Neuromorphic Computing Handbook

(In Writing)

-

Índice general

Part 1

1. Introduction

2. Neuron Models

Spiking neural network's neuron model describe how the membrance potential of a neuron change over the time. In this chapter, we dive into the biological neuron's dynamics and exhibit several examples neuron models in the spiking neural network.

2.1 Biological Neuron

2.2 Abstract of Neuron Models

In spiking neural network, Equation 2.2.1 to Equation 2.2.4 in the form of **state-space models (SMMs)**. provide a highly abstract description for describe a biological neuron.

In these equation, $\mathbf{x}(t)$ is the state vector at time t, $\dot{x}(t)$ is the state variation at time t, $\mathbf{I}(t)$ is the input current at time t, V is the membrane potential of the neuron, $\mathbf{y}(t)$ is the neuron's output at time t, and V_{out} is the output voltage tha will be sent to the synapses that depature from this neuron. $(\mathbf{x}(t))_i$ is the i-th element of the state vector. $\mathscr{P}$ is the model parameters.

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{I}(t), t, \mathscr{P}) \tag{2.2.1}$$

$$\mathbf{y}(t) = g(\mathbf{x}(t), \mathbf{I}(t), t, \mathscr{P}) \tag{2.2.2}$$

$$s.t., \ (\exists i \in [0, |\mathbf{x}(t)|])(\mathbf{x}(t))_i = V(t) \tag{2.2.3}$$

$$(\exists i \in [0, |\mathbf{y}(t)|])(\mathbf{y}(t))_i = V_{out}(t) \tag{2.2.4}$$

Equation 2.2.3 indicates that, a neuron should maintain a membrane potential V, and Equation 2.2.4 indicates that, the neuron's output should contains a voltage V_{out}.

A discretization version present in Equation 2.2.5 to Equation 2.2.8.

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{I}_t, t, \mathscr{P}) \tag{2.2.5}$$
$$\mathbf{y}_t = g(\mathbf{x}_t, \mathbf{I}_t, t, \mathscr{P}) \tag{2.2.6}$$
$$s.t., \ (\exists i \in [0, |\mathbf{x}_t|])(\mathbf{x}_t)_i = V_t \tag{2.2.7}$$
$$(\exists i \in [0, |\mathbf{y}_t|])_i \mathbf{y} = V_{out_t} \tag{2.2.8}$$

2.3 Neuron Model Examples

2.3.1 Hodgkin-Huxley (HH) Model

The Hodgkin-Huxley (HH) model can be utilize to accurately reproduce the bio-neuron's dynamics. Its mathematical formulation is presented in Equations 2.3.1 to 2.3.5. By combining all these equations, we arrive at Equation 2.3.6.

$$I(t) = C_m \frac{dV_m(t)}{dt} + I_i(t) \tag{2.3.1}$$
$$I_i(t) = I_{Na}(t) + I_K(t) + I_l(t) \tag{2.3.2}$$
$$I_{Na}(t) = g_{Na}(t)(V_m(t) - V_{Na}(t)) \tag{2.3.3}$$
$$I_K(t) = g_K(t)(V_m(t) - V_K(t)) \tag{2.3.4}$$
$$I_l(t) = \bar{g}_l(t)(V_m(t) - V_l(t)) \tag{2.3.5}$$

$$I(t) = C_m \frac{dV_m(t)}{dt} + g_{Na}(t)(V_m(t) - V_{Na}(t)) + g_K(V_m(t) - V_K(t)) + \bar{g}_l(t)(V_m(t) - V_l(t)) \tag{2.3.6}$$

Ion channel function $g.$ are function respect to time t and membrance potential V. Specifically, Equation 2.3.7 is held.

$$g_{Na}(t) = \bar{g}_{Na} m^3(t) h(t) \qquad g_K(t) = \bar{g}_K n^4(t) \qquad g_l(t) = \bar{g}_l \tag{2.3.7}$$

By combining Equations 2.3.1 to 2.3.7, we arrive at Equation 2.3.8.

$$I = C_m \frac{dV_m(t)}{dt} + \bar{g}_{Na}(t) m^3(t) h(t)(V_m(t) - V_{Na}(t)) + \bar{g}_K(t) n^4(t)(V_m(t) - V_K(t)) + \bar{g}_l(t)(V_m(t) - V_l(t)) \tag{2.3.8}$$

$\frac{d\cdot}{dt} = \alpha_{\cdot}(V_m)(1-\cdot) - \beta_{\cdot}(V_m)\cdot$ is held. In which, $\cdot$ is a placeholder for $m(t)$, $n(t)$ and $h(t)$. As such, Equation 2.3.9 to Equation 2.3.11 are held.

$$\frac{dn(t)}{dt} = \alpha_n(V_m(t))(1 - n(t)) - \beta_n(V_m(t))n(t) \tag{2.3.9}$$
$$\frac{dm(t)}{dt} = \alpha_m(V_m(t))(1 - m(t)) - \beta_m(V_m(t))m(t) \tag{2.3.10}$$
$$\frac{dh(t)}{dt} = \alpha_h(V_m(t))(1 - h(t)) - \beta_h(V_m(t))h(t) \tag{2.3.11}$$
$$\tag{2.3.12}$$

From experiment, Equation 2.3.13 to Equation 2.3.18 were obtained.

$$\alpha_n(V_m(t)) = \frac{0{,}01(10 - V_m(t))}{exp(\frac{10-V_m(t)}{10}) - 1} \tag{2.3.13}$$

$$\alpha_m(V_m(t)) = \frac{0{,}1(25 - V_m(t))}{exp(\frac{25-V_m(t)}{10}) - 1} \tag{2.3.14}$$

$$\alpha_h(V_m(t)) = 0{,}07 exp(-\frac{V_m(t)}{20}) \tag{2.3.15}$$

$$\beta_n(V_m(t)) = 0{,}125 exp(-\frac{V_m(t)}{80}) \tag{2.3.16}$$

$$\beta_m(V_m(t)) = 4 exp(-\frac{V_m(t)}{18}) \tag{2.3.17}$$

$$\beta_h(V_m(t)) = \frac{1}{exp(\frac{30-V_m(t)}{10}) + 1} \tag{2.3.18}$$

Finally, we write the state-space model representation from Equation 2.3.19 to Equation 2.3.25 for Hodgkin-Huxley model.

$$x(t) = \begin{bmatrix} V_m(t) & n(t) & m(t) & h(t) \end{bmatrix} \tag{2.3.19}$$

$$\dot{x}(t) = \begin{bmatrix} \dot{V}_m(t) & \dot{n}(t) & \dot{m}(t) & \dot{h}(t) \end{bmatrix} \tag{2.3.20}$$

$$\dot{V}_m(t) = -\frac{1}{C_m}(\bar{g}_K n^4(t)(V_m(t) - V_K) + \bar{g}_{Na} m^3(t)h(t) + \bar{g}_l(V_m(t) - V_l(t)) - I(t)) \tag{2.3.21}$$

$$\dot{n}(t) = \alpha_n(V_m(t))(1 - n(t)) - \beta_n(V_m(t))n(t) \tag{2.3.22}$$

$$\dot{m}(t) = \alpha_m(V_m(t))(1 - m(t)) - \beta_m(V_m(t))m(t) \tag{2.3.23}$$

$$\dot{h}(t) = \alpha_h(V_m(t))(1 - h(t)) - \beta_h(V_m(t))h(t) \tag{2.3.24}$$

$$V_{out}(t) = V_m(t) \tag{2.3.25}$$

2.3.2 Leaky Integrate-and-fire Model

Leaky Integrate-and-fire model is a computational effective model, in which a threshold is set, when membrance potential cross the threshold, the neuro emit a spike. In implementation, we may set the voltage output at time t, V_{out} be 1 mV. The scale problem has the potential to be solved by automatically adjusting the synapses' weights in trainning. A LIF neuron i's output form a **spike train** $S_i(t) = \sum_{t_m} \delta(t - t_m)$. In which t_m is the spike emission time. $\delta(\cdot)$ is the *Dirac Delta* function. A LIF's membrane potential variation can be discribe by the Equation 2.3.26. In which V_{reset} is the reset potential. The neuron's membrane potential will be reset to V_{rest} after a spike emission. A threshold potential V_{th} is defined. Each time the membrane potential exceed the V_{th}, A LIF neuron emit a spike, and reset its membrane potential to V_{rest}

$$\tau_m \frac{\dot{V}_m(t)}{dt} = -(V_m(t) - V_{rest}) + R_m I(t) \tag{2.3.26}$$

In practice, a *refactor period* usually associated with a LIF neuron. The LIF neuron does not emit a spike during the refractory period. Finally, we present the state-space model representation of the LIF neuron in Equations 2.3.65 to 2.3.60.

$$x(t) = \begin{bmatrix} V_m(t) & t_{last\_emit} \end{bmatrix} \tag{2.3.27}$$

$$\dot{x}(t) = \begin{bmatrix} \dot{V}_m(t) & \dot{t}_{last\_emit} \end{bmatrix} \tag{2.3.28}$$

$$\dot{V}_m(t) = \begin{cases} V_{rest}, & V_m(t) + \dot{V}_m(t) \geq V_{th} \; and \; t - t_{last\_emit} \geq t_{refactor} \\ -\frac{1}{\tau_m}(V_m(t) - V_{rest}) + R_m I(t), & otherwise \end{cases} \tag{2.3.29}$$

$$\dot{t}_{last\_emit}(t) = \begin{cases} t - t_{last\_emit}, & V_m(t) + \dot{V}_m(t) \geq V_{th} \; and \; t - t_{last\_emit} \geq t_{refactor} \\ 0, & otherwise \end{cases} \tag{2.3.30}$$

2.3.3 Izhikevich Model

$$x(t) = \begin{bmatrix} V_m(t) & u(t) & t_{last\_emit} \end{bmatrix} \tag{2.3.31}$$

$$\dot{x}(t) = \begin{bmatrix} \dot{V}_m(t) & \dot{u}(t) \end{bmatrix} \tag{2.3.32}$$

$$\dot{V}_m(t) = \begin{cases} V_{rest}, & V_m(t) + \dot{V}_m(t) \geq V_{th} \; and \; t - t_{last\_emit} \geq t_{refactor} \\ 0{,}04V_m^2(t) + 5V_m(t) + 140 - u(t) + I(t), & otherwise \end{cases} \tag{2.3.33}$$

$$\dot{u}(t) = a(bV_m(t) - u(t)) \tag{2.3.34}$$

$$\dot{t}_{last\_emit}(t) = \begin{cases} t - t_{last\_emit}, & V_m(t) + \dot{V}_m(t) \geq V_{th} \; and \; t - t_{last\_emit} \geq t_{refactor} \\ 0, & otherwise \end{cases} \tag{2.3.35}$$

2.3.4 FitzHugh-Nagumo Model

$$x(t) = \begin{bmatrix} V_m(t) & w(t) \end{bmatrix} \tag{2.3.36}$$

$$\dot{x}(t) = \begin{bmatrix} \dot{V}_m(t) & \dot{w}(t) \end{bmatrix} \tag{2.3.37}$$

$$\dot{V}_m(t) = V_m(t) - \frac{V_m^3(t)}{3} - w(t) - bw(t) \tag{2.3.38}$$

$$\dot{w}(t) = \frac{1}{\tau}(V_m(t) - a - bw(t)) \tag{2.3.39}$$

2.3.5 Morris-Lecar Model

$$x(t) = \begin{bmatrix} V_m(t) & N(t) \end{bmatrix} \tag{2.3.40}$$

$$\dot{x}(t) = \begin{bmatrix} \dot{V}_m(t) & \dot{N}(t) \end{bmatrix} \tag{2.3.41}$$

$$\dot{V}_m(t) = \frac{1}{C}[I(t) - g_L(V_m(t) - V_L) - g_{Ca}M_{ss}(V_m(t) - V_{Ca}) - g_K N(t)(V_m(t) - V_k)] \tag{2.3.42}$$

$$\dot{N}(t) = \frac{N_{ss} - N_(t)}{\tau_N} \tag{2.3.43}$$

In which,

$$M_{ss} = \frac{1}{2} \cdot (1 + tanh[\frac{V_m(t) - V_1}{V_2}])N_{ss} = \frac{1}{2} \cdot (1 + tanh[\frac{V_m(t) - V_3}{V_4}])\tau_N = \frac{1}{\varphi cosh[\frac{V - V_3}{2V_4}]} \tag{2.3.44}$$

2.3.6 Hindmarsh-Rose Model

$$\begin{aligned}
x(t) &= \begin{bmatrix} V_m(t) & y(t) & z(t) \end{bmatrix} && (2.3.45)\\
\dot{x}(t) &= \begin{bmatrix} \dot{V}_m(t) & \dot{y}(t) & \dot{z}(t) \end{bmatrix} && (2.3.46)\\
\dot{V}_m(t) &= y(t) + \phi(V_m(t)) - z(t) + I(t) && (2.3.47)\\
\dot{y}(t) &= \_\psi(V_m(t)) - y(t) && (2.3.48)\\
\dot{z}(t) &= r[s(V_m(t) - x_R) - z(t)] && (2.3.49)\\
& && (2.3.50)
\end{aligned}$$

In which,

$$\begin{aligned}
\phi(\_x(t)) &= -a\_x^3(t) + b\_x^2(t) && (2.3.51)\\
\psi(\_x(t)) &= c - d\_x^2(t) && (2.3.52)\\
& && (2.3.53)
\end{aligned}$$

2.3.7 Cable theory

$$\begin{aligned}
x(t) &= \begin{bmatrix} V_m(t) \end{bmatrix} && (2.3.54)\\
\dot{x}(t) &= \begin{bmatrix} \dot{V}_m(t) \end{bmatrix} && (2.3.55)\\
\frac{dV_m(t,x)}{dt} &= \frac{1}{\tau}[V_L - V_m + \lambda^2 \frac{\partial^2 V_m(x)}{\partial x^2}] && (2.3.56)
\end{aligned}$$

2.3.8 Perfect Integrate-and-fire

$$\begin{aligned}
x(t) &= \begin{bmatrix} V_m(t) & t_{last\_emit} \end{bmatrix} && (2.3.57)\\
\dot{x}(t) &= \begin{bmatrix} \dot{V}_m(t) & \dot{t}_{last\_emit} \end{bmatrix} && (2.3.58)\\
\dot{V}_m(t) &= \begin{cases} V_{rest}, & V_m(t) + \dot{V}_m(t) \geq V_{th} \; and \; t - t_{last\_emit} \geq t_{refactor} \\ R_m I(t), & otherwise \end{cases} && (2.3.59)\\
\dot{t}_{last\_emit}(t) &= \begin{cases} t - t_{last\_emit}, & V_m(t) + \dot{V}_m(t) \geq V_{th} \; and \; t - t_{last\_emit} \geq t_{refactor} \\ 0, & otherwise \end{cases} && (2.3.60)
\end{aligned}$$

2.3.9 Adaptive Integrate-and-fire (AdEx)

$$\begin{aligned}
x(t) &= \begin{bmatrix} V_m(t) & w(t) \end{bmatrix} && (2.3.61)\\
\dot{x}(t) &= \begin{bmatrix} \dot{V}_m(t) & \dot{w}(t) \end{bmatrix} && (2.3.62)\\
\dot{V}_m(t) &= -\frac{1}{C}(g_L(V_m(t) - E_L) + g_L \Delta_T exp(\frac{V_m(t) - V_T}{\Delta_T}) - w(t) + I(t)) && (2.3.63)\\
\dot{w}(t) &= \frac{1}{\tau_w}(a(V_m(t) - E_L) - w(t)) && (2.3.64)
\end{aligned}$$

2.3.10 Fring Rate Model

$$x(t) = \left[r(t)\right] \tag{2.3.65}$$
$$\dot{x}(t) = \left[\dot{r}(t)\right] \tag{2.3.66}$$
$$\tag{2.3.67}$$

The firing rate model is a model based on spike emission phenomena. A firing vatiation model $r(t)$ is utilized to describe the firing rate on time. A drawing function for generating spike with possibility $p(emit\_spike|t) = r(t)\Delta t$ can be utilize to generate a spike train that meet the firing rate $r(t)$.

2.3.11 Discussion

Spike Representation You may note that, although different neuron models have different dynamics and spike representations, they can still communicate with each other through synapses. Some neuron models require current input, while others do not. Nonetheless, spikes can transmit over synapses and cause current variations across them by utilizing a common output element V_{out}.
One challenge that may arise is the normalization of spike representations. To address this issue, one approach is to standardize the spike events by converting all spike representations into binary form. Another approach can involve adjusting the synaptic weights. Although we may face scaling issues with different types of spike representations, these can be mitigated during training by appropriately adjusting the synapse weights.

3. Synapse Dynamics

3.1 Biological Synapse

3.2 Overview of Synapse Models and Taxonomy

Similar to neuron dynamics, the synapse dynamics could be modeled utilize a state space model, and the current term $I(t)$ is included in the state $\mathbf{x}(t)$. We only need modified the constraint term, as shown in Equation 3.2.1 to Equation 3.2.3.

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{S}(t)) \tag{3.2.1}$$

$$\mathbf{y}(t) = g(\mathbf{x}(t)) \tag{3.2.2}$$

$$s.t.,\ (\exists i \in [0, |\mathbf{x}(t)|])(\mathbf{x}_t)_i = I(t) \tag{3.2.3}$$

$$\tag{3.2.4}$$

The synapse model can be categorized to current-based synapses and conductance-based synapses. The model that directly model the synapse current $I(t)$, it is also known as **current-based synpase model**. On the other hand, when we model the **synaptic kinetics** $s_{syn}(t)$ of a synapse and retrieve the synpase current by $I(t) = g_{syn}s_{syn}(t)(V_{syn} - V_m(t))$, the synpase model is also known as **conductance-based synpase**.
Both of current and synaptic kinetics can be represented as a convolution operations on the generalized weighted preneuron spike train. Specifically, we may write $z(t) = (\theta \cdot S(t)) * h(t)$, where $o \triangleq \{I, s_{syn}\}$, , $h(t)$ is a convolution kernel, and $S(t)$ is the pre-synapse neuron's spike train. $z(t)$ is the generalized output, which is the current I when it is a current-based synapse model, while s_{syn} when it is a conductance-based synapse model.

3.2.1 Current-based Synapse

3.2.2 Conductance-based Synapse

3.2.3 Chemical Synapse

Current-based Synapse

Conductance-based Synapse

3.3 Discussion

4. Trainning Algorithms

4.1 Taxonomy of Training Algorithms

From the form of data utilized in the training, we could categories training algorithms of spike neural network into largely two categories: unsupervior learning and supervise learning. While unsupervior learning not explictly specify the expect outcomes in the dataset and align the model to these explicit-specified outcomes , supervior learning explictly specify the expect outcomes in the dataset.

From the theory basis, in this book, we categories the training algorithms into 4 categories:

- **surrogate gradient descending methods**: The basic idea of surrogate gradient descending methods are utilize surrogate methods to solve the undifferntial term $\frac{\partial S(t)}{\partial w_{ij}}$ appear in the gradient descending-based loss optimizations. $S(t)$ is a spike train of a neuron, and w_{ij} is the synapse weight from neuron j to neuron i.
- **bio-phenomena-based methods**: The bio-phenomena-based inspired by bio-phenomena to adjust the weights dynamically. One kind of these methods is the spike-timing-dependent plasticity (STDP)-based methods, which inspired by the *long-term potentiation (LTP)* and the *long-term depression (LTD)* in the bioneural networks. Base on the difference in time of the the spike emit from pre-post neurons, STDP-based methods adjust the synapse weights.
- **statistical mechanism-based methods**: Statistical mechanism-based methods tackle the whole networks as a statistical mechanism system. Statistical mechanism methods are used to model the property (e.g., system energy), and the interaction between neurons. With optimize on the system energy, we adjust the weights of synapses.
- **observation methods**: rather than well-train the neural network, observation methods focus on design and training observation models on the neural network. As the system property of neural network at a certain time t, contains rich information about the dynamics of input data, by well-design an observation method, we has the potential to obtain the outcomes we expect.

4.2 Unsupervior Learning

4.2.1 Spike-timing-dependent plasticity (STDP)

4.2.2 Growing Spiking Neural Networks

4.2.3 Artola, Bröcher, Singer (ABS) rule

4.2.4 Bienenstock, Cooper, Munro (BCM) rule

4.2.5 Relationship between BCM and STDP rules

4.3 Supervised Learning

4.3.1 STDP-based Methods

Supervised STDP (SSTDP)

Spike-Timing-Dependent Plasticity (STDP) with Supervision

4.3.2 Spike-Timing Dependent Backpropagation (STDBP)

4.3.3 Liquid State Machine (LSM) and Readout Training

4.3.4 SpikeProp

Extension (McKennoch et al., 2006; Booij and tat Nguyen, 2005; Shrestha and Song, 2015; de Montigny and Mâsse, 2016; Banerjee, 2016; Shrestha and Song, 2017).
spike timing based methods is that they cannot learn starting from a quiescent state of no spiking.
Bohte (2011)
Huh and Sejnowski (2017)

4.3.5 ReSuMe

Related Work (Sporea and Grüning, 2013) Pfister et al. (2006) Gardner et al. (2015) Fremaux et al. (2010)

4.3.6 SuperSpike

SuperSpike [**super-spike**] is a supervised learning algorithm dedicated to deterministic Leaky Intergrate-and-Fire neuron model. While the backpropagation algorithm used in traditional neural network cannot directly be used in the training of spiking neural network, the author provide a surrogate gradient-based method to tackling with the problems facing in solving the $S_i/\partial w_{ij}$, where S_i is the i-th neuron's spike train, and w_{ij} is the connection weight from neuron j to neuron i. Specifically, $S_i(t) = \sum_k \delta(t - t_k)$, where t_k is the k-th spike emission time, and $\delta(\cdot)$ is the dirac delta function.
Approaches approximate the partial derivative of the hidden unit output by $f(S_{pre}, g(V_{post}))$.
Let $\hat{S}_i$ be the target spike train of neuron i. The cost model for optimization that make $\hat{S}_i$ approach the real S_i hold the form: $L = \frac{1}{2}\int_{-\infty}^{t} ds[(\alpha * \hat{S}_i - \alpha * S_i)(s)]^2$.
α is a normalized smooth temporal convolution kernel. The original SuperSPike use *double exponential causal kernel*.

$$\partial L/\partial w_{ij} = -\int_{-\infty}^{t} ds[(\alpha * \hat{S}_i - \alpha * S_i)(s)](\alpha * \frac{\partial S_i}{\partial w_{ij}})(s)$$

.
Some existing methods for tackling the term $\frac{\partial S_i}{\partial w_{ij}}$: (1) making derivation directly to the membrance voltage, (2) introducing noisy which render the likelihood of $\langle S_i \rangle$ a smooth function of the membrance potential.
The superspike convert calculation of $\frac{\partial S_i}{\partial w_{ij}} \to \sigma'(U_i)\frac{\partial U_i}{\partial w_{ij}}$. In which U_i is the membrance voltage. Original superspike choose $\sigma(U)$ be the negative side of a fast sigmoid. This function is objective to increase steeply and peak at the spiking threshold. Other monotonic functions may also work.

For current-based LIF models the membrane potential $U_i(t)$ can be written in integral form as a spike response model (SRM0 (Gerstner et al., 2014)): $U_i(t) = \sum_j w_{ij}(\varepsilon * S_j(t)) + (\eta * S_i(t))$
In which, ε corresponds to the postsynaptic potential (PSP) shape, η captures spike dynamics and reset.
The existence of term $(\eta * S_i(t))$ make us difficult to perform derivation. Now, $U_i(t) \approx (\varepsilon * S_j(t))$. The gradient calculation for a weight become:

$$\frac{\partial w_{ij}}{\partial t} = r \int_{-\infty}^{t} dse_i(s)\alpha * (\sigma'(U_i(s))(\varepsilon * S_j)(s))$$

.
r is the learning rate, $e_i(s) \equiv \alpha * (\hat{S}_i - S_i)$, $\lambda_{ij} = \alpha * (\sigma'(U_i(s))(\varepsilon * S_j)(s))$ is the eligibility trace. The form above is also known as *non-vanishing surrogate gradient*.
The neuron model utilized by the SuperSpike is

$$\tau^{mem}\frac{dU_i}{dt} = (U^{rest} - U_i) + I_i^{syn}(t)$$

. And synapse evolution model is

$$\frac{d}{dt}I_i^{syn}(t) = -\frac{I_i^{syn}(t)}{\tau^{syn}} + \sum_{j\in pre} w_{ij}S_j(t)$$

.

4.3.7 SPAN (Mohemmed et al., 2012)

4.3.8 Remote Supervised Method (ReSuMe)

4.3.9 FreqProp

4.3.10 Local error-driven associative biologically realistic algorithm (LEABRA)

4.3.11 Supervised Hebbian Learning

4.4 Reinforcement Learning

4.4.1 Spiking Actor-Critic method

4.4.2 STDP-based Methods

4.5 Convert Trandictional ANN to SNN

5. Network Architecture Investigation

5.1 Liquid Neural Network

State Space Models (SSM)
Linear State Space Models
Structured State Space Model (S4) [lnn-s4]
Liquid Time Constant (LTC) Model [ltc]
Linear Liquid Time Constant Model
High-order Polynomial Projection Operator (HiPPO)
Normal Plus Low-Rank (NPLR)
Scaled Legendre Measure (LegS)
HiPPO Matrix
black-box Cauchy Kernel
Coupled Bilinear
Dynamics Causal Model
Liquid structure state space model [**lnn-4s**] introduce the liquid time constant into the structure state model. Specifically it hold the form in Equation 5.1.1.

$$\frac{d\mathbf{x}(t)}{dt} = -[\mathbf{A} + \mathbf{B} \otimes f(\mathbf{x}(t), \mathbf{u}(t), t, \theta)] \otimes \mathbf{x}(t) + \mathbf{B} \otimes f(\mathbf{x}(t), \mathbf{u}(t), t, \theta) \tag{5.1.1}$$

A linear version could be expressed by the form of Equation 5.1.2.

$$\dot{\mathbf{x}}(t) = [\mathbf{A} + \mathbb{I}_N \mathbf{B} u(t)]\mathbf{x}(t) + \mathbf{B} u(t),\ y(t) = \mathbf{C}\mathbf{x}(t) \tag{5.1.2}$$

The output y_k for input u_0 to u_k can be represented by $y = \bar{K} * u + \bar{K}_{liquid} * u_{correlation}$. In which the $*$ is the convolution operator. This equation divide the calculation of y_k to the calculation of the kernel $\bar{K}$, and the convolution between the kernal and input, and the convolution between the kernal and input correlation.

5.2 Feedforward Neural Network

5.3 Recurrent Neural Network

5.4 Synfire Chain

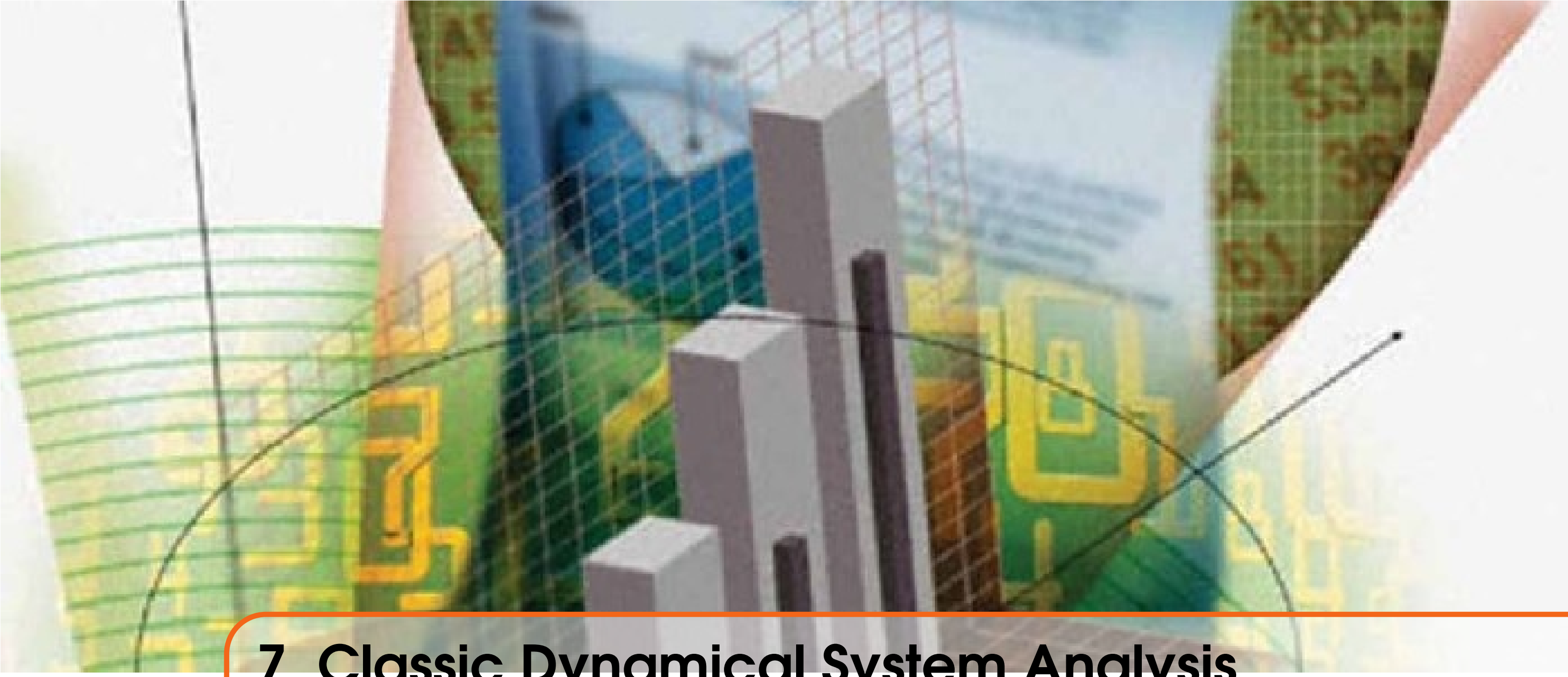
5.5 Reservoir Computing

5.5.1 Liquid State Machine

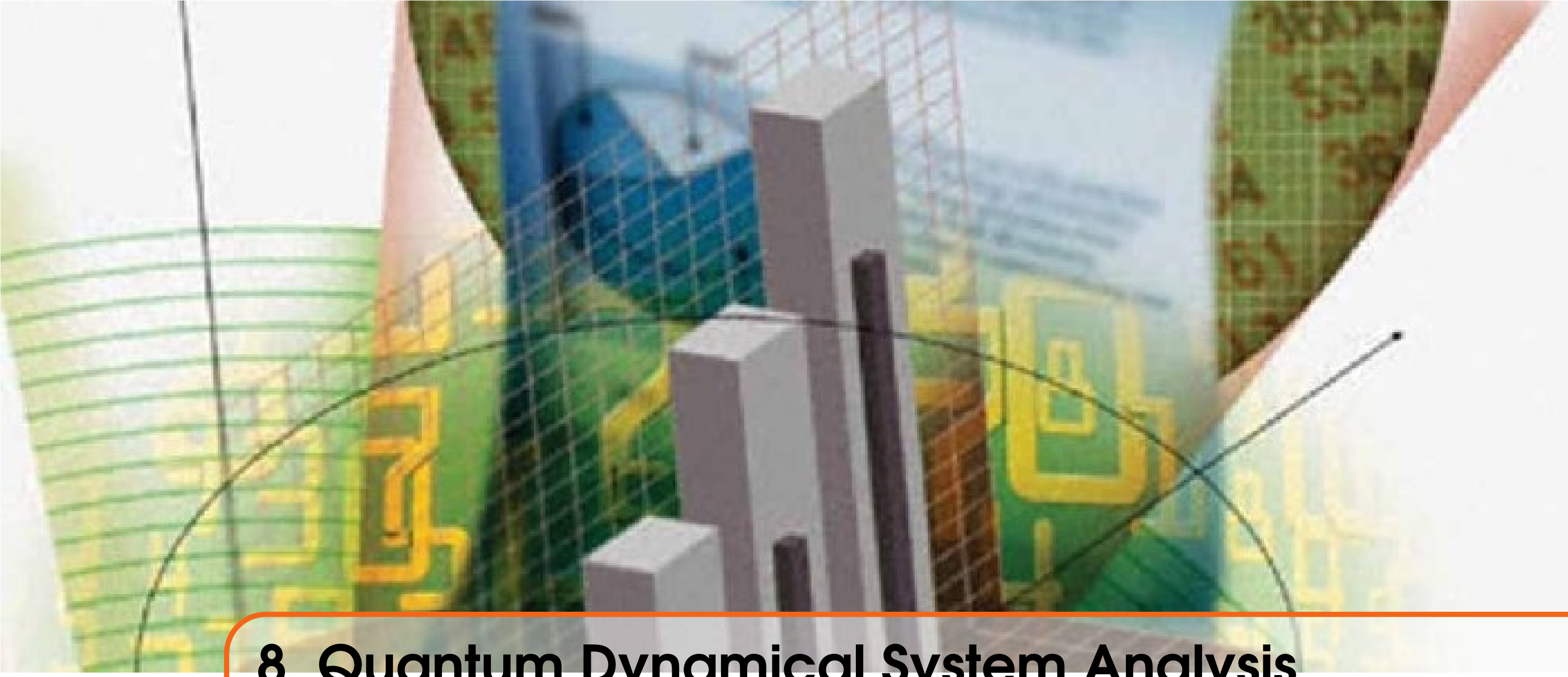
II

Part 2

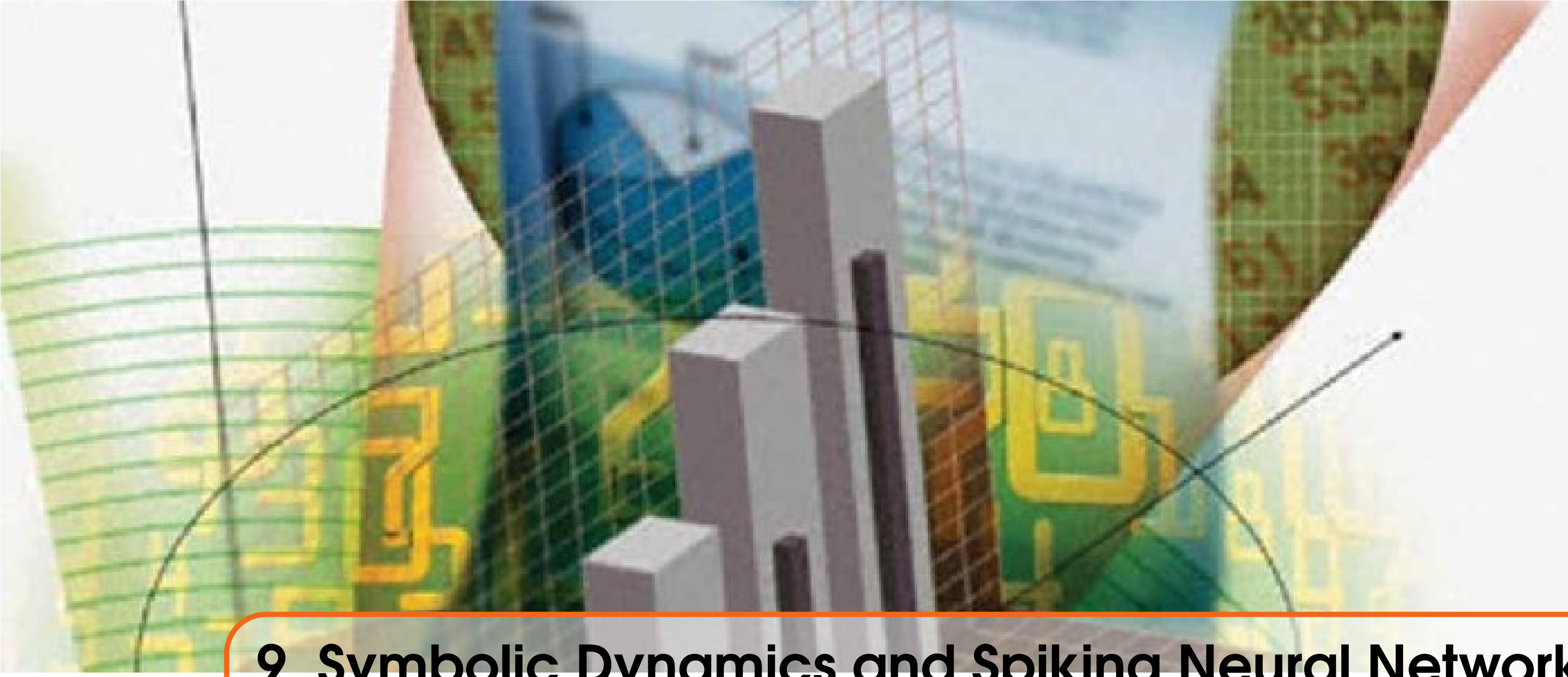
6. Dynamical System Basis



7. Classic Dynamical System Analysis

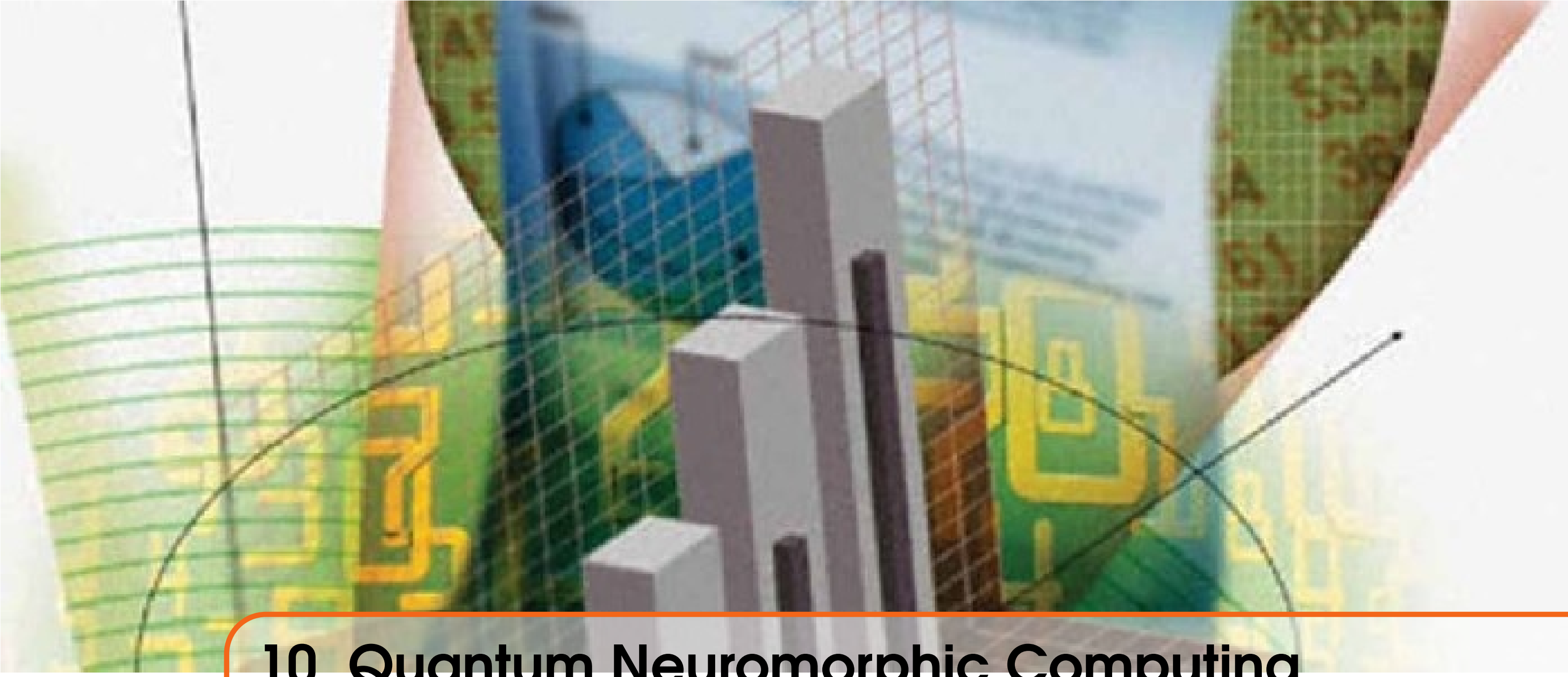


8. Quantum Dynamical System Analysis



9. Symbolic Dynamics and Spiking Neural Network

Part 3



10. Quantum Neuromorphic Computing

Simulation Method	**Philosophical Category**	**Characteristics**	**Example Application**
Element-Microscopic (Particle-Level)	Elementarism	Focuses on simulating fundamental components (e.g., particles, neurons) individually, often with reductionist assumptions.	Modeling individual qubits or synapses in Quantum SNNs using Schrödinger's equation.
System-Macroscopic (Whole-System)	Holism	Treats the system as a whole, capturing collective dynamics and emergent properties without decomposing into parts.	Analyzing the collective state of a Quantum SNN with wave functions.
Operational (Circuit/Annealing)	Operationalism	Focuses on practical, measurable implementations using existing technologies, like quantum circuits or annealing.	Implementing SNNs on quantum hardware such as Qiskit or D-Wave.
Field-Based Modeling	Field Theory	Treats phenomena as continuous fields rather than discrete entities; useful for spatial-temporal dynamics.	Modeling neural activity as a continuous field in the brain using PDEs.
Hybrid Quantum-Classical Simulations	Pragmatism	Combines quantum and classical techniques, prioritizing practical performance over theoretical purity.	Using hybrid quantum-classical algorithms for optimizing neural networks.
Topological Methods	Structuralism	Focuses on the structure of data; uses shapes and topological features to extract information from high-dimensional spaces.	Analyzing EEG microstates using persistent homology and topological features.
Dynamical Systems Modeling	Process Philosophy	Models systems based on continuous evolution over time, emphasizing processes and change.	Using differential equations to simulate spiking dynamics in neurons.
Generative Quantum SNNs	Generativism / Constructivism	Uses generative grammar and principles to model and generate/construct the evolution and dynamics of quantum spiking neural networks.	Applying generative models to simulate quantum states and guide the evolution of quantum SNNs.

Cuadro 10.1.1: *Philosophical Taxonomy of Simulation Methods for Quantum SNNs*

IV

Part 4

V

Part 5

Índice alfabético