Quadratic Leaky Integrate-and-Fire Neural Network tuned with an Evolution-Strategy for a Simulated 3D Biped Walking Controller

Lukasz Wiklendt, Stephan K. Chalup, and María M. Seron
School of Electrical Engineering and Computer Science
The University of Newcastle
Callaghan, NSW 2308, Australia

Abstract

This paper presents the results of experiments in applying a spiking neural network to control the locomotion of a simulated biped robot. The neural model used in simulations was developed to allow for an analytic solution to a neuron's fire time, while maintaining a non-instant post-synaptic potential rise time. The synaptic weights and delays were tuned using an evolution-strategy. Simulation experiments demonstrate that already within about two thousand generations the biped is able to acquire a dynamic walk which allows it to walk upright for several metres.

1 Introduction

Controlling the walk of a 3D biped with point-contact feet is a difficult control problem, as the robot is highly unstable, requiring continuous control to maintain an upright walk. In this computer simulation study, experimental results from the application of a spiking neural network to controlling the walk of such a biped are presented.

The motivation to use spiking neural networks in this study is to broaden the available set of tools which can be applied to robot control problems from a machine learning perspective. An evolved spiking neural network coupled with a conventional control method has been shown to work when applied to the acrobot swing-up and balance task [15]. In this study, the spiking neural network controller is not augmented with conventional control methods.

The field of biped walking control is quite advanced, with many successful solutions implementing sophisticated control methods that require thorough analysis of the dynamics of the particular biped models used, such as virtual constraints [13], or Zero Moment Point preview control [5]. In contrast, biped dynamics were not investigated in this study in order to derive a walking control strategy. Instead the control strategy emerged from the evolution of synaptic weights and delays.

Other machine learning approaches to solve the biped walking problem include reinforcement learning of the Poincaré map [11], a hand-designed non-spiking neural network topology with online adaptive weights [10], and evolving a neural oscillator network [4].

Another contribution of this paper is the presentation of a spiking neuron model which features a non-instantaneous post-synaptic potential rise time and an analytic solution to the future firing time of neurons. Maass [7] showed that spiking neural networks are computationally more powerful than sigmoid neural networks for certain types of problems, provided that post-synaptic potential rise times are non-instantaneous, and that delay coding is used for input and output neurons. Although delay coding is not used in this study, the neuron model presented here could be useful for other studies where such coding is necessary.

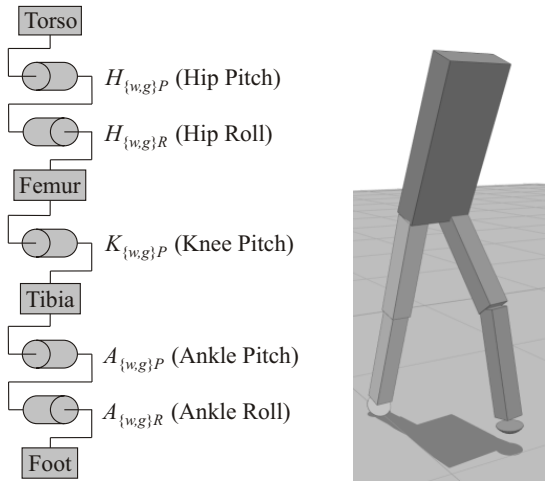
This paper is organised as follows. Section 2 presents the simulated biped used in experiments. Section 3 describes the neural network controller in detail, including the neuron models used. Section 4 describes the evolution-strategy used to modify the synaptic weights and delays. Results of the evolution-strategy and walking simulations are presented in Section 5, and the conclusion in Section 6.

2 Biped Robot Simulation

The 3D biped robot (Figure 1) used in simulations in this study is a 7-link biped with 6 joints totalling 10 degrees of freedom (DOF). It was simulated in a dynamics package called Open Dynamics Engine (ODE) [14] with a time-step of 10ms. It has a 2-DOF universal joint at each hip and ankle, and a 1-DOF hinge joint at each knee. The biped's feet are spherical caps resulting in a single-point contact per foot with the flat ground used in simulations. Initially, the biped is oriented such that its anterior (forward-pointing) axis lies along the global x-axis, its lateral axis lies along the global z-axis, and its superior (up) axis lies along the

978-0-7695-3326-1/08 $25.00 © 2008 IEEE
DOI 10.1109/HIS.2008.146

global y-axis.



(a) Joint-body relations of one leg (b) Simulator image

Figure 1: On the left are shown the relations between joint axes and bodies. On the right is a simulator screenshot of the biped taking a step.

The sensor and actuator data between the biped and the controller are given with respect to the current *support* leg, and its opposite *swing* leg. At each time-step the support leg is set to the leg with the highest foot-to-ground contact reaction force. If the forces on both feet are equal then the support leg is the same one as the last time-step.

The angles of joint axes at any time-step during the simulation are given by: H_{wP}, K_{wP}, H_{gP}, A_{gP}, H_{wR}, H_{gR}, A_{gR}. The H, K and A represent the hip, knee and ankle joints. The subscripts w and g represent the swing and support (grounded) legs respectively, and the subscripts P and R represent the pitch and roll axis of a particular joint. The chain of rigid bodies and joints from the torso to a foot are shown in Figure 1a.

Joint motors are driven by supplying the desired velocity of the motor. The ODE package would attempt to speed up each motor to the desired velocity as fast as possible without going over the motor's maximum torque. In simulations the maximum torque of all joint motors was set to 100Nm. To speed up learning, the maximum velocity of joints was limited to levels which are not far from normal human-like walking. The maximum velocities of the ankle joints were limited to 1rad/s, hips were limited to 3rad/s, and knees to 6rad/s. The sped up learning was achieved because when no such restrictions were made the biped would learn to make unrealistically large steps or even sometimes leap or hop.

The rigid bodies of the biped are modeled as cuboids, with a uniform density of $1000\text{kg}/\text{m}^3$. The total mass of the biped is approximately 25kg, and it stands at approximately 1.46m. Gravity was set to -9.81ms^{-2} along the global y-axis.

Spherical cap feet are used to provide smooth continuous contact with the ground. This helped to eliminate the dedicated ankle actuation required to maintain useful ground contact when using flat rectangular feet found on most 3D biped robots with actuated ankles.

3 Spiking Neural Network Controller

A spiking neural network is a directed graph, with vertices representing neurons, and edges representing synapses. Information is transmitted between neurons with the timing of spikes sent via synapses in the direction of the edge. Each synapse has two parameters, a weight and a delay. When a neuron fires, a spike is sent to each adjacent neuron, and the time that those neurons receive the spike depends on the delay of the synapse along which each spike was sent. The time that a neuron fires is based on previously received spikes. The weight of a synapse modifies the efficacy with which a received spike helps to cause a neuron to fire, the exact time this happens depends on the neuron model detailed in Section 3.1. In this study both weights and delays of the synapses were modified during the learning session, but were kept constant during simulation.

The neural network was used to control the angular position of each joint's axis, based on the state of the biped at each time-step. The network was split into two parts, the x-coordinate deals with forward-backward movement and is controlled by the pitch axis of joints, and the z-coordinate deals with the side-to-side movement and is controlled by the roll axis of joints. The y-coordinate is required in both forward-backward and side-to-side movement, and so those elements are connected to both parts of the network.

Two of the joints were not controlled by the neural network, and instead were given a trivial hand-designed controller to achieve a particular task. The support knee was always kept extended, and as soon as the swing leg switched to support then this controller would take over and apply a velocity equal to 10 times the difference between the current knee angle and the desired knee angle of 0rad, but saturating at the maximum 6rad/s velocity for knees. The swing ankle was reset to its initial state in a similar manner. Automatic support knee extension was implemented to reduce the search space of the evolution-strategy, and was observed to help speed up learning when compared with pilot experiments that allowed for neural control of the support knee.

The control-simulation loop was performed as follows. After each time-step that the biped simulation takes, the biped's state was read and sent as input to the network. Then the network was simulated for 10 time units, after which the network output was read and sent as input to the biped for the next time-step. The network state was not reset between successive invocations during a biped simulation run. De-

tails of the input and output values are given in sections 3.2 and 3.3.

An image of the neural network topology is given in Figure 2, showing the input-layer on the left, a single two-part hidden-layer, and a two-part output-layer on the right, where $K_{m,n}$ represents the connections of a complete directed bipartite graph from m to n vertices.

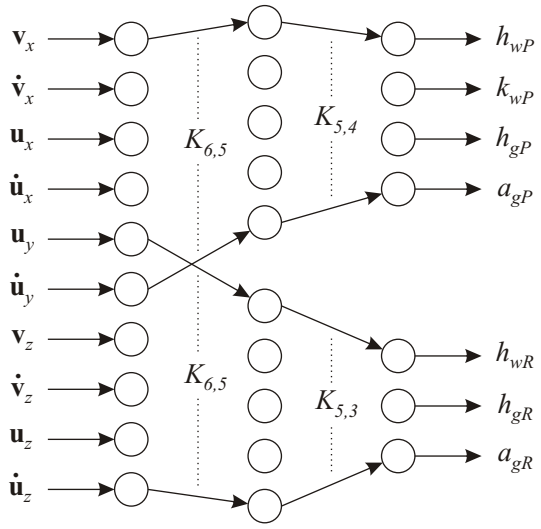


Figure 2: Neural network topology.

3.1 Quadratic LIF Neuron Model

The neuron model used in this study was derived from the leaky integrate-and-fire (LIF) neuron model [2, 3]. The LIF neuron model consists of a membrane-potential for each neuron. Incoming spikes temporarily raise (or lower) the membrane potential which then exponentially decays back to rest. If the membrane potential reaches the threshold value then the neuron fires, and a period of refraction is entered where the membrane potential is subsequently reduced to prevent further immediate firing. The membrane potential between spike events is given by

$$u(t) \quad = \quad -ae^{-\mu t} + be^{-\sigma t} - \nu e^{-\rho t} \tag{1}$$

where a, b, and ν are state values for the membrane potential at time $t = 0$. The time-constants μ, σ, and ρ decide the speed of decay. The neuron fires when $u(t) = \theta$, for some real valued θ. Simulating this neuron model requires finding the solution to the equation $u(t) = \theta$, which gives the time t when the neuron will fire. A time-step based simulation such as the one in [15] can be used to decay the membrane potential while checking to see if it has reached the threshold. However, an event-based simulation was used in this study instead, since it is easier to implement real-valued learnable synaptic delays.

The model shown in equation (1) cannot be analytically solved to find the future firing time of a neuron given its current state. Calculating the future firing time of a neuron with a gradual (non-instant) membrane potential rise time such as this requires either an iterative approach [9], look-up tables [12], or some simplification of the neural model. This study implemented the latter approach, since a spiking neural network was used as a machine learning tool rather than in an investigation into the similarities between the model and biological neurons.

The simplification developed for this study features a gradual membrane potential rise and decay time, along with learnable weights and delays. This fits within the requirements needed to make the neural network model more computationally powerful than sigmoid neural networks as discussed in [7]. However, the inputs and outputs used in this study were not delay-coded as suggested by [6], and for which the result of computational power was derived. Delay-coded inputs and outputs were initially attempted in this study, but better experimental results were achieved with the input and output coding described in Sections 3.2 and 3.3.

The simplification is achieved with two changes. Firstly, the $-\nu e^{-\rho t}$ term in equation (1) governing the refraction is removed, and when the neuron fires, its state is reset to rest. Secondly, the values of the remaining decay time-constants are chosen such that $\mu = 2\sigma$. What follows is a detailed description of the model used in this study.

The state of neuron i at time t is given by the variables $a_i(t), b_i(t) \in \mathbb{R}$. The neuron is at rest when $a_i = 0$ and $b_i = 0$. In the traditional sense $b_i(t) - a_i(t)$ can be thought of as the membrane potential, and when this reaches the threshold θ then the neuron fires a spike. Given some initial state of neuron i at time t_0, it decays to a time $t_0 + s$ in the following manner, provided there are no incoming spikes during this time

$$a_i(t_0 + s) \quad = \quad a_i(t_0)\, e^{-2\mu_i s} \tag{2}$$
$$b_i(t_0 + s) \quad = \quad b_i(t_0)\, e^{-\mu_i s} \tag{3}$$

where $\mu_i > 0$ is a parameter specifying the decay rate, and $s \geq 0$. In this study, the value $\mu = 1$ was used. The unit of time is arbitrary and only serves to provide the mechanism for change of a neuron's state between two instances in time as a result of decay. An incoming spike arriving at time $t_0 + s$ from neuron j modifies the state of neuron i to

$$a_i(t_0 + s) \quad = \quad w_{ji} + a_i(t_0)\, e^{-2\mu_i s} \tag{4}$$
$$b_i(t_0 + s) \quad = \quad w_{ji} + b_i(t_0)\, e^{-\mu_i s} \tag{5}$$

where w_{ji} is the weight of the synapse from neuron j to i along which the spike arrived. A firing time $t_0 + s$ of neuron

i can be determined from the roots of the equation

$$f(s) \quad = \quad -a_i(t_0)\, e^{-2\mu_i s} + b_i(t_0)\, e^{-\mu_i s} - \tfrac{1}{4} \qquad (6)$$

The size of f is dependant on the synaptic weights, so the value of θ can be chosen arbitrarily, and is set to $\theta = \frac{1}{4}$ such that given an initial state $a_i(t_0) = 1$ and $b_i(t_0) = 1$ then $f(s) \leq 0$. Equation (6) can be analytically solved, which results in two solutions, one of them given by

$$s \quad = \quad \left[\log 2 + \log\left(b - \sqrt{b^2 - a}\right)\right] \frac{1}{\mu_i} \qquad (7)$$

where $a = a_i(t_0)$ and $b = b_i(t_0)$. The maximum height of f occurs when $\frac{df(s)}{ds} = 0$ provided that $a > 0$.

$$\frac{df(s)}{ds} \quad = \quad 2ae^{-2s\mu}\mu - be^{-s\mu}\mu \qquad (8)$$

$$= \quad \left(2ae^{-s\mu} - b\right)\mu e^{-s\mu} \qquad (9)$$

solving $\frac{df(s)}{ds} = 0$ for finite s yields

$$s \quad = \quad \frac{1}{\mu}\log_e\left(\frac{2a}{b}\right) \qquad (10)$$

resulting in a maximum height of

$$f\left(\frac{1}{\mu}\log_e\left(\frac{2a}{b}\right)\right) \quad = \quad \frac{1}{4}\left(\frac{b^2}{a} - 1\right) \qquad (11)$$

It is interesting to note that the maximum height occurs at the same time regardless of the synaptic weight w which is added to both a and b when a spike arrives, provided the neuron is at rest state when the spike arrives.

When an incoming spike arrives which modifies the receiving neuron's state, the necessary and sufficient conditions required for the neuron to fire at some time in the future given no other spike activity are:

- $b > 0$
- $b^2 \geq a > \frac{b}{2}$

When these conditions hold, then equation (7) represents the time since t_0 that the membrane potential first reaches the threshold (from below), resulting in a firing time of $t_0 + s$. If one of these conditions does not hold then the neuron will never fire without additional incoming spikes. The second solution of equation (6) would yield the second time the membrane potential reaches the threshold (from above) if no refraction was used.

If neuron i fires at time t_f, then both variables of the neuron's state are reset to 0, forcing the neuron to the rest state

$$a_i(t_f) \quad = \quad 0 \qquad (12)$$

$$b_i(t_f) \quad = \quad 0 \qquad (13)$$

and a spike is received at a neuron k at time $t_f + d_{ik}$ for all receiving neurons, where d_{ik} is the synaptic delay from neuron i to neuron k.

To ensure correct dynamics of a neuron, the initial state must be set such that $b_i(t_0) - a_i(t_0) < \frac{1}{4}$, which sets the neuron's membrane potential below the threshold. In this study the initial state of each neuron was always set to its rest state.

The future firing time of a neuron needs to be recalculated every time the neuron receives a spike, since it will most likely either alter the firing time, or cause the neuron to never fire without additional incoming spikes.

Hidden-layer neurons are modeled exactly as described in this section. Output-layer neurons differ only in that they never fire, that is, their threshold is infinite $\theta = \infty$. Input-layer neurons behave very differently and are described in the next section.

3.2 Network Input

Input to the network was derived from two vectors $\mathbf{u},\ \mathbf{v} \in \mathbb{R}^3$, shown in Figure 3. The biped's centre of mass, $\mathbf{c}_M$, projected to the ground along the gravity vector gives the centre of gravity $\mathbf{c}_G$. The vector from $\mathbf{c}_M$ to the location of the centre of mass of the swing foot is given by $\mathbf{u}$. The vector from the ground contact position of the support foot to $\mathbf{c}_G$ is given by $\mathbf{v}$. Before sending the input to the network both $\mathbf{u}$ and $\mathbf{v}$ were rotated about the global y-axis, so that their x-coordinate lies parallel to the torso's anterior axis. This allows the controller to ignore the direction the biped is facing with respect to global coordinates.

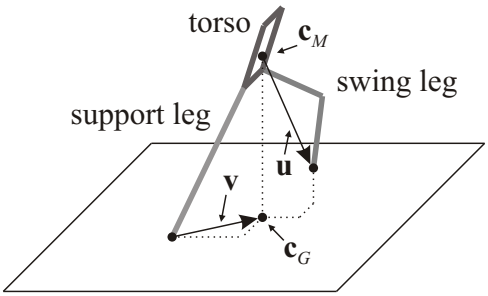


Figure 3: Vectors $\mathbf{u}$ and $\mathbf{v}$ used as input to the neural network.

The total 10 input values to the neural network are given by: $\mathbf{v}_x$, $\dot{\mathbf{v}}_x$, $\mathbf{u}_x$, $\dot{\mathbf{u}}_x$, $\mathbf{u}_y$, $\dot{\mathbf{u}}_y$, $\mathbf{v}_z$, $\dot{\mathbf{v}}_z$, $\mathbf{u}_z$, $\dot{\mathbf{u}}_z$, where $\dot{\mathbf{u}}$ and $\dot{\mathbf{v}}$ are the first time derivatives, and $\mathbf{v}_y = \dot{\mathbf{v}}_y = 0$ always holds.

After each biped simulation time-step all inputs are written to the network, and the network is simulated for 10 time units. The input neurons do not have any time-dependent state such as the hidden and output layer neurons. All input

neurons fire spikes along their incident synapses at the same time, which occurs at the start of network simulation.

The way the input is observable by the network is that the synaptic weights of the synapses incident to each input neuron are momentarily multiplied by the value of the corresponding input value. The effect is that a receiving hidden-layer neuron i perceives a spike arriving along a synapse from an input neuron j with a weight equal to $\alpha_j w_{ji}$, where α_j is the value of the input for neuron j.

3.3 Network Output

There are 7 output values from the neural network given by: h_{wP}, k_{wP}, h_{gP}, a_{gP}, h_{wR}, h_{gR}, a_{gR}. The h, k and a represent the desired hip, knee and ankle joint angles. For a given output value x, and it's corresponding joint angle X, the joint was driven by setting the velocity of its motor to $\gamma(\mathrm{X} - \mathrm{x})$, where γ is a scaling factor, which was set to $\gamma = 10$.

The support leg hip motors were treated a little differently. Instead of using the network's output value as the desired joint angle, h_{gP} and h_{wR} were treated as the desired offsets from angles required to keep the torso vertical. So the velocities were set to $\gamma(\Theta - h_{gP})$ and $\gamma(\Phi - h_{gR})$, where Θ is the Euler angle representing the amount of pitch of the torso in the global frame, and Φ the amount of roll. If $h_{gP} = 0$ and $h_{wR} = 0$, then this control scheme tends the torso towards an upright posture.

The outputs of each output neuron are all read at the same time at the end of network simulation. The value of each output is equal to the membrane potential of the corresponding output neuron at that end time t, that is, for an output neuron i the value of the output is $b_i(t) - a_i(t)$.

4 Evolution Strategy

The synaptic weights and delays were evolved using a $(\mu + \lambda)$ evolution-strategy (ES) [1], with $\mu = 10$ parents and $\lambda = 70$ offspring per generation. Standard self-adaption was used to modify the strategy parameter vectors, and no recombination was used for the strategy parameters and chromosomes.

The fitness evaluation consisted of transcribing the chromosome vector to the weights and delays of the synapses in the network, and resetting the state of each neuron. The number of elements in each chromosome vector was twice the number of synapses. Each weight was directly set to the corresponding element in the chromosome vector. Since delays must always be positive, each delay was set to $e^{\mathbf{x}_i}$ where $\mathbf{x}_i$ is the chromosome element corresponding to that delay.

Once transcription was complete, the biped was set to its default standing posture (shown in the top-left frame of Figure 5), and given a slight impulse at the torso's centre of mass, 5Ns forward and 5Ns to the right. The simulation was then executed for a maximum of 20 seconds, and stopped early if either the biped fell over or the legs made contact with each other. The motivation for adding the impulse was to help the biped to fall over, which stopped the simulation early given small or no network output. The impulse also helped to add some initial asymmetry to the foot-ground contact forces.

The fitness value for each run was given by the formula

$$\text{fitness} \quad = \quad e^{-X} \tag{14}$$

where X was the displacement of the robot along the global x-coordinate at the time the simulation ended. The use of the exponential function has no impact on the learning, since only relative fitness values are important in the standard $(\mu+\lambda)$-ES, but was used to transform the fitness value so that a smaller fitness is better.

5 Results

There were 5 independent evolution runs conducted in this study, which ran for approximately 28 hours wall-time on a multi-core server. The fitness trace of each run is shown in Figure 4. Since the fitness evaluation could be stopped prior to completing the full 20 second simulation then some evolutions ran more generations than others. It was usually the case that the better the fitness the longer the evolution took, since the biped walked a further distance each fitness evaluation.

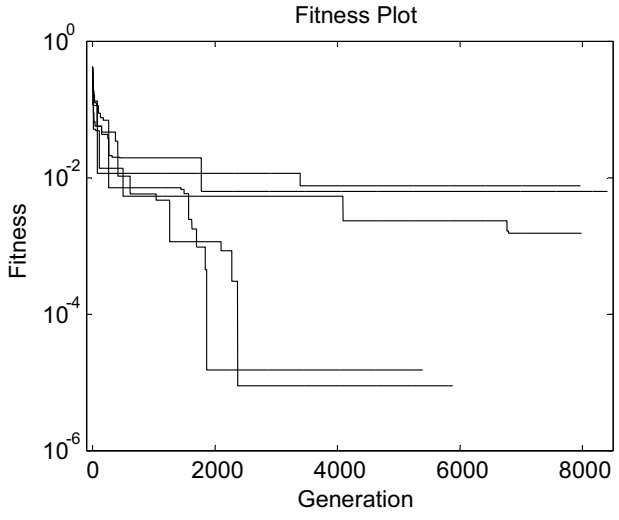


Figure 4: Fitness traces of each of the 5 evolution runs, where lower is better.

Of the 5 evolution runs, 4 produced a good gait similar to each other, although the displacement achieved before

falling varied from 4.5m to 11m. The remaining evolution run was able to get the robot to walk 4.5m, however the gait was awkward since the controller didn't bend the swing knee resulting in more of a hobble than a walk. The best walk was 11m in 8s from a standing start, and the 100ms interval snapshots of the walk are shown in Figure 5.

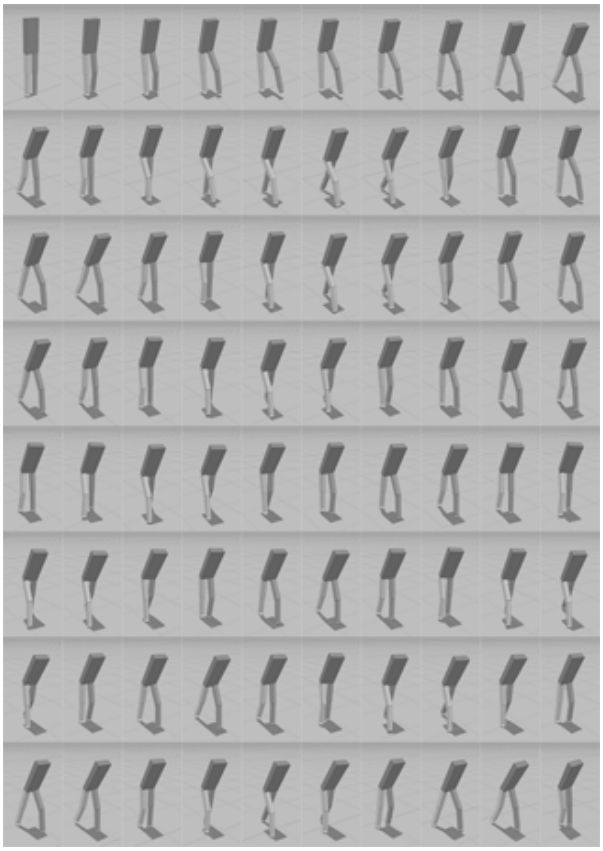


Figure 5: First 8s of a walk, shown at 100ms intervals. Starting at the top-left, the frames progress left to right and then top to bottom.

6 Conclusion

A simplified leaky integrate-and-fire neuron model was developed which allowed for both non-instant membrane potential rise dynamics as well as an analytic solution to the firing time. Being able to analytically solve the future firing time is advantageous for event-based neural network simulations, and the non-instate membrane potential rise has advantages with the computation power of spiking neurons.

This study has demonstrated that a spiking neural network evolved with an evolution-strategy can produce an acceptable walking gait in an unstable single-point foot contact 3D biped without any extra complicated traditional control methods. Even though the biped only walked just over 10 metres before collapsing, the simplicity of the small neural network used in this study suggests that neuron models such as these deserve further investigation for the control of unstable biped robots.

References

[1] H. Beyer and H. Schwefel. Evolution strategies–a comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

[2] W. Gerstner. *Pulsed Neural Networks*, chapter 1: Spiking Neurons, pages 3–53. In Maass and Bishop [8], 2001.

[3] W. Gerstner and W. M. Kistler. *Spiking Neuron Models: Single Neurons, Populations, Plasticity*, chapter 4: Formal Spiking Neuron Models. Cambridge University Press, 2002.

[4] D. Hein, M. Hild, and R. Berger. Evolution of biped walking using neural oscillators and physical simulation. In *RoboCup 2007: Proceedings of the International Symposium*, LNAI. Springer, 2007.

[5] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, 2:1620–1626, 2003.

[6] W. Maass. An efficient implementation of sigmoidal neural nets in temporal coding with noisy spiking neurons. Technical Report NC-TR-96-031, NeuroCOLT Technical Report Series, 1996.

[7] W. Maass. Networks of spiking neurons: the third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997.

[8] W. Maass and C. M. Bishop, editors. *Pulsed Neural Networks*. MIT Press, 2001.

[9] T. Makino. A discrete-event neural network simulator for general neuron models. *Neural Computing & Applications*, 11(3):210–223, 2003.

[10] P. Manoonpong, T. Geng, T. Kulvicius, B. Porr, and F. Wörgötter. Adaptive, fast walking in a biped robot under neuronal control and learning. *PLoS Computational Biology*, 3:1305–1320, 2007.

[11] J. Morimoto and C. Atkeson. Learning biped locomotion. *Robotics & Automation Magazine, IEEE*, 14(2):41–51, 2007.

[12] E. Ros, R. Carrillo, E. Ortigosa, B. Barbour, and R. Agís. Event-driven simulation scheme for spiking neural networks using lookup tables to characterize neuronal dynamics. *Neural Computation*, 18(12):2959–2993, 2006.

[13] C. Shih, J. Grizzle, and C. Chevallereau. Asymptotically stable walking of a simple underactuated 3d bipedal robot. *Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE*, pages 2766–2771, 2007.

[14] R. Smith. "(ODE) Open Dynamics Engine" http://www.ode.org.

[15] L. Wiklendt, S. Chalup, and R. Middleton. A small spiking neural network with LQR control applied to the acrobot. *Neural Computing & Applications*, 2008.