2016 IEEE 55th Conference on Decision and Control (CDC)
ARIA Resort & Casino
December 12-14, 2016, Las Vegas, USA

Spiking Neural Network (SNN) Control of a Flapping Insect-scale Robot

Taylor S. Clawson *Student Member, IEEE,* Silvia Ferrari *Senior Member, IEEE,*
Sawyer B. Fuller, and Robert J. Wood *Senior Member, IEEE*

***Abstract*— The flapping microrobot known as RoboBee is the first robot to demonstrate insect-scale flight, as well as the most capable flying robotic insect to date. Controlled hover, trajectory-following, and perching have been accomplished by means of onboard sensors and actuators fabricated with the robot using a "pop-up book MEMS" process based on smart composite microstructures. This paper presents a RoboBee bio-inspired controller that closes the loop between the onboard sensors and actuators by means of a leaky integrate-and-fire spiking neural network that adapts in flight using a reward-modulated Hebbian plasticity mechanism.**

I. Introduction

Flying insects are some of the most agile members of the natural world, capable of dodging swatting hands, landing on wind-blown flowers, and flipping upside-down to land on ceilings [1], [2]. Insects are able to maintain stable control of their flight paths during complicated maneuvers despite their relatively small nervous systems [3]. To achieve this performance on a robotic platform, one approach is to develop custom accelerator-based chips that are computationally efficient and fast and combine the scalability of homogeneous multi-core architectures with the high performance of system-on-a-chip power-efficient hardware accelerators [4]. Another approach, explored in this paper, is to develop neuromorphic chips inspired by biological systems, in an attempt to replicate not only their size and power consumption but also the functionalities of biological brains. Through the use of the training algorithm presented in Section IV, spiking neural networks can learn to perform adaptive control and potentially provide robust and reconfigurable control for the RoboBee [5]–[7].

In an effort to replicate these capabilities, many examples of bio-inspired robots and micro aerial vehicles (MAVs) have been realized at various scales, e.g., 10 grams and above [8]–[14]. The benefits for reducing the size of flying robots to gram and sub-gram or *insect* scale include increased platform robustness and survivability (since the strength to weight ratio is inversely proportional to size), as well as improved agility, covertness, and access to confined spaces. These benefits, combined with the consideration that insect-scale robots pose no danger when collisions or failures do happen, make them ideal platforms for a broad range of sensing, reconnaissance, and surveillance applications [3], [15], [16].

Conventional propulsion and manufacturing technologies are not viable at the gram and sub-gram (or millimeter) scale, because of inefficiencies that arise from force scaling [15]. In recent years, the "smart composite microstructures" or SCM design and manufacturing methodology, developed by the Harvard Microrobotics Lab, has been shown highly effective at fabricating robotic insects capable of stable flight and high-speed locomotion. In particular, the "RoboBee", considered in this paper (Fig. 1), uses only an onboard visual sensor comprised of a pyramidal structure with four phototransistors to estimate pitch and roll rates by measuring changes in light intensity from a fixed source. These estimates are used to control piezoelectric actuators for the wings, allowing for stable hovering. Additional sensors have recently been developed and used to demonstrate improved flight capabilities, such as perching and trajectory-following using proximity sensors and an on-board IMU [17], [18].

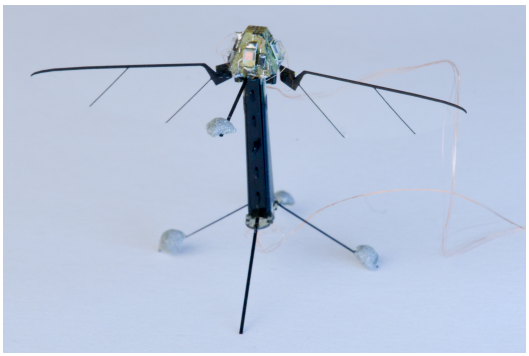

Fig. 1. The RoboBee.

Because of their small size, insect-scale robots are in principle well suited for agile maneuvers due to the low mass and inertia of the robot. To avoid hazardous conditions such as turbulence, gusts, or obstacles, these robots require accurate sensors with low latency. This typically translates into more processing and battery power, as well as greater weight. Examples of fully autonomous flight systems at larger (e.g. sub-meter) scales have shown that maneuvers beyond attitude stabilization require position estimates with low latency that are typically obtained by external motion-capture systems, such as Vicon or OptiTrack [8]–[13], [19]. In addition to being too heavy and inefficient, traditional

Taylor S. Clawson is with the Laboratory for Intelligent Systems and Control (LISC), Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY 14853 tsc83@cornell.edu

Silvia Ferrari is the director of the Laboratory for Intelligent Systems and Control (LISC), Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY 14853 ferrari@cornell.edu

Sawyer B. Fuller is with the Department of Mechanical Engineering, University of Washington, Seattle, WA

Robert J. Wood is with the John A. Paulson School of Engineering and Applied Sciences and the Wyss Institute for Biologically Inspired Engineering, Harvard University, Cambridge, MA 02138

sensors to date have also required external reference systems, such as a fixed light source and surface markers [9].

Event-based or *neuromorphic* sensors and computer chips (Fig. 2) are emerging technologies that have been shown to drastically reduce size and power requirements by performing computations based on asynchronous events that, similarly to neuron spikes, occur when a threshold is exceeded at the hardware level [20]. By this approach, it has been shown that a significant amount of computation and compression occurs in the chip, before the signal is ever digitized. Moreover, event-based programmable VLSI systems are now being fabricated at the nanoscale, with power consumptions and device densities that approach those of biological brains [21], [22].

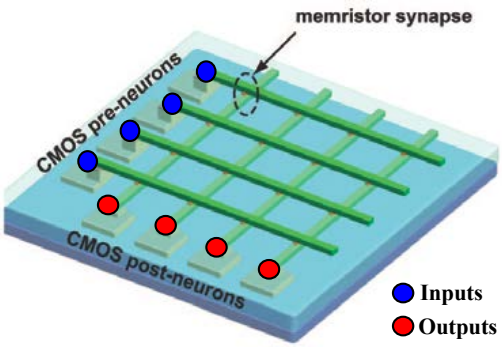

Fig. 2. Neuromorphic circuit implementation of LIF SNN controller.

This paper presents a method for developing a neuromorphic controller modeled by a leaky integrate-and-fire (LIF) spiking neural network (SNN), as shown in [20]. The method consists of training the SNN controller online to follow a Linear Quadratic Regulator (LQR) controller with known performance guarantees. A bio-inspired learning rule, based on reward-modulated Hebbian plasticity, is used to adjust the synaptic weights such that the SNN control performance matches that of the LQR controller with high accuracy. The simulation results demonstrate the SNN ability to learn and adapt quickly to changes in the reference input, while in flight. Thus, they also provide an important basis for the future development of SNN controllers that can control and adapt online to parameter variations and unmodeled RoboBee dynamics.

II. Flapping-Wing Robot

Micro Aerial Vehicles (MAVs) such as quad rotors have become increasingly popular for both consumer and commercial use due to their affordability, maneuverability, and the relative ease with which they can be controlled. When scaled down to an insect-scale, however, significant disadvantages make these designs infeasible. The electromagnetic motors and bearings that are typical in such designs become inefficient at such small scales as surface forces such as friction begin to dominate volumetric forces such as gravity and inertia. Flapping-wing designs inspired by insects do not suffer from the same force scaling issues as they can be operated without any rotating parts or motors. robots on this scale are desirable for a number of applications due to their extremely small size. For example, in a search and rescue situation, insect-scale robots would be capable of searching tight spaces for survivors where other robots would be too large.

A. Robot Description

Insect-scale flapping wing robots have potential uses in a variety of applications, including reconnaissance, search and rescue, and agricultural assistance. Their development can also provide insight into the flight mechanics of insects and other flapping-wing organisms that would be difficult to obtain through experimental methods such as wind tunnel tests. The small scale of these robots provides flexibility that larger robots do not. For instance, the lower cost of each individual robot would allow an entire swarm of hundreds or thousands of insect-scale robots to be deployed at the same cost as a single larger robot. The swarm would also be better suited to many tasks. In the situation of searching for a target, only a single member of the swarm must succeed for the entire swarm to have successfully completed its task. The RoboBee, a millimeter-scale, 80 mg flapping-wing robot, is an example of a robot currently being developed for these potential applications.

A simulation of the RoboBee is used in this paper. The robots are manufactured from laser-cut carbon fiber using the process described in [23], [24]. Because of the difficulties involved in deriving an accurate nonlinear model for the entire flight envelope and estimating the parameters of a newly fabricated robot, a nonlinear, adaptive SNN controller may prove better capable of controlling the RoboBee in these and other challenging situations.

The RoboBee control methods implemented thus far have been shown to be capable of stabilizing the robot during hovering and basic lateral maneuvers by relying on linearized models obtained for the trim conditions of hovering flight [3], [15], [17]. An iterative learning method has also been demonstrated to perch the robot on a vertical wall [18]. A nonlinear adaptive controller capable of learning while in flight may allow the RoboBee to perform more advanced maneuvers mimicking biological insects. Furthermore, it may compensate for flight parameter errors, thereby eliminating the need for detailed system identification required by previous designs.

The coordinate system shown in Fig. 3 is used to describe the motion of the robot, where $\{\hat{x}, \hat{z}\}$ is an inertial reference frame and $\{\hat{x}', \hat{z}'\}$ is a body-fixed frame. The wings rotate about the $\hat{z}'$ axis while flapping. The wing membranes are made of a flexible material that allows them to twist about the top of the wing while in motion, thus generating lift. They are operated near the resonant frequency of the wing structures to maximize the generated lift force. The robot is capable of generating torque about all three axes by modulating the relative flapping frequency and mean flapping angle of each wing as described in [17]. The robot is described in greater detail in [3], [17].

B. Rigid Body Dynamics

The governing equations for the flapping-wing robot used in the simulation are derived using rigid-body dynamics. The full model includes six degrees of freedom in three dimensions, but by assuming that the torque about the $\hat{z}'$ axis is negligible, the dynamics can be reduced to a planar model with three degrees of freedom. The planar model includes a single rotational degree of freedom about the y axis, θ and two translational degrees of freedom in the directions of the $\hat{x}'$ axis and $\hat{z}'$ axis. Figure 3 shows the planar model of the robot and corresponding coordinate system used for deriving the equations of motion.

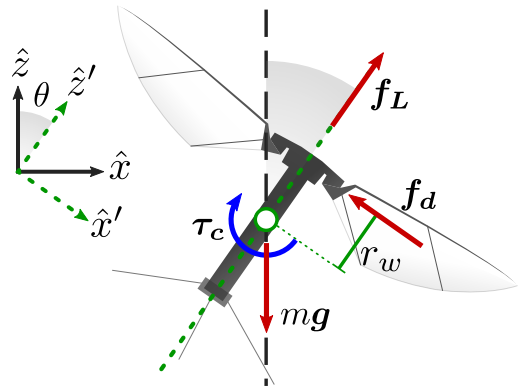

Fig. 3. Free body diagram of the RoboBee.

The vibration caused in the robot body due to flapping has been experimentally validated to be negligible compared to the wing motions. Neglecting the motion of the wings allows for the use of a stroke-averaged model [17] to represent aerodynamic forces. Rigid body dynamics are used for the main body of the robot. The effects of the wings on the dynamics of the robot can be modeled by the use of two separate forces and a torque, all acting at the intersection of the two wings. A lift force $\mathbf{f}_L \in \mathbb{R}^3$ is generated as the wings flap and the wing membrane flexes about the top of the wing. This lift force is proportional to the flapping rate of both wings. In reality, this force also depends on the velocity of the robot body, but this contribution is neglected for simplicity. For the small spaces and relatively low speeds used in the simulations presented in Section V, the body velocity is an order of magnitude smaller than the velocity of the wings, which are flapping at 120 Hz.

A stroke-averaged drag force $\mathbf{f}_d \in \mathbb{R}^3$ was found to be nearly linearly proportional to the incident airspeed v_w in [3]. The drag force can thus be calculated as $\mathbf{f}_d = -b_w v_w \hat{x}'$, where b_w is a constant that was obtained from the wind tunnel tests. The wings are capable of generating a torque $\boldsymbol{\tau}_c \in \mathbb{R}^3$ about the y axis by increasing the flapping rate of one wing relative to the other. Additional aerodynamic forces are neglected in this model as they are typically small in steady, hovering flight.

Using linear and angular momentum balance on the body following Newton's second law for rigid-body dynamics as described in [25] yields the following equations for the forces and torques described previously:

$$m\dot{\mathbf{v}} + \boldsymbol{\omega} \times m\mathbf{v} = \mathbf{f}_d + m\mathbf{g} + \mathbf{f}_L \tag{1}$$

$$I_y\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times I_y\boldsymbol{\omega} = \boldsymbol{\tau}_c + (\mathbf{r}_{w/G} \times \mathbf{f}_d) \tag{2}$$

where m is the mass of the robot, $\mathbf{v} \in \mathbb{R}^3$ is the linear velocity of the body, $\boldsymbol{\omega} = \dot{\theta}y$, and $\boldsymbol{\omega} \in \mathbb{R}^3$ is the body rotation rate with respect to the inertial frame. The acceleration of gravity is $\mathbf{g} \in \mathbb{R}^3$, I_y is the moment of inertia of the robot about the y axis, and $\mathbf{r}_{w/G} \in \mathbb{R}^3$ is the position of the center of the wings with respect to the center of gravity. $\mathbf{f}_L$ is physically limited to $f_{L,max} = 1.5 \cdot m\|\mathbf{g}\|$ and $\boldsymbol{\tau}_c$ is limited to $\tau_{c,max} = 1\ \mu Nm$.

The aerodynamic drag $\mathbf{f}_d$ is a key component in the model as it causes the robot to be naturally unstable. As the robot tilts, $\mathbf{f}_L$ rotates and creates increased lateral acceleration. As the robot accelerates laterally, the drag force increases until it generates a torque large enough to rotate the robot in the opposite direction. The torque generated by the drag force results in growing oscillations until the robot tumbles.

For the simulation, equations (1) and (2) are rearranged to the form $\dot{\mathbf{v}} = \boldsymbol{f}(\mathbf{q}, \mathbf{f}_L, \mathbf{f}_d, \mathbf{g})$ and $\dot{\boldsymbol{\omega}} = \boldsymbol{f}(\mathbf{q}, \mathbf{f}_d, \boldsymbol{\tau}_c)$. Once in this form, the equations can be used to update the state vector

$$\mathbf{q} = [\theta, \dot{\theta}, x, z, \dot{x}, \dot{z}]^T \tag{3}$$

using numerical integration. The final form of the governing equations for the rigid body dynamics is

$$\dot{\mathbf{v}} = \frac{\mathbf{f}_d + \mathbf{f}_L}{m} + \mathbf{g} - \boldsymbol{\omega} \times \mathbf{v} \tag{4}$$

$$\dot{\boldsymbol{\omega}} = I_y^{-1}[\boldsymbol{\tau}_c + (\mathbf{r}_{w/G} \times \mathbf{f}_d) - \boldsymbol{\omega} \times I_y\boldsymbol{\omega}] \tag{5}$$

III. Spiking Neural Network Control Structure

The control system used for the simulated RoboBee consists of two decoupled controllers. The altitude controller is responsible for controlling the lift force $\mathbf{f}_L$ through the control output u_{alt} and the lateral controller modulates the torque $\boldsymbol{\tau}_c$ through the control output u_{lat}. This structure is a simplification of the controller demonstrated in [15]. That controller uses a lateral controller consisting of a proportional term opposing error from a reference orientation and a derivative term that opposes rotational velocity. This controller was successful in stabilizing a RoboBee in both hovering flight and basic lateral maneuvers. Several other control methods have also been used on both manufactured and simulated RoboBees.

An LQR method has been used as the lateral controller for simulations of the RoboBee and is capable of stabilizing the linearized model more effectively than the PD controller. One of the most capable controllers used on a manufactured RoboBee to date is an adaptive method using sliding mode control [17]. This method showed an improvement in steady state error compared to the PD controller when used in hovering flight and basic lateral maneuvers. It was also used to successfully land the RoboBee and adapted to differences in robot parameters.

The assumptions made in deriving the planar rigid body dynamics shown in Section II-B create a reasonably accurate model in regions of the state space near hovering flight, but break down during rapid maneuvers or in the presence of strong disturbances.

In simulation, the input signals are processed before reaching the SNN to scale them to an appropriate range and encode them in event sequences known as spike trains. In future experiments, the architecture can be modified to receive event sequences directly from the onboard sensors, without need for coding and decoding. When converting continuous input signals, as required in simulations, if the input values from the state variables are too high, then the network will become saturated and the input groups will reach their maximum firing rate before being able to replicate the input. If the input values are small or negative, the input nodes will never fire and the signal will not propagate throughout the network. It would also be possible to adjust the firing thresholds on the neuron groups so that they are scaled to the appropriate range for the input, but it is convenient to design the network without concern for the expected strength of the input signals. To address this issue, the actual signal $\hat{q}_i$ passed into the neural network is the shifted and scaled value of the original signal q_i for each state variable, so that $\hat{q}_i = a_i(q_i + b_i)$, where a_i and b_i are positive constants. This pre-processing ensures that state variables in the expected range will produce a positive current in the network and cause the input groups to fire.

The SNN is designed as a feed forward network composed of three layers: an input layer, a hidden layer, and an output layer. The input layer consists of four groups of neurons containing 50 neurons each, responsible for receiving input from each of the lateral state variables: θ, $\dot{\theta}$, x, and $\dot{x}$. The neurons in the input layer are connected to the neurons in the hidden layer with a probability of 30%. Lowering the number of connections decreases the required computation time to run the simulation. The hidden layer contains a single group of 100 neurons. Neurons in the hidden layer are connected to the neurons in the output layer, which contains a single group of 50 neurons, with the same probability of 30%. Instead of using single neurons for each input and for the output, groups of neurons were used to facilitate *population rate coding* as described in Section III-B.

Each neuron is governed by the LIF equation, with an increasing decay rate τ in the later layers in the system such that $\tau_{input} < \tau_{hidden} < \tau_{output}$. This helps to normalize the current flowing through the network so that it does not become amplified in the hidden and output layers.

The spike trains from the neurons in the output layer are decoded using the approach described in Section III-B and the decoded output is used as the lateral control input as shown in Fig. 4. The plant uses the output from both the altitude controller and the lateral controller to adjust $\mathbf{f}_L$ and $\boldsymbol{\tau}_c$, respectively.

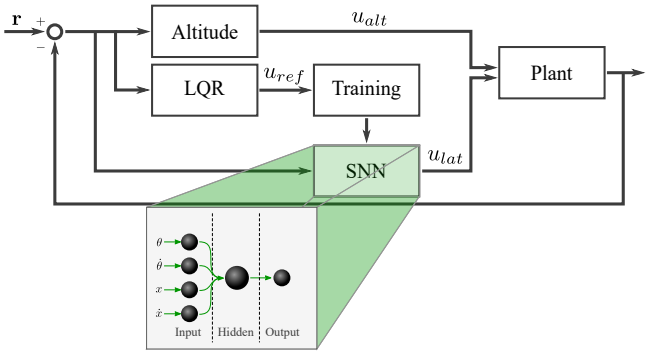

Fig. 4. The control structure, including the spiking neural network structure.

A. Neuron Model

There are many different models used to represent the dynamics of an individual neuron [26], [27]. The *Hodgkin-Huxley* model is a well-known example that models different types of ion current flowing across the neuron membrane, including a sodium channel, a potassium channel, and a leakage channel. This model captures the basic spike generation properties for certain types of neurons, and can be expanded to capture the behavior of more types of biological neurons by increasing the number of ion channels [26]. The *spike response* model gives a simple formulation for the membrane potential in neurons. It generates a spike when the membrane potential crosses a predefined threshold and includes a refractory time after each spike during which the neuron will not fire. A special case of the spike response model is the *leaky integrate-and-fire* model, which is based on the dynamics of a simple RC circuit. This model is used in this paper for its low computational complexity. In the model, the change in the neuron membrane potential v over time is governed by,

$$\frac{dv}{dt} = \frac{RI - v}{\tau} + \xi \tag{6}$$

where I is the input current, R is the resistance, τ is a time constant controlling the "leak" rate, and ξ is an additive disturbance scalar used to represent noise in the neural circuit. The second part of the *leaky integrate-and-fire* model resets the membrane potential v to a rest value v_r after reaching a threshold v_t. A *firing time* is defined as the time at which v reaches v_t as follows:

$$t^f : \quad v(t^f) = v_t \tag{7}$$

Following a *firing time*, the dynamics of the neuron again follow equation (6). The *leaky integrate-and-fire* model is shown in Fig. 5.

B. Spike Train Decoding

Biological neural networks encode information through sequences of spike times denoted by $\mathbf{t}^f = [t_0, t_1, ...t_n]$, generated by all neurons within the network. In order to be useful in representing continuous-time functions for control input, the output from the neural network must be decoded.

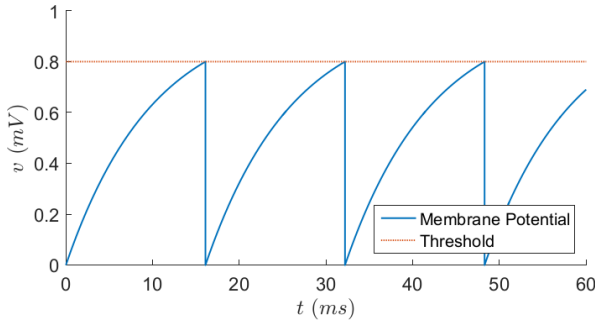

Fig. 5. Leaky integrate and fire model.

While the precise method that biological systems use to decode spike trains is not yet fully understood, one useful group of decoding methods is based on mean firing rates using a temporal average. This category of methods is called rate coding and includes schemes based on decoding the output from a single neuron or a population of neurons. However, rate coding methods neglect any information that may be contained in the precise timing of spikes. There is some evidence to indicate that the precise timing of spikes in different neurons is important to how biological systems encode information. Decoding methods based on precise timing of spikes are categorized as spike codes. A detailed description of several different methods from each group is provided in [26].

As described in [26], population rate coding has certain advantages over rate coding for a single neuron and thus, it is adopted in this paper. For a single neuron, the time window over which the spike count is averaged must be relatively large in order to create a smooth function. This has the drawback of an output that is slow to respond to changes in the instantaneous firing rate of the neuron. Using a population of homogenous neurons, all receiving the same input, the firing rate can be taken as the average firing rate of the entire population. The increase in spikes of the population compared to a single neuron allows the averaging window to be much smaller and the decoded output can therefore respond much more quickly to changes in the instantaneous firing rate.

In this paper, the leaky integrator equation is used to decode population firing rates into a continuous output as follows

$$\hat{y}(t) = \alpha \sum_{t^f \in S_i(T)} e^{\beta(t^f - t)} - \gamma \tag{8}$$

where for all t^f, $t^f < t$. The scaling constant α is used to scale the output so that the neural network can generate the appropriate range of output values for the target function. The decay constant β can be tuned for the appropriate trade-off between a faster output response and less noise, and an offset constant γ offsets the entire output curve such that the minimum firing rate of the system can be mapped to any real number.

Figure 6 shows $\hat{y}(t)$ for an arbitrary spike train. The jagged output is the byproduct of a lower firing rate. Using a population rate instead of the firing rate of a single neuron solves this problem by increasing the number of spikes in the spike train so that the output $\hat{y}(t)$ does not decay noticeably between spike times.

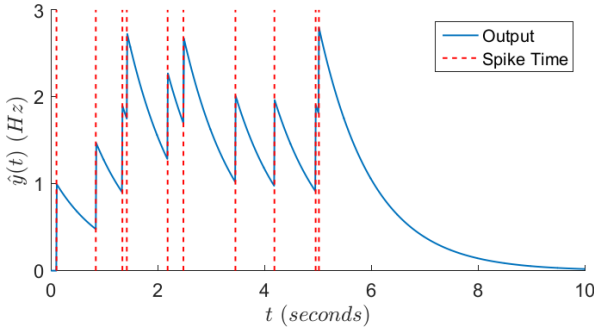

Fig. 6. Spike train decoding using rate coding.

During the simulation, the decoding method shown in equation (8) is running constantly in order to give an accurate measure of the current system output. It is critical for the training algorithm to have the current system output $\hat{y}(t)$ at all times during the simulation so that it can adjust the synaptic weights and alter the output to approach the target output y.

IV. Spiking Neural Network Training Algorithm

In the proposed approach, the SNN controller is trained based on an existing LQR design with known performance guarantees. The weights in the neural network are adjusted directly following a reward-modulated Hebbian approach presented in [28], which imitates a chemical reward based on the error between a reference value y and the decoded network output $\hat{y}$, defined as $e = (y - \hat{y})$. The output from the LQR controller computed from the RoboBee state is fed back to the training algorithm as the reference value y. The LQR design is based on the RoboBee lateral state $[\theta, \dot{\theta}, x, \dot{x}]$ in continuous time, t. The interaction between the LQR controller and the rest of the system is shown in figure 4.

The LQR controller is tuned to have a steady state error of much less than 1% and provides guaranteed stability, as the model used in the controller perfectly matches the simulation. Additionally, since the LQR control law is a linear operation ($u_{ref} = \mathbf{K} \cdot \mathbf{e}$), it is a relatively easy function for the SNN to learn. These traits make the LQR controller a good candidate for a reference control input to the SNN.

The SNN training algorithm minimizes the error between the SNN decoded output, $\hat{y}$, and the LQR reference inputs, y, adjusting the synaptic weights based on the reward function,

$$r(t) = [sgn(y - \hat{y}) + r(t - \Delta t)] \cdot e^{(\hat{t}_i - t)/\tau} \tag{9}$$

where $\hat{t}_i$ is the time of the last presynaptic spike occurring before the current time t, and τ is a time constant that controls the decay rate of the reward. This function, inspired by biological mechanisms for synaptic plasticity, has the effect of changing the weights in the direction of the sign

of the error $(y - \hat{y})$ slowly when the time since the last spike is large and quickly when the time since the last spike is small. Figure 7 shows the reward function over time if $sgn(y - \hat{y}) > 0$ for an arbitrary spike train.

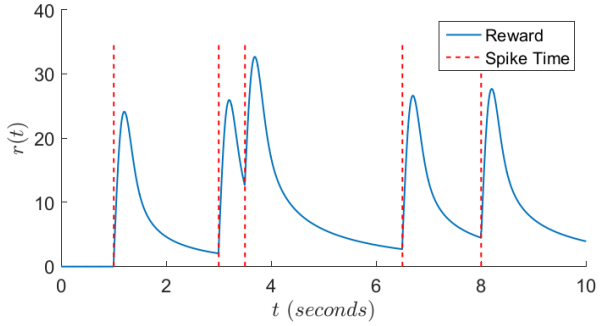

Fig. 7. Reward function for an arbitrary spike train.

Finally, a learning rate constant μ is included in the weight change equation for more control over the speed at which the network is allowed to change its weights. Thus, the complete function for the change in synaptic weights is

$$\Delta w_{ij}(t) = \mu \cdot r(t) \tag{10}$$

where the subscript ij denotes that this is the change in weight of the synapse between neurons i and j.

V. Simulation Results

The spiking neural network described in Section III is simulated using the brian neural simulator developed in [29]. At every time step, the simulator calls a function to update the RoboBee state and another function to compute the weight changes for each set of synapses following equation (10). At the onset of the simulation, the neural network is initialized with synaptic weights sampled from a standard uniform distribution $\mathcal{U}(0, 1)$. The neuron firing thresholds were all initialized by sampling a uniform distribution $\mathcal{U}(0.7, 1.7)$. The network was not trained *a priori*. Thus, its ability to successfully control the simulated RoboBee when starting from a semi-random state demonstrates the ability of the network to quickly adapt to unknown dynamics and flight conditions.

The initial robot configuration is characterized by a nonzero attitude θ_0. The goal of the robot controller is to navigate to two target states in sequence, each one with a different x and z value, but with zero attitude and velocity. Based on the general form of the state vector $\mathbf{q}$ given in equation (3), the first study is characterized by the following initial state and two target states:

$$\mathbf{q}_0 = [-0.6 \quad 0 \quad 0 \quad 0.02 \quad 0 \quad 0]^T$$
$$\mathbf{q}_1 = [0 \quad 0 \quad 0.05 \quad 0.08 \quad 0 \quad 0]^T$$
$$\mathbf{q}_2 = [0 \quad 0 \quad 0.01 \quad 0.05 \quad 0 \quad 0]^T$$

For the first $500\ ms$ of the simulation, the state $\mathbf{q}$ was held constant at $\mathbf{q}_0$ so that the leaky integrator used to decode the output of the network had sufficient time to build up to an initial steady-state value accurately representing the output of the network corresponding to $\mathbf{q}$.

The trajectory of the robot during simulation is overlaid on the trajectory of the robot as controlled by the LQR controller in Fig. 8, which shows the ability of the SNN controller to closely follow the LQR controller. The initially skewed attitude of the robot forces it to begin its movement in the negative $\hat{x}$ direction, but the controller is able to successfully correct the trajectory to settle just above $\mathbf{q}_1$. For the purposes of this paper, it is unimportant that the robot slightly overshoots the target $\mathbf{q}_1$. The key metric is the ability of the spiking neural network to control the robot in a manner that closely mimics the LQR-controlled robot. Improved reference controllers would reduce the steady-state error and overshoot, and the spiking neural network would likely be able to follow the slightly modified control outputs similarly well.

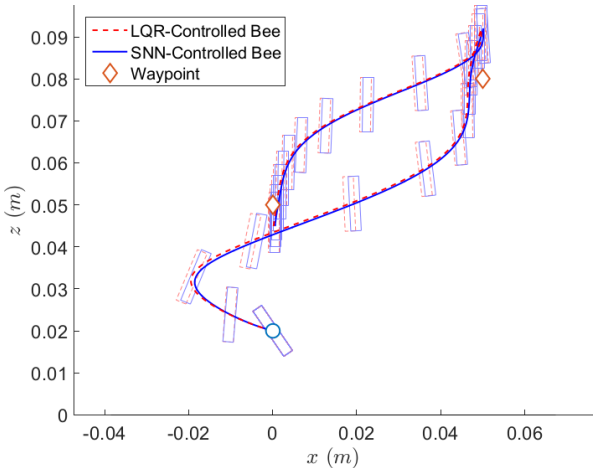

Fig. 8. The RoboBee trajectory for the first simulation.

The output-error time history plotted in Fig. 9 illustrates the difference between the desired LQR control inputs and the SNN control inputs. Large deviations in error are visible at both $0.5\ s$ and $1.5\ s$, when the robot receives a new target waypoint. Otherwise, the error resembles gaussian noise centered at approximately 0. The output of the spiking neural network, $\hat{y}(t)$, is then plotted against the target output from the LQR controller y in Fig. 10. What appears to be noise in the output is an artifact of the rate coding used to decode the network output. A larger group of output neurons would produce a smoother decoded output signal. Below the network output in Fig. 10, the time histories of the closed-loop RoboBee state variables, θ and x, show the relationship between the change in the state values and SNN control input $\hat{y}(t)$.

A second simulation was run to test the ability of the controller to transition between different waypoints. The structure of the SNN was identical to that used for the first simulation (Section III), and again, the training algorithm described in Section IV was used to adapt the weights of the network online. For this simulation, the waypoints were chosen to exemplify landing of the RoboBee. The robot is

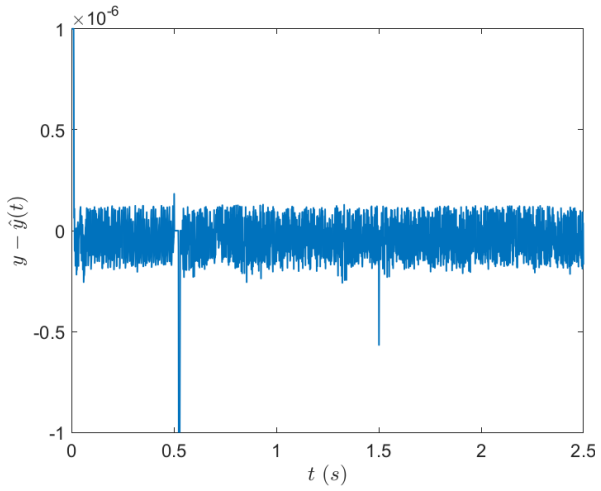

Fig. 9. The output error $(y - \hat{y}(t))$.

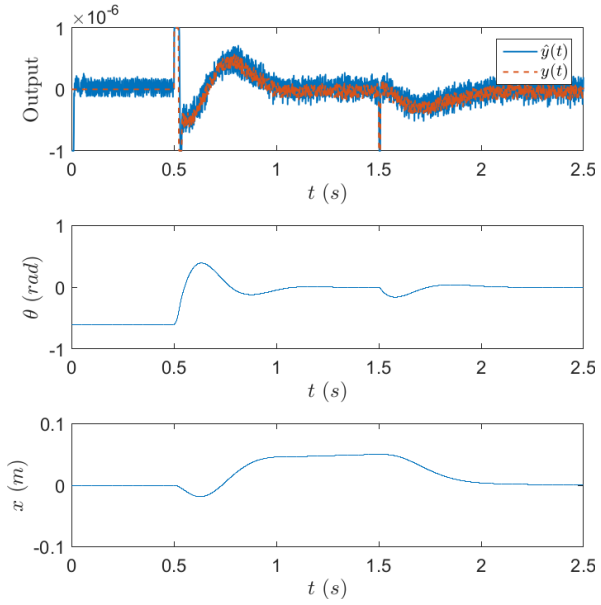

Fig. 10. Comparing the lateral control signals.

first guided to a hovering position slightly above the ground before a second target waypoint causes the robot to slowly lower and contact the ground at a low velocity $\dot{z}$ while in an upright position. The resulting trajectory is shown in Fig. 11, demonstrating that the SNN controller is capable of closely following the target LQR controller. Moreover, these results demonstrate the ability of the bio-inspired SNN controller and training algorithm to learn from a reference input accurately and rapidly online, as required for adaptive in-flight control.

VI. Conclusions

This paper presents a RoboBee SNN controller that closes the loop between the onboard sensors and actuators by means of a leaky integrate-and-fire spiking neural network. The spiking neural network adapts in flight using a reward-modulated Hebbian plasticity mechanism that is biologically

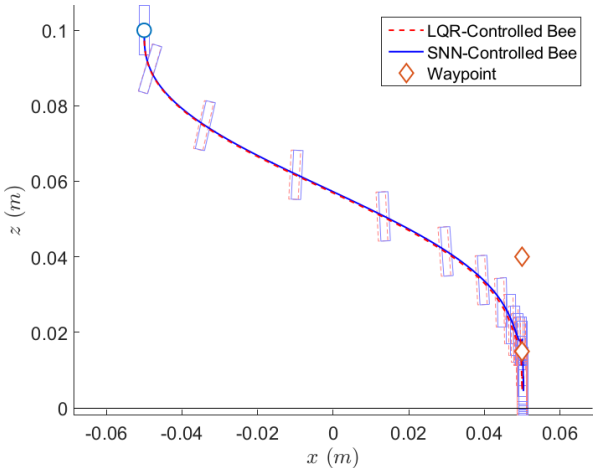

Fig. 11. Trajectory for the second simulation.

inspired and rapid enough to allow for online implementations. The proposed SNN architecture is modeled based on neuromorphic chips and biological brains such that it can potentially enable nanoscale hardware fabrication and studies aimed at reverse-engineering the insect brain and flight apparatus. Furthermore, the spiking neural network ability to learn rapidly and accurately from a target reference signal provides the opportunity for the future development of control systems that can account for significant parameter variations, unmodeled dynamics, and unstructured environmental uncertainty.

Acknowledgment

This research was funded by the National Science Foundation grants ECCS-1545574, ECCS-1028506 and CMMI-1251729.

References

[1] M. F. Land and T. Collett, "Chasing behaviour of houseflies (fannia canicularis)," *Journal of Comparative Physiology*, vol. 89, no. 4, pp. 331–357, 1974.

[2] S. Dalton, *Borne on the Wind*. Reader's Digest Press; distributed by Dutton, 1975.

[3] S. B. Fuller, M. Karpelson, A. Censi, K. Y. Ma, and R. J. Wood, "Controlling free flight of a robotic fly using an onboard vision sensor inspired by insect ocelli," *Journal of the Royal Society Interface*, vol. 11, 2014.

[4] M. Hempstead, M. J. Lyons, D. Brooks, and G.-Y. Wei, "Survey of hardware systems for wireless sensor networks," *Journal of Low Power Electronics*, vol. 4, no. 1, pp. 11–20, 2008.

[5] S. Ferrari and R. F. Stengel, "Online adaptive critic flight control," *Journal of Guidance, Control, and Dynamics*, vol. 27, no. 5, pp. 777–786, 2004.

[6] ——, "An adaptive critic global controller," in *Proceedings of the 2002 American Control Conference (IEEE Cat. No. CH37301)*, vol. 4. IEEE, 2002, pp. 2665–2670.

[7] S. Balakrishnan and V. Biega, "Adaptive-critic-based neural networks for aircraft optimal control," *Journal of Guidance, Control, and Dynamics*, vol. 19, no. 4, pp. 893–898, 1996.

[8] A. Censi, J. Strubel, C. Brandli, T. Delbruck, and D. Scaramuzza, "Low-latency localization by active led markers tracking using a dynamic vision sensor," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 891–898.

[9] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-slam–based navigation for autonomous micro helicopters in gps-denied environments," *Journal of Field Robotics*, vol. 28, no. 6, pp. 854–874, 2011.
[10] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The grasp multiple micro-uav testbed," *Robotics & Automation Magazine, IEEE*, vol. 17, no. 3, pp. 56–65, 2010.
[11] J. P. How, B. Bethke, A. Frank, D. Dale, and J. Vian, "Real-time indoor autonomous vehicle test environment," *Control Systems, IEEE*, vol. 28, no. 2, pp. 51–64, 2008.
[12] B. Hérissé, T. Hamel, R. Mahony, and F.-X. Russotto, "A terrain-following control approach for a vtol unmanned aerial vehicle using average optical flow," *Autonomous robots*, vol. 29, no. 3-4, pp. 381–399, 2010.
[13] S. Lupashin and R. DAndrea, "Adaptive fast open-loop maneuvers for quadrocopters," *Autonomous Robots*, vol. 33, no. 1-2, pp. 89–102, 2012.
[14] J. M. Grasmeyer, M. T. Keennon *et al.*, "Development of the black widow micro air vehicle." *Progress in Astronautics and aeronautics*, vol. 195, pp. 519–535, 2001.
[15] K. Y. Ma, P. Chirarattananon, S. B. Fuller, and R. J. Wood, "Controlled flight of a biologically inspired, insect-scale robot," *Science*, vol. 340, no. 6132, pp. 603–607, 2013. [Online]. Available: http://science.sciencemag.org/content/340/6132/603
[16] A. T. Baisch, O. Ozcan, B. Goldberg, D. Ithier, and R. J. Wood, "High speed locomotion for a quadrupedal microrobot," *The International Journal of Robotics Research*, p. 0278364914521473, 2014.
[17] P. Chirarattananon, K. Y. Ma, and R. J. Wood, "Adaptive control of a millimeter-scale flapping-wing robot," *Bioinspiration & Biomimetics*, vol. 9, no. 2, p. 025004, 2014. [Online]. Available: http://stacks.iop.org/1748-3190/9/i=2/a=025004
[18] ——, "Fly on the wall," in *2014 5th IEEE RAS & EMBS, International Conference on*, Aug 2014, pp. 1001–1008.
[19] G. C. de Croon, M. Groen, C. De Wagter, B. Remes, R. Ruijsink, and B. W. van Oudheusden, "Design, aerodynamics and autonomy of the delfly," *Bioinspiration & biomimetics*, vol. 7, no. 2, p. 025003, 2012.
[20] P. Mazumder, D. Hu, I. Ebong, X. Zhang, Z. Xu, and S. Ferrari, "Digital implementation of a virtual insect trained by spike-timing dependent plasticity," *Integration, the VLSI Journal*, 2016.
[21] J. V. Arthur and K. Boahen, "Learning in silicon: Timing is everything," *Advances in neural information processing systems*, vol. 18, p. 75, 2006.
[22] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano letters*, vol. 10, no. 4, pp. 1297–1301, 2010.
[23] P. S. Sreetharan, J. P. Whitney, M. D. Strauss, and R. J. Wood, "Monolithic fabrication of millimeter-scale machines," *J. Micromech. Microeng.*, vol. 22, no. 5, 2012.
[24] K. Y. Ma, S. M. Felton, and R. J. Wood, "Design, fabrication, and modeling of the split actuator microrobotic bee," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 1133–1140.
[25] W. F. Phillips, *Mehanics of Flight*. John Wiley & Sons, Inc., 2010.
[26] W. Gerstner and W. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge, UK: Cambridge University Press, 2006.
[27] P. Dayan and L. F. Abbott, *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. Cambridge, MA: MIT Press, 2001.
[28] G. Foderaro, C. Henriquez, and S. Ferrari, "Indirect training of a spiking neural network for flight control via spike-timing-dependent synaptic plasticity," in *Decision and Control (CDC), 2010 49th IEEE Conference on*, Dec 2010, pp. 911–917.
[29] M. Stimberg, D. F. M. Goodman, V. Benichoux, and R. Brette, "Equation-oriented specification of neural models for simulations," *Frontiers in Neuroinformatics*, vol. 8, no. 6, 2014.