as well as noise $\mathbf{w}$. As it can be seen from matrix C, the heading ψ is not available as an output; hence a state observer is needed to estimate this variable. To this aim, a Luenberger observer can be designed [12]. The state equation of this observer can be written as:

$$\dot{\hat{\mathbf{x}}} = (\mathbf{A} - \mathbf{J}\mathbf{C})\hat{\mathbf{x}} + (\mathbf{B}, \mathbf{J}) \begin{pmatrix} u \\ y \end{pmatrix}, \tag{5}$$

where $\mathbf{J}$ is such that $\mathbf{A} - \mathbf{J}\mathbf{C}$ has desired (i.e. negative) eigenvalues, to ensure convergence of the observer's state towards the original system's state.

C. Balanced spiking networks

Neural network described in [11] implements a linear dynamical system of the form

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{c}(t), \mathbf{x} = (x_1, , x_J).$$

The network contains N neurons, and the estimate of x is calculated:

$$\dot{\hat{\mathbf{x}}} = -\lambda_d \hat{\mathbf{x}} + \mathbf{\Gamma}\mathbf{o}(t), \tag{6}$$

where $\mathbf{o}(t) = (o_1(t), \ldots, o_N(t))$ are the network's spike trains: $o_i(t) = \sum_k \delta(t - t_i^k)$, $\mathbf{\Gamma}$ is the $J \times N$ output weights matrix. Neuron's potential differential equation is given by:

$$\dot{V}_i = -\lambda_V V_i + \sum_{k=1}^{N} W_{ik} * o_k(t) + \mathbf{\Gamma}_i^T \mathbf{c}(t) + \sigma_V \eta(t),$$

$$W_{ik}(u) = \Omega_{ik}^s h_d(u) - \Omega_{ik}^f \delta(u), \tag{7}$$

$$\mathbf{\Omega}^f = \mathbf{\Gamma}^T\mathbf{\Gamma}, \mathbf{\Omega}^s = \mathbf{\Gamma}^T(\mathbf{A} + \lambda_d \mathbf{I})\mathbf{\Gamma}.$$

where $\mathbf{W}$ is a connection matrix, $\eta(t)$ is white noise with unit variance.

These networks have the following properties similar to properties of real biological networks: high trial-to-trial variability, asynchronous firing, tight balance between excitation and inhibition.

D. Network Design

We performed software simulations of networks described in subsection II-C solving the systems from subsection II-B: the original 5-d system, and the observer estimating the system's variables from the compass measurement and control input. For this, a python code using Brian simulator [14] was developed. With some limitations of neuromorphic hardware implementations in mind, we set the following constraints for the network: 1) limited firing rate to keep energy cost low, 2) connection weights need to be set in advance (no learning), 3) the network should track the solution with a required precision, 4) network contains 1000-10000 neurons.

Testing showed that the network produces more accurate results if the output weight vectors of all neurons have approximately equal euclidean norms. Also, notice that with each spike, the estimate of the solution is corrected roughly by the corresponding neuron's output weight vector (see (6)). These two facts would cause problems with dynamical systems in which system variables exhibit different dynamics. The solution is to scale the variables of the dynamical systems using a linear transform chosen according to the following requirements: 1) the result of each spike should be large enough relative to the solution size so that the network can keep up with all variables' rate of change and decoder leaks under the condition of limited firing rate, 2) it should be small enough to avoid large relative error, 3) firing thresholds should be small enough compared to the input so that neuron leaks don't create a large error.

The matrix $\mathbf{\Gamma}$ is set by choosing random vectors $\mathbf{\Gamma}_i$ with endpoints uniformly distributed on a unit sphere. Network weights are set according to the parameters of wave response and ship models using (7). The scaling transform is computed using known information about the dynamical systems and networks.

III. Simulations and Results

We took an example of Mariner class cargo ship [13]: $K = 0.185$ Hz, $T = 107.3$ s, with bias constant $T_b = 100.0$ s. Linear wave model constants were set to: $\lambda = 0.1, \omega_0 = 1.2$ rad/s.

Initial conditions for the original and observer systems were:

$$\mathbf{x}(0) = [0, 0, 0, 0, 1]^T, \hat{\mathbf{x}}(0) = [0, 0, 0, 0, 0]^T.$$

Input (see Fig. 2) was

$$u = \begin{cases} sin(\frac{\pi t}{30}) & : 0 \leq t \leq 30 \\ 0 & : t > 30. \end{cases}$$

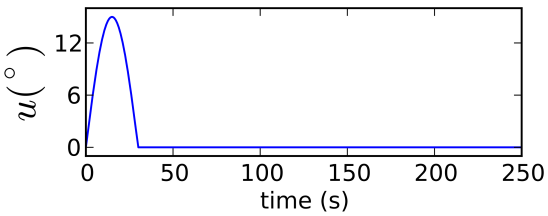

Fig. 2. Control input of the original system

Standard deviations of the noise were: $\sigma_{w_w} = 3.0$ deg, $\sigma_{w_r} = 0.001$ deg/s, $\sigma_{w_b} = 0.05$ deg. Network constants: $\nu = 0.0001, \sigma_V = 0.1, time\_step = 0.01$ s, $\lambda_d = 0.04$ Hz, $\lambda_v = 0.08$ Hz.

The linear transform $\mathbf{z} = \mathbf{D}\mathbf{x}$ with

$$\mathbf{D} = \begin{bmatrix} 4.80769231 & 0 & 0 & 0 & 0 \\ 0 & 4.80769231 & 0 & 0 & 0 \\ 0 & 0 & 0.64935065 & 0 & 0 \\ 0 & 0 & 0 & 200 & 0 \\ 0 & 0 & 0 & 0 & 10 \end{bmatrix}$$

was chosen using knowledge about the dynamical systems, and expected parameters of neural network implementation. We assumed that $|\psi_\omega| < 4$ deg, $|\dot{\psi}_\omega| < 4$ deg/s, $|\xi_\omega| < 4$ deg·s, $|\dot{\xi}_\omega| < 4$ deg, $|\psi| < 720$ deg, $|\dot{\psi}| < 2$ deg/s, $|r| < 2$ deg/s, $|\dot{r}| < 0.02$ deg/s^2, $|b| < 2$ deg, $|\dot{b}| < 1$ deg/s.

We made 500 simulation runs of networks implementing the observer and the original systems with different instances of generated driving noise, and calculated mean and standard deviation of the heading NRMSE.

For the observer system, mean heading NRMSE over 500 runs was 0.013 with standard deviation of 0.0019. For the original system these values were 0.016 and 0.004 respectively. Fig. 3 shows distribution of NRMSE for the observer system.

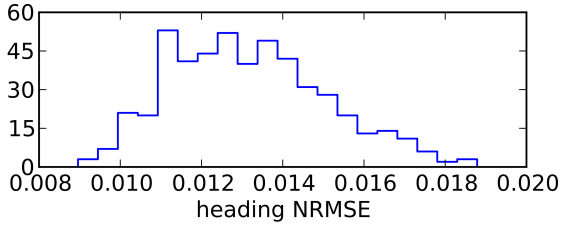

Fig. 3. Histogram of heading NRMSE for the observer system over 500 runs

Fig. 4 shows results of one simulation run for the observer system.

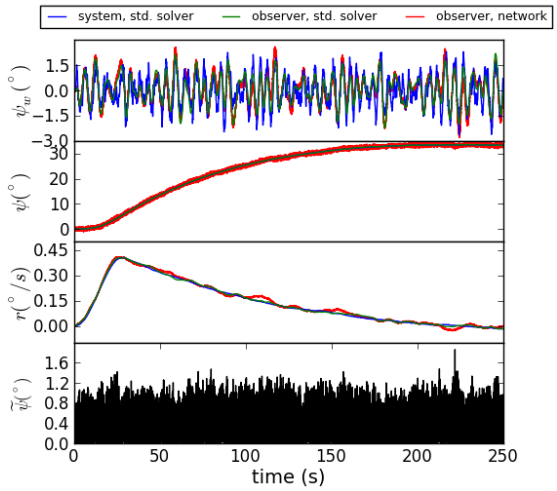

Fig. 4. Results of spiking neural network calculations for the observer system

IV. Discussion

Our results show that the neural network described in [11] can be used for dynamical systems observation. The software simulations of networks implementing example systems representing a ship yielded mean NRMSE of the ship heading under 2%.

These neural networks can be implemented in hardware using analog neurons [15]. Such an architecture can take advantage of robustness to noise and low power consumption. Another advantage of this approach is that there is no need to train the network, since weights are preset using direct computation derived from the control theory and spiking networks framework. Future work will involve testing this approach on neuromorphic hardware, in order to validate the potential energy gain when compared with traditionnal signal processing approaches.

References

[1] W. Gerstner, "Spiking neurons," in *Pulsed Neural Networks*, W. Maass and C. M. Bishop, Eds. MIT Press (Cambridge), 1999, pp. 55–85.

[2] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10, pp. 1659–1671, 1996.

[3] C. Eliasmith and C. H. Anderson, *Neural Engineering (Computational Neuroscience Series): Computational, Representation, and Dynamics in Neurobiological Systems*. Cambridge, MA, USA: MIT Press, 2002.

[4] S. Furber, "To build a brain," *Spectrum, IEEE*, vol. 49, no. 8, pp. 44–49, 2012.

[5] H. Markram, "The blue brain project," *Nature Reviews Neuroscience*, vol. 7, no. 2, pp. 153–160, Feb. 2006.

[6] D. S. Modha, R. Ananthanarayanan, S. K. Esser, A. Ndirango, A. Sherbondy, and R. Singh, "Cognitive computing," *Commun. ACM*, vol. 54, no. 8, pp. 62–71, 2011.

[7] J. J. Hopfield, "Pattern recognition computation using action potential timing for stimulus representation," *Nature*, vol. 376, no. 6535, pp. 33–36, Jul. 1995.

[8] T. Natschläger and B. Ruf, "Spatial and temporal pattern analysis via spiking neurons," *Network*, vol. 9, no. 3, pp. 319–332, Jan. 1998.

[9] H. Hagras, A. Pounds-Cornish, M. Colley, V. Callaghan, and G. Clarke, "Evolving spiking neural network controllers for autonomous robots," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 5, 2004, pp. 4620–4626 Vol.5.

[10] D. Floreano, Y. Epars, J. christophe Zufferey, and C. Mattiussi, "Evolution of spiking neural circuits in autonomous mobile robots," *International Journal of Intelligent Systems*, vol. 21, pp. 1005–1024, 2006.

[11] M. Boerlin, C. Machens, and S. Deneve, "Predictive coding of dynamic variables in balanced spiking networks," unpublished.

[12] D. G. Luenberger, *Introduction to Dynamic Systems: Theory, Models, and Applications*. Wiley, 1979.

[13] T. I. Fossen, "Lecture notes: Ttk4190 guidance and control of vehicles," 2013. [Online]. Available: http://www.itk.ntnu.no/fag/gnc/Wiley/slides.html

[14] D. Goodman and R. Brette, "Brian: a simulator for spiking neural networks in python," *Frontiers in Neuroinformatics*, vol. 2, 2008.

[15] R. Heliot, A. Joubert, and O. Temam, "Robust and low-power accelerators based on spiking neurons for signal processing applications," in *International Workshop on Design for Reliability (DFR)*, January 2011.