Methodology of Recurrent Laguerre–Volterra Network for Modeling Nonlinear Dynamic Systems

Kunling Geng, *Student Member, IEEE*, and Vasilis Z. Marmarelis, *Fellow, IEEE*

***Abstract*— In this paper, we have introduced a general modeling approach for dynamic nonlinear systems that utilizes a variant of the simulated annealing algorithm for training the Laguerre–Volterra network (LVN) to overcome the local minima and convergence problems and employs a pruning technique to achieve sparse LVN representations with ℓ_1 regularization. We tested this new approach with computer simulated systems and extended it to autoregressive sparse LVN (ASLVN) model structures that are suitable for input–output modeling of nonlinear systems that exhibit transitions in dynamic states, such as the Hodgkin–Huxley (H-H) equations of neuronal firing. Application of the proposed ASLVN to the H-H equations yields a more parsimonious input–output model with improved predictive capability that is amenable to more insightful physiological/biological interpretation.**

***Index Terms*— Hodgkin–Huxley (H-H) equations, Laguerre–Volterra network (LVN), nonlinear system modeling, principal dynamic modes (PDMs), recurrent networks, simulated annealing (SA), Volterra modeling.**

I. Introduction

THE data-based nonparametric methods using the Volterra–Wiener formulation of nonlinear input–output models have been used over the last 40 years to study the dynamics of neural systems [5], [6], [21], [35], [36]. One key challenge for these methods has been the parsimonious model representation that allows computational efficiency, estimation accuracy, and physiological interpretation of the obtained models. A significant advance in this regard has been the practicable use of Laguerre expansions of the Volterra kernels [14]. Subsequent methodologies such as principal dynamic modes (PDMs) [23] and Laguerre–Volterra networks (LVNs) [24] are developed to achieve model parsimony, estimation efficiency, and model interpretation, as demonstrated in various physiological systems [15] and, more recently, in the study of cerebral hemodynamics and neuronal dynamics [6], [18]–[22].

In the Laguerre expansion technique (LET), we use the orthonormal basis of discrete Laguerre functions (DLFs) to compactly represent the Volterra kernels and achieve estimation efficiency. Although the LET significantly reduces the number of unknown parameters in the model, it does not remove the curse of dimensionality that comes from the incorporation of higher order kernels and multiple inputs. To overcome this limitation, the concept of PDMs was introduced aiming to find the minimum set of basis functions for adequate representation of the kernels of a given system [23]. Least-squares estimation methods have been used to obtain estimates of the expansion coefficients of the kernels in the LET. Singular value decomposition of first-order and second-order kernels has been used to obtain the PDMs by selecting the singular vectors that correspond to the significant singular values (usually $>10\%$ of maximum singular value). The PDMs obtained from this method are orthonormal, forming a functional coordinate basis for the system kernels. Using the obtained PDMs to represent/expand the kernels, the associated nonlinear functions (ANFs) are subsequently estimated via regression to obtain the Volterra-equivalent PDM-based model of the system [15]. This approach has been successfully used in various applications [6], [14]–[17], [51]–[53].

The LVN [24] combines the idea of Laguerre kernel expansions with the connectionist modeling notion of artificial neural network (ANN). It employs cost minimization training algorithms to obtain estimates of nonorthogonal PDMs of a given system in the context of arbitrary orders of nonlinearity (generally distinct from the original set of PDMs that are orthogonal and extracted from second-order models). The fact that the PDMs obtained from LVN are not necessarily orthogonal is viewed as providing more flexible representations of real physiological systems. Another advantage of LVN is the capability to train the critical Laguerre parameter α, which the original PDM method determines through a search procedure.

Although the LVN approach has found some useful applications [2], [25], [26], its efficacy has been hindered by the limitations of the employed backpropagation (BP) training algorithm, which as a gradient-descent based algorithm, suffers from local minima trapping and convergence problems [9]. Local minima trapping depends critically on the initialization points that are arbitrarily or randomly selected. The convergence problems relate to the learning rates for each LVN parameter. Several enhancements of BP have been introduced, such as the delta–delta or delta–bar–delta rules [12] and the momentum method [31], which mitigate these problems to some extent for many ANNs. However, they have not been

Manuscript received February 24, 2016; accepted June 9, 2016. This work was supported by the National Institutes of Health (NIH) under Grant P41-EB001978 through the Biomedical Simulations Resource (BMSR) at the University of Southern California.

The authors are with the Biomedical Simulations Resource Center, Department of Biomedical Engineering, University of Southern California, Los Angeles, CA 90089 USA (e-mail: kgeng@usc.edu; vzm@usc.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2016.2581141

always effective in the case of LVN–perhaps due to the nonmonotonic activation functions of the LVN (in contrast to the monotonic activation functions of ANN). We have found that these problems can be largely overcome by the use of the simulated annealing (SA) algorithm [4], [9], [13], which is adopted in this study as a stochastic global optimization method for training LVN.

Another challenge of the LVN method is the selection of the proper structural model parameters, viz., the number of DLFs L, the number of hidden units H, and the degree of polynomial nonlinearity Q. Different structures of LVN may have similar predictive power, but the corresponding PDMs in each structure (defined by the parameters of the hidden units) may be significantly different, which would complicate the interpretation of the obtained model. One common method for determining the model order is to search through all possible combinations of the structural parameters (Q, L, H) and use the Bayesian information criterion (BIC) [32] to select the model order with the lowest BIC value. This approach requires an evaluation of $Q_m L_m(L_m + 1)/2$ models, where Q_m and L_m are the maximum values of Q and L, which results in heavy computational burden, since SA training requires considerable computational effort. We propose a more efficient approach to determine the model order by introducing the concept of sparse LVN, which will force many elements of an initially high-order LVN to zero and achieve a compact model by successive pruning of the LVN until convergence.

To illustrate the efficacy of the SA training algorithm and the model pruning approach, we present first the results from simulated data and, subsequently, propose a novel autoregressive sparse LVN (ASLVN) to model the extensively studied Hodgkin–Huxley (H-H) equations of neuronal firing [11]. H-H equations are crucial for understanding the underlying dynamic processes for the generation of an action potential (AP) in computational neuroscience. However, the kinetics-based H-H model does not capture the variabilities within biological systems [39], [40]. The recent development of a stochastic H-H model is trying to capture some of these variabilities by introducing noise terms in the deterministic H-H equations [41]–[46]. On the other hand, the data-based nonparametric methods, such as nonlinear autoregressive Volterra (NARV) modeling [5], conventional PDM analysis [6], and recurrent ANN [47], have been utilized to approximate the AP generation process. These nonparametric methods, however, cannot provide biological insights through interpretation of the obtained models. The proposed ASLVN method in this paper yields a parsimonious input–output representation of the H-H equations with improved predictive capability and more insightful interpretations. Some of the preliminary results were previously presented in [38].

This paper is organized as follows. Section II describes the SA training algorithm for LVN. Section III introduces the concept of sparse LVN and model order pruning technique for LVN. In Section IV, we use simulated data to evaluate the performance of the SA training algorithm and the model order pruning technique. In Section V, we apply ASLVN for modeling the H-H equations and obtain some insightful interpretations. Finally, conclusions are drawn in Section VI. The Appendix gives the mathematical underpinnings of LVN and the details of the traditional BP training method.

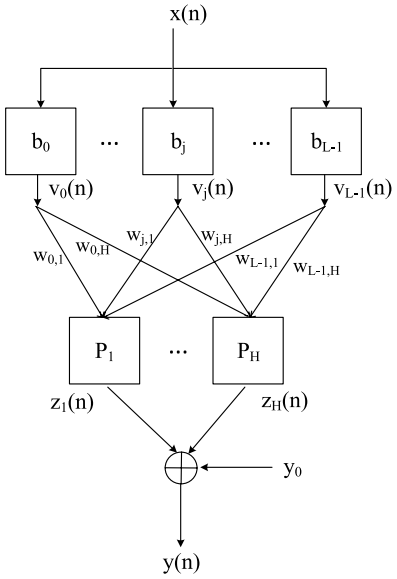


Fig. 1. Architecture of LVN [24].

II. LVN Training With Simulated Annealing

The structure of the LVN is shown in Fig. 1. Its fundamental properties and BP training method are summarized in the Appendix. As mentioned in Section I, the BP algorithm is hindered by local minima limitations and convergence problems. For this reason, we propose using the SA approach for training the LVN. For each step in SA, a random move to a neighboring state is performed and if the cost function decreases, then the new state is accepted. Otherwise, the move will be accepted with the probability that is determined by the Metropolis sampling algorithm (see below). Here is the basic description of how SA works to find the best parameter set $\mathbf{p}$ of LVN, which minimizes the cost function J.

1) Initialize the parameter set $\mathbf{p}$ randomly and set the temperature T to a sufficiently large value T_0.
2) Perform a small random move $\Delta\mathbf{p}$ based on the current parameter set so that the new one is $\mathbf{p}' = \mathbf{p} + \Delta\mathbf{p}$.
3) Propagate the input through the LVN using current and new parameters sets and compute the cost functions $J(\mathbf{p})$ and $J(\mathbf{p}')$, respectively, to get the difference: $\Delta J = J(\mathbf{p}) - J(\mathbf{p}')$.
4) Calculate the acceptance probability

$$r = \begin{cases} 1 & \text{if } \Delta J < 0 \\ e^{-\Delta J/T} & \text{otherwise} \end{cases} \tag{1}$$

   i.e., if $\Delta J < 0$, we accept the new parameter set $\mathbf{p}'$; otherwise, $\mathbf{p}'$ is accepted with the probability defined above.
5) Repeat steps 2)–4) until it reaches an equilibrium, which may close to a local minimum. In practice, we often choose a large but fixed number of iterations.

6) Decrease the temperature T according to the cooling schedule and repeat steps 2)–5) again, until T reaches the frozen point (i.e., a small positive value).

Initially, the temperature T should be sufficiently high and $e^{-\Delta J/T} \approx 1$, so the new state will almost always be accepted. Thus, in the beginning, SA allows random steps that explore the parameter space without fear of getting trapped in a local minimum. In order to ease the selection of the initial temperature, we use as cost function the normalized mean squared error (NMSE) rather than the sum of squared errors (SSE), since the SSE varies a lot for different data lengths. As the temperature decreases, the term $e^{-\Delta J/T}$ will be dominated more and more by ΔJ and SA will take new states more cautiously. When T reaches the frozen point, it allows only steps decreasing J that eventually lead to the global minimum.

The effectiveness and speed of the SA method are largely dependent on the cooling schedule [7]. A practical method is the exponential decay cooling schedule [29]

$$T(i) = \eta T(i-1), \quad i = 1, 2, 3 \ldots \tag{2}$$

where η is the cooling constant between 0.9 and 0.99. This schedule achieves only quasi-optimal results, but it significantly reduces the time required by the SA algorithm.

To apply the SA algorithm to LVN, another practical issue is how to perform a small random move in step 2) to get a new configuration of the parameter set. To achieve this, a single parameter p is randomly chosen from the whole set following a uniform distribution, so that each time every parameter would have the same probability to be selected. Then, p is updated by the following rule:

$$p^{\text{new}} = \begin{cases} p + \gamma_p & \text{if } rn > 0.5 \\ p - \gamma_p & \text{if } rn \leqslant 0.5 \end{cases} \tag{3}$$

where the constant γ_p is the step size, and rn is a random number drawn from a uniform distribution with range [0, 1]. In this way, this parameter p has an equal chance to increase or decrease by a small amount the value of γ_p.

When the DLF critical parameter α is updated, the outputs of the DLF filter bank $v_j(n)$ need to be recomputed (see (A3) and (A4) in the Appendix). Since each time α is updated by decreasing or increasing by a step size γ_α and α is within the range (0, 1), there are only $1/\gamma_\alpha - 1$ possible values. Therefore, we can calculate all the possible $v_j(n)$ for each α and store them into a look-up table in order to reduce the overall computational burden.

III. Sparse LVN and Model-Order Pruning Algorithm

Define the cost function of LVN as

$$J = \text{NMSE} + \lambda(||w||_0 + ||c||_0) \tag{4}$$

where $||\cdot||_0$ is the ℓ_0 norm, which is the number of nonzero elements. By adding the ℓ_0 regularization term, the optimization results would prefer zero elements, which would reduce the cost function. Therefore, the trained w and c would contain many zeros and thus the LVN structure should be sparse. This property allows pruning of LVN by removing the unnecessary elements and determining the appropriate model structure. The nonlinear order Q corresponds to the highest degree term in the polynomial activation functions (PAFs) that are associated with each PDM in each hidden node.

We can use SA to optimize the cost function in (4), but the results are often unsatisfactory because the ℓ_0 norm makes the cost function highly nonsmooth and very hard to optimize. To relieve this problem, we are using ℓ_1 regularization, which is obtained by relaxing the hard constraints of ℓ_0 regularization [10], [28], and then the cost function becomes

$$J = \text{NMSE} + \lambda(||w||_1 + ||c||_1) \tag{5}$$

where $||\cdot||_1$ is the summation of the absolute values of the parameter. With the ℓ_1 regularization term, the cost function is much easier to optimize by SA and the trained results also have the sparse properties. It should be mentioned that traditionally in ANN BP training methods, the ℓ_2 term, $||w||_2 = \sum w^2$, is added to the cost function in order to prevent overfitting, which is known as the weight decay method [27]. However, the ℓ_2 norm usually cannot generate sparse results in LVN. The ℓ_1 regularization has been recently used in Volterra-based models for neural functional connectivities [33], [34].

Here is a description of the model order pruning algorithm utilizing the sparse LVN.

1) Start with a full model order [e.g., $(Q, L, H) = (3, 9, 9)$ in practice] or a user-specified high model order.
2) Optimize the cost function defined in (5) with the SA algorithm under the current model order.
3) Prune unnecessary elements as many as possible by proposing a new model order.
4) Repeat steps 2) and 3) until convergence.

More elaboration is needed in step 3. First, we have to prune H because unnecessary hidden nodes would be inactivated by sparse LVN and the output of these hidden nodes would be around zero. One method to determine the significance of hidden node is to calculate $\text{RMS}_h = (\sum_n z_h^2(n))^{1/2} / \sum_{h=1}^{H} (\sum_n z_h^2(n))^{1/2}$, where RMS_h is the normalized root mean square of the hidden node output $z_h(n)$. In practice, when $\text{RMS}_h < 0.1$, the corresponding hidden node can be pruned. Then we can inspect the weights of the PDMs and reduce L as much as possible by removing weights that are below a chosen threshold (in this study <0.05). The degree Q of the PAFs can be decreased if it can achieve an adequate low-degree approximation of the system nonlinearity within the operating range of the PAF. An illustrative example using the model order pruning algorithm is shown in Section IV.

IV. Method Evaluation Using Simulated Data

The performance of the training algorithm and the model order pruning technique is evaluated in this section through computer simulations where ground truth is available. The simulated system has the two PDMs shown in Fig. 2 and is described by the following equations:

$$p_1(m) = e^{-m/20} \sin(\pi m/12) \tag{6}$$

$$p_2(m) = e^{-(m-20)^2/180}. \tag{7}$$

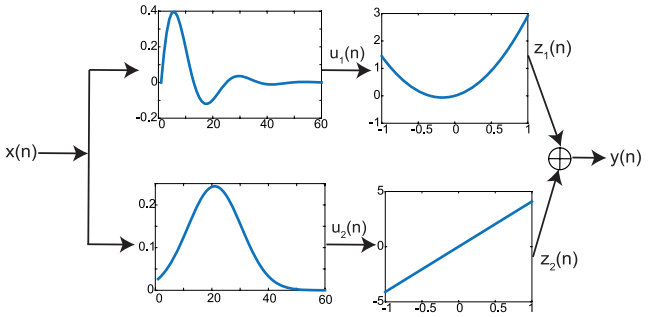


Fig. 2. Simulated system contains two distinct PDMs and the associated PAFs (second-order polynomial function for PDM 1 and linear function for PDM 2).

In this second-order simulated example, the PAFs associated with each PDM are given by the following polynomials (i.e., quadratic and linear):

$$z_1(n) = 0.5u_1(n) + u_1(n)^2 \tag{8}$$

$$z_2(n) = u_2(n). \tag{9}$$

The PDMs are normalized as $\sum_{m=1}^{M} p_i^2(m) = 1, i = 1, 2$, and the coefficients of the associated PAFs are adjusted so that the overall model prediction remains the same. The input of the simulated system is Gaussian white noise (GWN) with unit variance and the data length is $N = 1000$ samples. In order to test the robustness of the algorithm, an independent GWN is added to the output as noise for a signal-to-noise ratio (SNR) of 5 dB. Using the simulated input–output data, we train the LVN with the SA algorithm and apply the model order pruning algorithm to determine the structural parameters (Q, L, H). We also compare the results with the traditional BIC model order selection method.

First, we use SA to train the LVN without regularization ($\lambda = 0$ in the cost function J) for all possible model orders. The SA-related parameters are given in Table I. Then we use the BIC criterion to select the model order with the lowest BIC value, which is defined as

$$\text{BIC} = -2 \ln \hat{L} + P \ln N \tag{10}$$

where $\hat{L}$ is the maximum likelihood function estimate based on a sample size N and $P = (L + Q)H$. Since the noise is additive GWN, the BIC can be written as

$$\text{BIC} = N \ln \text{SSE} + P \ln N \tag{11}$$

The BIC values for all 135 models evaluated in this step are summarized in Fig. 3. As we can see, the number of parameters is less for a linear model than nonlinear models, but the prediction error is greater and the BIC values are significantly higher. For nonlinear models with $Q = 2$ and $Q = 3$, the BIC values are similar for different L and H, and the best BIC values are the same. Since we favor model parsimony, the selected model order by BIC is (Q, L, H) = (2, 7, 2). This is an appropriate model order because the simulated system has two PDMs and the highest degree of nonlinearity is 2. Evidently, seven DLFs are able to represent the PDM waveforms defined in (6) and (7). For this model order, the training results of LVN are summarized

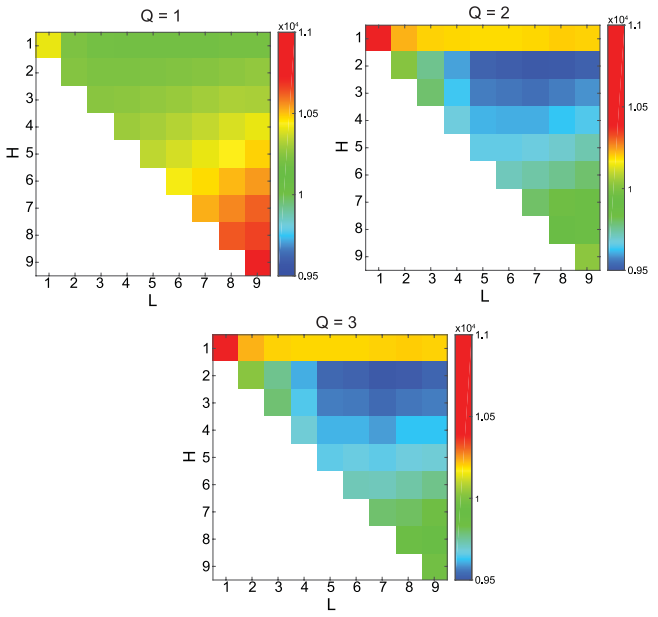


Fig. 3. BIC values for LVN for all model orders ($Q_{\max} = 3$ and $L_{\max} = 9$). The lowest BIC values are 1.018×10^4, 0.9496×10^4, 0.9496×10^4, and the corresponding model orders are (Q, L, H) = (1, 6, 1), (2, 7, 2), (3, 7, 2) for $Q = 1$, 2, 3, respectively.

TABLE I
SA PARAMETERS

Symbol	Name	Value
T_0	Initial temperature	100
η	Cooling constant	0.99
N_t	No. of temperature droppings	2000
N_i	No. of iterations at each temperature	200
$\gamma_\alpha, \gamma_w, \gamma_c, \gamma_{y_0}$	Step sizes	0.01

in Fig. 4, where NMSE = 0.216, $\alpha = 0.590$, and $y_0 = -0.0665$. The obtained PDMs and the associated PAFs are very close to their exact counterparts of the simulated system, indicating the efficacy of the SA training algorithm.

Then, we use the pruning technique of sparse LVN to determine the model order for the cost function defined in (5) and the SA parameters given in Table I. Here we set $\lambda = 0.01$ and a detailed description of the effects of λ will be discussed later. In the first step, we start with the full model order (Q, L, H) = (3, 9, 9), and the training results are shown in Fig. 5(a) for the two significant PDMs, since all other PAF coefficients are almost zero indicating that the corresponding hidden nodes do not contribute to the LVN output. Therefore, we can reduce the number of hidden nodes from 9 to 2. Another observation is that one of the PAFs is a straight line, showing linear relation, while the other PAF is parabolic, showing second-order nonlinearity, as in the simulated system. Therefore, we can reduce Q from 3 to 2. If we look into the PDM weights, we see that we can reduce the number of DLFs from 9 to 8 based on the rules from Section III. In the second step, we retrain the sparse LVN again but with a pruned

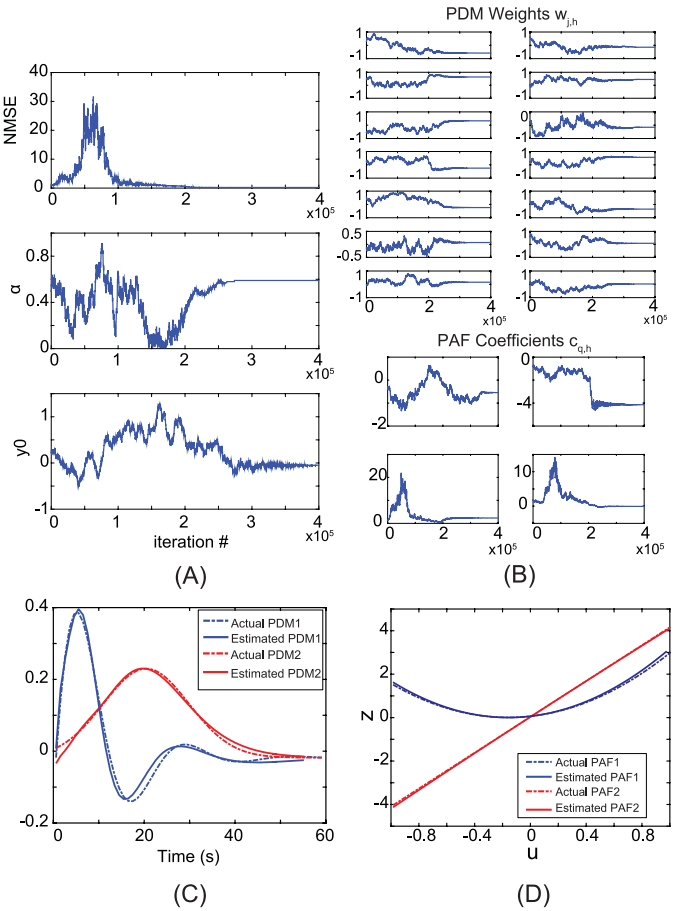


Fig. 4. After 400 000 iterations of the SA algorithm. (a) Tracking curves of NMSE, α, and y_0. (b) Tracking curves of the LVN weights and PAF coefficients. (c) Actual and estimated PDMs. (d) Actual and estimated PAFs. The model order is $(Q,\ L,\ H) = (2,\ 7,\ 2)$, which is determined by the BIC.

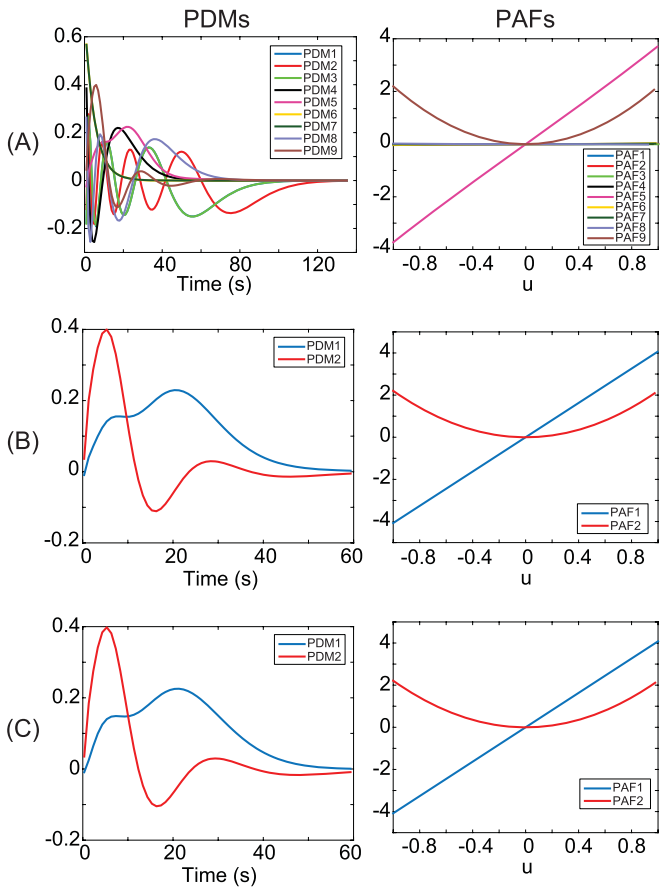


Fig. 5. PDMs and PAFs estimated in each step of the model order pruning technique. (a) Step 1 with full model order $(Q,\ L,\ H) = (3,\ 9,\ 9)$. (b) Step 2 with pruned model order $(Q,\ L,\ H) = (2,\ 8,\ 2)$. (c) Step 3 with final selected model order $(Q,\ L,\ H) = (2,\ 7,\ 2)$.

model order $(Q,\ L,\ H) = (2,\ 8,\ 2)$, and the training results are shown in Fig. 5(b). This time, we cannot change H or Q, since there are no redundant PDMs and the PAFs still show second-order nonlinearity. However, inspection of the weights shows that L can be further reduced to 7. The next step is to examine the model order $(Q,\ L,\ H) = (2,\ 7,\ 2)$, and the results are shown in Fig. 5(c). This time we cannot prune the LVN anymore, because there are no redundant elements. Thus, $(Q,\ L,\ H) = (2,\ 7,\ 2)$ is the final selected model order.

The PDMs and PAFs estimated by sparse LVN are very close to their actual counterparts of the simulated system, although for a lag of 0–10 s in PDM 1, we can see some deviations. The reason is that with ℓ_1 regularization, the estimates are biased. Therefore, once we have determined the model order with sparse LVN, we should reestimate the LVN elements without regularization terms.

In the previous training case, we set the regularization parameter $\lambda = 0.01$. It is very important to determine a proper value of λ, because if λ is too small, then the regularization effect is too limited to yield sparse results. On the other hand, if λ is too large, then the training results would be strongly biased by the regularization term. Therefore, how the regularization parameter λ affects the training results must be assessed carefully.

TABLE II

MODEL ORDER PRUNING RESULTS FOR DIFFERENT REGULARIZATION STRENGTHS

	λ=0.0001	λ=0.001	λ=0.005	λ=0.01	λ=0.02	λ=0.05
Step 1	(3, 9, 9)	(3, 9, 9)	(3, 9, 9)	(3, 9, 9)	(3, 9, 9)	(3, 9, 9)
Step 2	(3, 9, 3)	(2, 8, 4)	(2, 8, 2)	(2, 8, 2)	(2, 7, 2)	(2, 6, 2)
Step 3	(3, 8, 2)	(2, 8, 2)	\	(2, 7, 2)	\	(2, 5, 2)

Note that for λ= 0.05 and 0.02, the model order pruning technique stops at step 2.

As an example, we use the same system shown in Fig. 2 with noisy data (SNR = 5, additive GWN) and the same pruning technique to examine how the training results vary for different λ values. The pruned model orders in each step are summarized in Table II and the estimated PDMs and PAFs corresponding to the final determined model order are shown in Fig. 6. As we can see, when $\lambda = 0.0001$, the selected model order is $(Q,\ L,\ H) = (2,\ 8,\ 3)$, which has the wrong number of PDMs and the estimated PDMs are very different from the actual ones. Therefore, when the regularization strength is low, it cannot yield sparse results and the correct model order.

When we increase λ from 0.001 to 0.02, we can see that the pruning algorithm works well and although the selected L varies a little, the overall model orders are very close to the best one. In addition, the estimated PDMs and PAFs

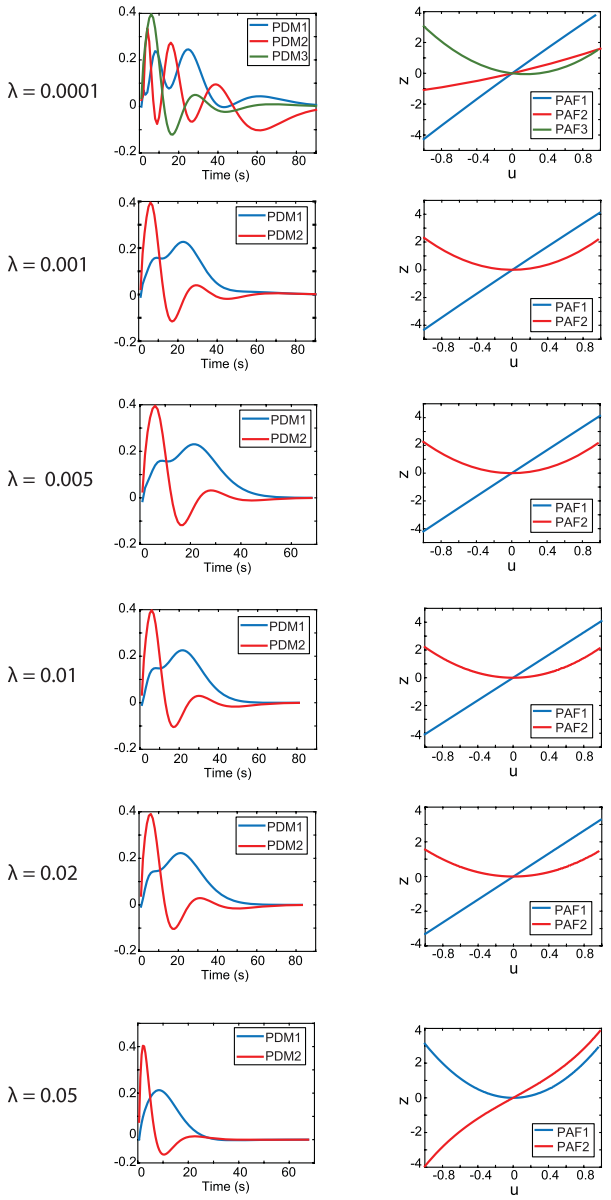


Fig. 6. PDMs and PAFs that are estimated by sparse LVN under different regularization strengths with $\lambda = 0.0001$, 0.001, 0.005, 0.01, 0.02, and 0.05.

are very consistent and close to their counterparts in the simulated system. When $\lambda = 0.05$, the determined model order is $(Q, L, H) = (2, 5, 5)$ and the estimated PDMs and ANFs are not accurate. The result is expected, because as the regularization strength increases, the selected model order decreases, but the estimated PDMs and ANFs are highly biased.

From the above results, we can conclude that although the regularization strength generally affects the selected model order of sparse LVN, the effect is relatively small over a wide range of λ values (from 0.001 to 0.02). Therefore, in practice, we can set λ to 0.01 or 0.02. In the following, λ is set to 0.01 unless otherwise explicitly stated.

V. Application to the Hodgkin–Huxley Equations

The H-H equations are a classic conductance-based model describing the underlying dynamic processes for the generation of an AP by a neuron [11]. The main H-H equation is

$$I = C_m \frac{dV}{dt} + \bar{g}_K n^4 (V - E_K) + \bar{g}_{\mathrm{Na}} m^3 h (V - E_{\mathrm{Na}}) + \bar{g}_L (V - E_L) \tag{12}$$

where I is the injected current, V is the membrane potential, and C_m is the membrane capacitance. The maximum potassium and sodium conductances are represented by $\bar{g}_K$ and $\bar{g}_{\mathrm{Na}}$, and $\bar{g}_L$ is the conductance of the leakage ion channel. E_K, E_{Na}, and E_L are the reverse potentials for each channel. We use the standard H-H parameters from the giant axon of a squid, which are summarized in Table III, and the resting potential is set at 0 mV. The gating variables m, n, and h are responsible for the voltage-dependent activation and deactivation of the channels, which are expressed in the first-order differential equations

$$\begin{aligned} \frac{dn}{dt} &= \alpha_n (1 - n) - \beta_n n \\ \frac{dm}{dt} &= \alpha_m (1 - m) - \beta_m m \\ \frac{dh}{dt} &= \alpha_h (1 - h) - \beta_h h \end{aligned} \tag{13}$$

where

$$\begin{aligned} \alpha_n(V) &= \frac{-0.01(V - 10)}{e^{-(V-10)/10} - 1}, \quad \beta_n(V) = 0.125 e^{-V/80} \\ \alpha_m(V) &= \frac{-0.1(V - 25)}{e^{-(V-25)/10} - 1}, \quad \beta_m(V) = 4 e^{-V/18} \\ \alpha_h(V) &= 0.07 e^{-V/20}, \quad \beta_h(V) = \frac{1}{e^{-(V-30)/10} + 1}. \end{aligned} \tag{14}$$

To obtain an equivalent I–V relationship of the H-H equations, we propose a novel ASLVN, which is shown in Fig. 7. The ASLVN is derived from traditional two-input single-output LVN. The exogenous input of ASLVN is $x_1(n) = I(n)$, which is the current injected in the H-H model, and the autoregressive input $x_2(n) = y(n-1)H(y(n-1) - \theta)$ is the thresholded output of the ASLVN with one lag delay, where $H(x)$ is the unit step function. The reason for introducing the threshold θ is to separate the two distinct dynamic states caused by bifurcation of the H-H model [8]. It should be mentioned that in the model estimation phase, the open-loop ASLVN and the membrane potential $V(n)$ from H-H equations

TABLE III
Parameters of H-H Equations

Symbol	Name	Value
C_M	Membrane conductance	1 uF/cm^2
E_K	Reverse potential of K^+ channel	-12 mV
E_{Na}	Reverse potential of Na^+ channel	115 mV
E_L	Reverse potential of leak channel	10.6 mV
$\bar{g}_{Na}$	Maximum conductance of Na^+ channel	120 mS/cm^2
$\bar{g}_K$	Maximum conductance of K^+ channel	36 mS/cm^2
$\bar{g}_L$	Conductance of leak channel	0.3 mS/cm^2

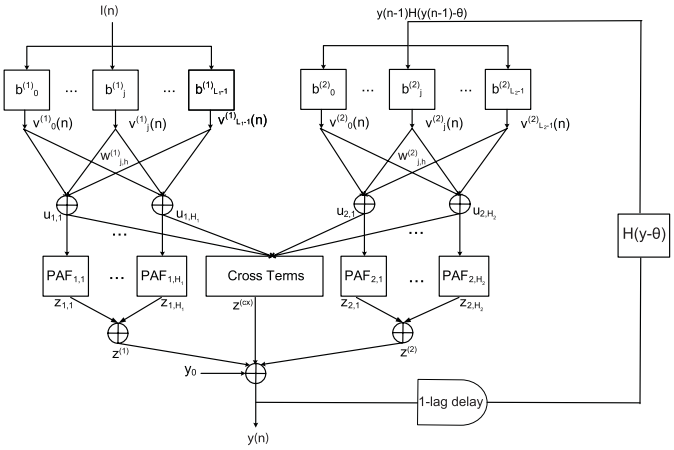


Fig. 7. Structure of ASLVN for modeling H-H equations. It consists of three parts: 1) the forward branch with input $I(n)$; 2) the autoregressive branch with input $y(n-1)H(y(n-1)-\theta)$; and 3) the cross terms representing the interactions between the forward and autoregressive branches.

are used to train ASLVN for the sake of stability. Thus, the autoregressive input is $V(n-1)H(V(n-1)-\theta)$ and the cost function is NMSE $= (\sum_n (V(n)-y(n))^2 / \sum_n V(n)^2)$. After the training, the ASLVN predicted output $y(n)$ should be used as the autoregressive input.

The outputs of the forward branch $z^{(1)}(n)$ and autoregressive branch $z^{(2)}(n)$ are

$$\begin{aligned} z^{(i)}(n) &= \sum_{h=1}^{H_i} z_h^{(i)}(n) \\ &= \sum_{h=1}^{H_i} \sum_{q=1}^{Q_i} c_{q,h}^{(i)} \left[T \sum_{m=0}^{M_i-1} p_h^{(i)}(m) x_i(n-m) \right]^q \end{aligned} \tag{15}$$

and the output of the cross term is

$$\begin{aligned} z^{(cx)}(n) &= \sum_{h_1=1}^{H_1} \sum_{h_2=1}^{H_2} \sum_{q_1+q_2=2}^{Q_{cx}} c_{q_1,q_2,h_1,h_2}^{(cx)} \left[u_{h_1}^{(1)}(n)\right]^{q_1} \left[u_{h_2}^{(2)}(n)\right]^{q_2} \\ &= \sum_{h_1=1}^{H_1} \sum_{h_2=1}^{H_2} \sum_{q_1+q_2=2}^{Q_{cx}} c_{q_1,q_2,h_1,h_2}^{(cx)} \\ &\quad \times \left[T \sum_{m_1=0}^{M_1-1} p_{h_1}^{(1)}(m_1) x_1(n-m_1) \right]^{q_1} \\ &\quad \times \left[T \sum_{m_2=0}^{M_2-1} p_{h_2}^{(2)}(m_2) x_2(n-m_2) \right]^{q_2}. \end{aligned} \tag{16}$$

We get the output of the ASLVN by adding all the outputs of above three parts

$$y(n) = y_0 + z^{(1)}(n) + z^{(2)}(n) + z^{(cx)}(n). \tag{17}$$

We can show that ASLVN is equivalent to two-input Volterra model. For the second-order system when $Q_1 = Q_2 = Q_{cx} = 2$

$$\begin{aligned} y(n) = k_{0,0} &+ T \sum_{m=0}^{M_1-1} k_{1,0}(m) x_1(n-m) \\ &+ T \sum_{m=0}^{M_2-1} k_{0,1}(m) x_2(n-m) \\ &+ T^2 \sum_{m_1=0}^{M_1-1} \sum_{m_2=0}^{M_1-1} k_{2,0}(m_1,m_2) x_1(n-m_1) x_1(n-m_2) \\ &+ T^2 \sum_{m_1=0}^{M_2-1} \sum_{m_2=0}^{M_2-1} k_{0,2}(m_1,m_2) x_2(n-m_1) x_2(n-m_2) \\ &+ T^2 \sum_{m_1=0}^{M_1-1} \sum_{m_2=0}^{M_2-1} k_{1,1}(m_1,m_2) x_1(n-m_1) x_2(n-m_2) \end{aligned} \tag{18}$$

where $k_{1,0}$ and $k_{0,1}$ are the first-order self-kernels, $k_{2,0}$ and $k_{0,2}$ are the second-order self-kernels for inputs 1 and 2, respectively, and $k_{1,1}$ is the cross kernel. The equivalent Volterra kernels of ASLVN can be expressed as

$$\begin{aligned} k_{0,0} &= y_0 \\ k_{1,0}(m) &= \sum_{h=1}^{H_1} c_{1,h}^{(1)} p_h^{(1)}(m) \\ k_{0,1}(m) &= \sum_{h=1}^{H_2} c_{1,h}^{(2)} p_h^{(2)}(m) \\ k_{2,0}(m_1,m_2) &= \sum_{h=1}^{H_1} c_{2,h}^{(1)} p_h^{(1)}(m_1) p_h^{(1)}(m_2) \\ k_{0,2}(m_1,m_2) &= \sum_{h=1}^{H_2} c_{2,h}^{(2)} p_h^{(2)}(m_1) p_h^{(2)}(m_2) \\ k_{1,1}(m_1,m_2) &= \sum_{h_1=1}^{H_1} \sum_{h_2=1}^{H_2} c_{h_1,h_2}^{(cx)} p_{h_1}^{(1)}(m_1) p_{h_2}^{(2)}(m_2). \end{aligned} \tag{19}$$

The training data of the H-H model are generated by the Euler forward method with the step size of 1 μs and the input (units in microamperes) is drawn from a Gaussian distribution $\mathcal{N}(0, 50)$. The current is injected at a frequency of 200 Hz and each injection lasts 0.5 ms. The total length of the training data is 10^3 ms. In order to evaluate the model prediction performance, we also generate a set of testing data with the length of 10^4 ms. A proper sampling interval $T = 0.25$ ms is carefully chosen based on the autocorrelation method [3] to avoid redundancy and irrelevance problems. Since the resting potential is 0 mV, we set the constant offset y_0 as 0. Although ASLVN has the ability to train the DLF critical parameter α in both forward and autoregressive branches, the values are fixed to 0.7 in this case. This is due to the fact that the iterative training procedures are drawn to the fast dynamics of the autoregressive component rather than their complete dynamics, resulting in very small α.

Using the SA training algorithm and the model order pruning technique, described in Sections II and III, the obtained

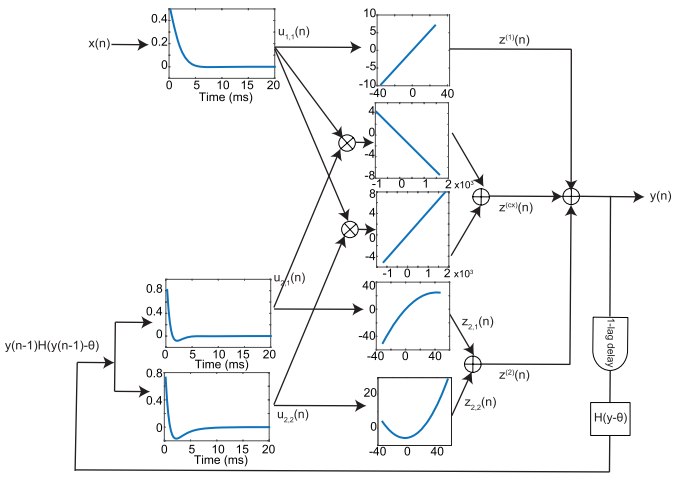


Fig. 8. Obtained ASLVN model using the SA training algorithm and the model order pruning technique. The model orders for the forward branch and the autoregressive branch are $(Q_1, L_1, H_1) = (1, 3, 1)$ and $(Q_2, L_2, H_2) = (2, 4, 2)$, respectively. The model contains only one forward PDM with a linear PAF, two autoregressive PDMs with second-order PAFs, and two bilinear cross terms.

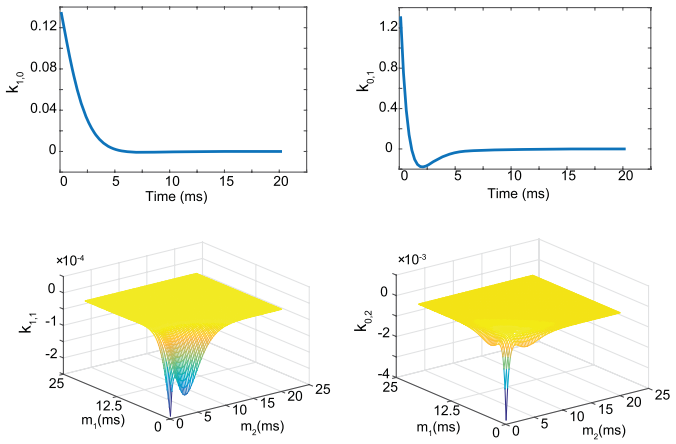


Fig. 9. Equivalent Volterra kernels constructed from the trained ASLVN model, where $k_{1,0}$ is the first-order forward kernel, $k_{0,1}$ and $k_{0,2}$ are the first- and second-order autoregressive kernels, and $k_{1,1}$ is the cross kernel.

ASLVN model is shown in Fig. 8, and the equivalent Volterra kernels of ASLVN are shown in Fig. 9. Our pruned ASLVN model has thee PDMs (one forward and two autoregressive) and a total of 18 parameters.

The output of each ASLVN component is illustrated in Fig. 10 and makes a distinct contribution to the model prediction. The forward PDM has the form of a leaky integrator and represents the subthreshold dynamics of the H-H model in Fig. 10(c). The autoregressive PDMs are responsible for the shape of the AP in Fig. 10(a), which consists of the depolarization and repolarization components, as well as the hyperpolarization after potential in Fig. 10(b). The two cross terms in the model help to maintain the refractory period by counteracting the effect of the exogenous input in a transient manner (absolute and relative refractory period). These three elements, including the capacitive leaky integrator, the AP, and the refractory period, constitute the essential functional

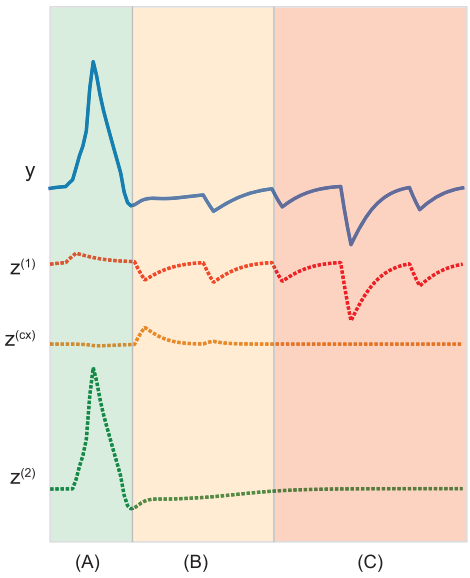


Fig. 10. Traces of total output y, forward branch output $z^{(1)}$, autoregressive branch output $z^{(2)}$, and cross-term output $z^{(cx)}$ in (a) spike (AP) generation phase, (b) refractory phase, and (c) subthreshold phase.

features described by the H-H model and should be present in equivalent models that try to emulate the H-H dynamics [1].

The threshold θ is chosen to be 4.5 mV in this paper. It should be noted that this θ separates the subthreshold from the suprathreshold nonlinear dynamics, which is different from the traditional hard threshold defined in the integrate-and-fire type of models [48]–[50]. The effect of different values of θ on the estimated PDMs are insignificant within a certain range (2–10 mV) and the model predictive performance is not affected. However, θ affects the cross-term coefficients that are estimated in the ASLVN, because the SA training process adapts these coefficients to accommodate the changes of θ. For example, when θ is small, the cross-term contribution would become more negative at the spike initiation phase to prevent overfiring due to the positive integration of the ASLVN forward branch. We can embed the optimization of θ into the training process, but the estimated θ values will not be the same in every SA optimization. Hence, we fix θ to guarantee consistency and chose 4.5 mV to allow the comparison with previous NARV and PDM models [5], [6]. Another advantage of fixing θ is that the computational burden is significantly reduced by avoiding the convolution computations at each iteration when θ is updated.

It is intriguing that the forward PDM in our pruned ASLVN system resembles the capacitive behavior in the leaky integrate-and-fire (LIF) model, which is expressed by

$$C_m \frac{dV}{dt} = \frac{1}{R}(E - V) + I. \tag{20}$$

We can compare the forward PDM with the LIF model in terms of the rate of exponential decay, which is determined by the time constant $\tau = RC_m$. It has been shown that the H-H model can be reduced to the LIF model by replacing the gating variables m, n, and h with their steady values $m_\infty(V), n_\infty(0), h_\infty(0)$ [1]. The reason for this approximation is that under subthreshold conditions, m reaches its steady

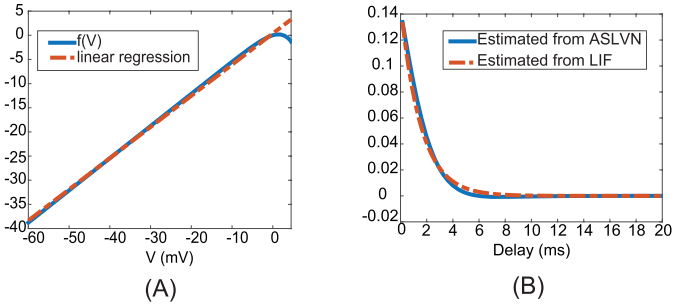


Fig. 11. (a) Linear approximation $f(V) = 0.65V + 0.37$ of the nonlinear characteristic function $f(V)$ under subthreshold conditions ($V \leq 4.5$ mV) and (b) comparison of subthreshold dynamics of the ASLVN and LIF models.

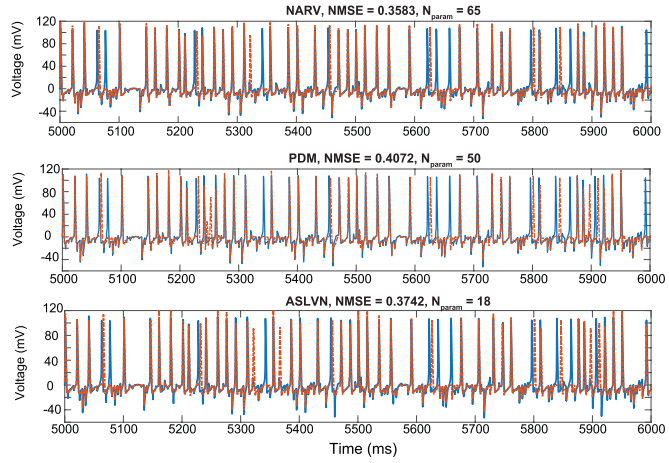


Fig. 12. Out-of-sample prediction results by NARV (top), PDM (middle), and ASLVN (bottom) models. For NARV, the model order is $Q_1 = Q_2 = 2$ and $L_1 = L_2 = 5$. For PDM, the model order is $(Q_1, L_1, H_1) = (2, 5, 2)$, $(Q_2, L_2, H_2) = (2, 5, 4)$. For ASLVN, the model order is $(Q_1, L_1, H_1) = (1, 3, 3)$, $(Q_2, L_2, H_2) = (2, 4, 2)$. The number of parameters N_{param} for these three methods is 65, 50, and 18, respectively.

value $m_\infty(V)$ very fast, while n and h change very slowly over time and do not deviate much from the values at the resting potential $n_\infty(0), h_\infty(0)$. Therefore, the main H-H equation can be rewritten as

$$I = C_m \frac{dV}{dt} + f(V) \tag{21}$$

where

$$f(V) = \bar{g}_K n_\infty^4(0)(V - E_K) + \bar{g}_{\text{Na}} m_\infty^3(V) h_\infty(0)(V - E_{\text{Na}}) + \bar{g}_L(V - E_L). \tag{22}$$

The graph of $f(V)$ is shown in Fig. 11(a) and $f(V)$ can be approximated by a linear function, where $R = 1/0.65$ kΩ and the time constant for LIF is $\tau = RC_m = 1.54$ ms. The comparison between the LIF model and the forward kernel from the ASLVN are shown in Fig. 11(b), and they match very well in their time course.

We can use the out-of-sample data to compare the performances of different Volterra-type models. The prediction results of NARV, the traditional PDM method, and ASLVN are summarized in Fig. 12, where ASLVN has a prediction power very similar to the NARV model, but the number of parameters N_{param} of ASLVN is significantly reduced to 18.

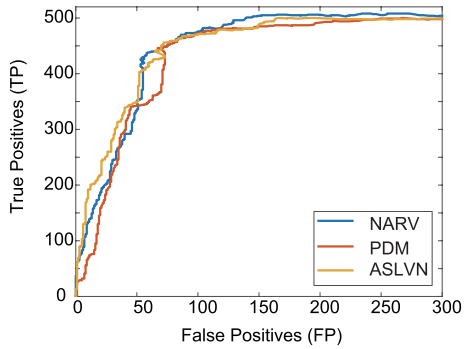


Fig. 13. ROC plots for NARV, PDM, and ASLVN models.

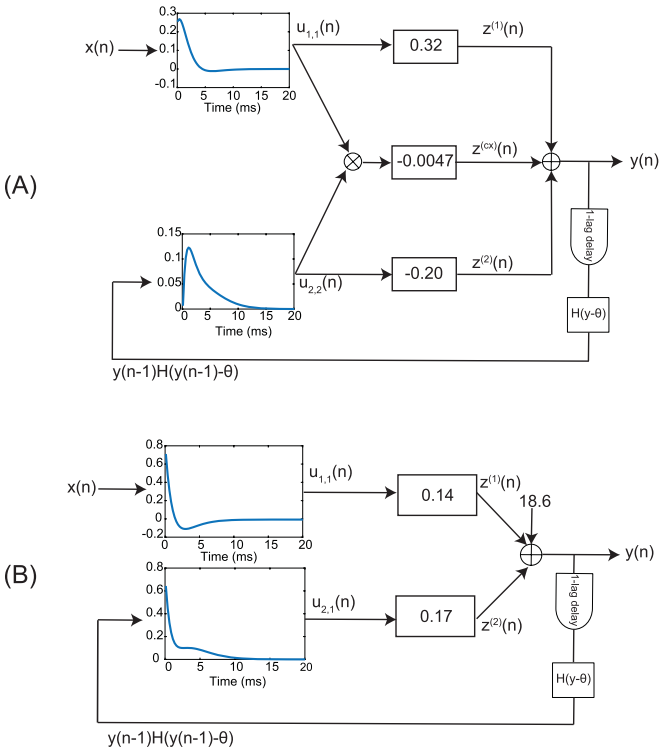


Fig. 14. Obtained ASLVN input–output model of the H-H equations with blocked ion channels. (a) Blocked Na^+ channel. (b) Blocked K^+ channel.

For the traditional PDM method, the prediction power is a bit worse than the other two methods and N_{param} is still large because many cross terms are still included. Another common way to evaluate the model performance is by receiver operating characteristic (ROC) plots, which are shown in Fig. 13. It can be seen that the performance of ASLVN is superior to others at low false-positive rates (FPRs), while at a high FPR, the ASLVN is just slightly worse than the NARV model, but better than the traditional PDM model.

It is instructive to explore the changes of underlying dynamics in the ASLVN for modeling the H-H equations under the effects of tetrodotoxin (TTX) and tetraethylammonium (TEA), which block the Na^+ and K^+ channel, respectively, i.e., we set $\bar{g}_{\text{Na}}$ or $\bar{g}_K$ equal to zero in the H-H equations to represent the blocking effects of TTX or TEA, respectively. The obtained ASLVN models are shown in Fig. 14(a) and (b). As we

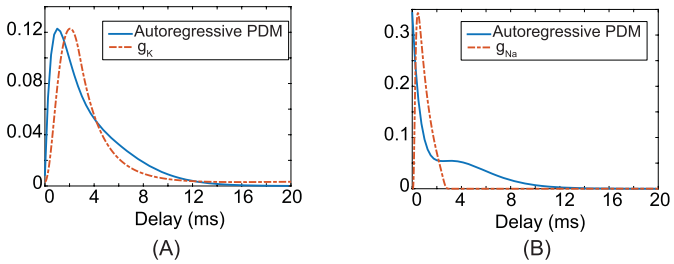


Fig. 15. (a) Comparison of the estimated autoregressive PDM in Fig. 14(a) and K^+ conductance during an AP. (b) Comparison of the estimated autoregressive PDM in Fig. 14(B) and Na^+ conductance during an AP.

can see, blocking the Na^+ channel has little effect on the forward PDM but affects the autoregressive PDM that exhibits the dynamics of the K^+ channel (the difference is caused by the autoregressive delay in ASLVN), as shown in Fig. 15(a). The associated PAF coefficient is negative, which corresponds to the fact that the K^+ current is flowing out of the neuronal membrane. On the other hand, as shown in Fig. 14(b), blocking the K^+ channel should cause the membrane potential to stay depolarized, so that a constant offset y_0 is added to the ASLVN output to compensate for the bias, and cross terms are not necessary because no refractory property exists for the blocked K^+ channel. The estimated forward PDM is underdamped and the autoregressive PDM is different from the control state in Fig. 15(b), but still retains some of the characteristics of the Na^+ channel.

VI. Conclusion

In this paper, we have examined the use of the SA algorithm for training the LVN, which is a Volterra-equivalent general model form for nonlinear dynamic systems. SA training offers some advantages over traditional BP training in terms of avoiding local minima and convergence problems, which are caused by the nonmonotonic activation functions (typically polynomial) employed in the LVN–in contrast to the monotonic activation functions employed by perceptron-type ANNs. Another advantage of SA over BP is that it does not require complex gradient calculations (thus reducing the computational burden) and it removes the restriction of differentiability of the cost function. Thus, ℓ_1 regularization (nondifferentiable cost function) becomes feasible.

On the other hand, SA has the disadvantage of slow training speed because it can only update one parameter at each iteration, while the BP method can update all the parameters. This is probably why BP still dominates the ANN training applications. However, in the context of LVN, the training speed of SA is comparable to BP for the following three reasons.

1) Unlike ANN, which has usually many hidden nodes, LVN has usually a small number of hidden units (<10).
2) BP requires complex calculations of gradients, whereas SA needs only simple calculations.
3) Several techniques exist for accelerated SA (e.g., using exponential decay cooling schedule).

We also introduce a model order pruning technique to determine the model order for sparse LVN with ℓ_1 regularization. In a simulated system study, the pruning technique was shown to be effective and converged fast to the true model order.

We tested this new nonlinear model training approach with a new model structure of ASLVN that is applicable to input–output modeling of complex nonlinear systems that exhibit transitions in dynamic state (e.g., between subthreshold and suprathreshold behaviors), initially demonstrating its use on the celebrated H-H equations of neuronal firing. A compact and accurate ASLVN model was obtained using the SA algorithm and the model order pruning technique (shown in Fig. 8) that consists of only one forward PDM and two autoregressive PDMs. The resulting model predictions (shown in Figs. 12 and 13) indicate that the ASLVN has better predictive power than the traditional PDM method and is comparable to the recently introduced NARV method. The compactness of the ASLVN H-H model (shown in Fig. 10) also facilitates the physiological interpretation of the model. Compared with the traditional LIF model, our ASLVN model (shown in Fig. 11) exhibits richer and more accurate refractory and afterpotential dynamics, while it shows similar subthreshold capacitive characteristics. For example, the LIF utilizes a fixed absolute refractory period, while the ASLVN model explicitly captures the refractory characteristics of the neuron via the autoregressive components and cross terms. Finally, it was shown that the ASLVN can delineate the effects of altered dynamics in the Na^+ and K^+ ion channels upon AP generation, as shown in Figs. 14 and 15.

We finally note that the PDMs obtained from LVN/ASLVN are not orthogonal (as in the traditional PDM analysis), which may provide more flexible model representations and better physiological interpretation. Furthermore, LVN/ASLVN uses the SA training algorithm and the prunning technique for sparse modeling to adjust the (nonorthogonal) PDMs so that they fit the system structure in a parsimonious manner, which may minimize the estimation errors and information loss caused by the kernel-based estimation method of traditional (orthogonal) PDMs.

Appendix
Mathematical Representation of LVN and the BP Training Method

The basic structure of the LVN is shown in Fig. 1. First, the input data $x(n)$ is preprocessed by a filter bank of DLFs $b_0, \ldots, b_{L-1}$, which are given by [26]

$$b_j(m) = \alpha^{(m-j)/2}(1-\alpha)^{1/2} \times \sum_{k=0}^{j}(-1)^k \binom{m}{k}\binom{j}{k}\alpha^{j-k}(1-\alpha)^k \qquad \text{(A1)}$$

where m is from 0 to $M-1$ (M is the memory length of the system). The parameter α is critical for controlling the exponential declining rate of DLF: larger α means slower and longer response. Therefore, system kernels with longer memory length would require larger α to guarantee efficient representation. The outputs of filter bank can be written

as the convolution between input $x(n)$ and DLFs

$$v_j(n) = \sum_{m=0}^{M-1} b_j(m)x(n-m). \tag{A2}$$

To reduce the computational complexity, an iterative method is proposed in [30] to autorecursively calculate $v_j(n)$

$$v_j(n) = \sqrt{\alpha}[v_j(n-1) + v_{j-1}(n)] - v_{j-1}(n-1) \tag{A3}$$

and

$$v_0(n) = \sqrt{\alpha}v_0(n-1) + T\sqrt{(1-\alpha)}x(n) \tag{A4}$$

where T is the sampling interval and the inputs of the hidden units $u(n)$ is the weighted summation of $v_j(n)$

$$u_h(n) = \sum_{j=0}^{L-1} w_{j,h}v_j(n). \tag{A5}$$

Then $u_h(n)$ is propagate to the hidden nodes with Qth-order PAFs

$$z_h(n) = \sum_{q=1}^{Q} c_{q,h}u_h^q(n). \tag{A6}$$

Finally, adding all the outputs of hidden units and the constant offset value y_0, we can get the output for the entire LVN

$$y(n) = \sum_{h=1}^{H} z_h(n) + y_0. \tag{A7}$$

By summarizing the previous equations, the output $y(n)$ can be written as

$$y(n) = \sum_{h=1}^{H}\sum_{q=1}^{Q} c_{q,h}\left[\sum_{j=0}^{L-1} w_{j,h}v_j(n)\right]^q + y_0. \tag{A8}$$

PDM can be represented by the weighted summation of the DLFs

$$p_h(m) = \sum_{j=0}^{L-1} w_{j,h}b_j(m). \tag{A9}$$

The outputs of the hidden nodes can be written as

$$u_h(n) = T\sum_{m=0}^{M-1} p_h(m)x(n-m). \tag{A10}$$

The output of LVN can also be expressed in terms of PDMs

$$y(n) = \sum_{h=1}^{H}\sum_{q=1}^{Q} c_{q,h}\left[T\sum_{m=0}^{M-1} p_h(m)x(n-m)\right]^q + y_0. \tag{A11}$$

Compare the output of LVN in (A11) with that of the discrete Volterra model (DVM)

$$y(n) = k_0 + T\sum_{m=1}^{M} k_1(m)x(n-m) + T^2\sum_{m_1=1}^{M}\sum_{m_2=1}^{M} k_2(m_1,m_2)x(n-m_1)x(n-m_2) + \cdots \tag{A12}$$

We can show that LVN is equivalent to DVM

$$k_1(m) = \sum_{h=1}^{H} c_{1,h}p_h(m) \tag{A13}$$

$$k_2(m_1,m_2) = \sum_{h=1}^{H} c_{2,h}p_h(m_1)p_h(m_2) \tag{A14}$$

$$k_Q(m_1,\ldots m_Q) = \sum_{h=1}^{H} c_{Q,h}p_h(m_1)\ldots p_h(m_Q). \tag{A15}$$

The training of the aforementioned four groups of parameters, i.e., the DLF parameter α, the PDM weights $w_{j,h}$, the polynomial activation coefficients $c_{q,h}$, and the constant offset value y_0, was previously achieved by the use of the BP algorithm that is outlined as follows. Define the error for each data point as

$$\varepsilon(n) = d(n) - y(n) \tag{A16}$$

where $d(n)$ is the actual output and $y(n)$ is the estimated output of LVN. The cost function is often defined as the square of the error

$$J(n) = \frac{1}{2}\varepsilon(n)^2 = \frac{1}{2}(d(n) - y(n))^2. \tag{A17}$$

Use p denoted by a single parameter in LVN each time we update p using gradient descent, and the step change for p in each iteration should be

$$\Delta p = p^{\text{new}} - p^{\text{old}} = -\gamma_p\frac{\partial J(n)}{\partial p} = \gamma_p\varepsilon(n)\frac{\partial y(n)}{\partial p} \tag{A18}$$

where γ_p is the learning rate for parameter p. Applying chain rules for (A8) with respect to each parameter, we can get the first-order derivative $(\partial y(n)/\partial p)$. Substituting it in (A18), the step change for each type of parameter can be expressed by

$$\Delta\beta = \gamma_\beta\varepsilon(n)\sum_{h=1}^{H} f_h'[u_h(n)]\sum_{j=0}^{L-1} w_{j,h}[v_j(n-1)+v_{j-1}(n)] \tag{A19}$$

$$\Delta w_{j,h} = \gamma_w\varepsilon(n)f_h'[u_h(n)]v_j(n) \tag{A20}$$

$$\Delta c_{q,h} = \gamma_c\varepsilon(n)[u_h(n)]^q \tag{A21}$$

$$\Delta y_0 = \gamma_y\varepsilon(n) \tag{A22}$$

where $\beta = \sqrt{\alpha}$ and $f_h'[u_h(n)]$ is the derivative of the PAF of the hth hidden unit

$$f_h'[u_h(n) = \sum_{q=1}^{Q} qc_{q,h}[u_h(n)]^{q-1}. \tag{A23}$$

REFERENCES

[1] L. F. Abbott and T. B. Kepler, "Model neurons: From Hodgkin–Huxley to Hopfield," in *Statistical Mechanics of Neural Networks*, L. Garrido, Ed. New York, NY, USA: Springer, 1990, pp. 5–18.

[2] K. Alataris, T. W. Berger, and V. Z. Marmarelis, "A novel network for nonlinear modeling of neural systems with arbitrary point-process inputs," *Neural Netw.*, vol. 13, no. 2, pp. 255–266, Mar. 2000.

[3] S. A. Billings and L. A. Aguirre, "Effects of the sampling time on the dynamics and identification of nonlinear models," *Int. J. Bifurcation Chaos*, vol. 5, no. 6, pp. 1541–1556, 1995.

[4] V. Černý, "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm," *J. Optim. Theory Appl.*, vol. 45, no. 1, pp. 41–51, Jan. 1985.

[5] S. E. Eikenberry and V. Z. Marmarelis, "A nonlinear autoregressive Volterra model of the Hodgkin–Huxley equations," *J. Comput. Neurosci.*, vol. 34, no. 1, pp. 163–183, Feb. 2013.

[6] S. E. Eikenberry and V. Z. Marmarelis, "Principal dynamic mode analysis of the Hodgkin–Huxley equations," *Int. J. Neural Syst.*, vol. 25, no. 2, p. 1550001, Mar. 2015.

[7] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-6, no. 6, pp. 721–741, Nov. 1984.

[8] J. Guckenheimer and J. S. Labouriau, "Bifurcation of the Hodgkin and Huxley equations: A new twist," *Bull. Math. Biol.*, vol. 55, pp. 937–952, Sep. 1993.

[9] M. H. Hassoun, *Fundamentals of Artificial Neural Networks*. Cambridge, MA, USA: MIT Press, 1995.

[10] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. New York, NY, USA: Springer, 2009.

[11] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *J. Physiol.*, vol. 117, no. 4, pp. 500–544, 1952.

[12] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Netw.*, vol. 1, no. 4, pp. 295–307, 1988.

[13] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[14] V. Z. Marmarelis, "Identification of nonlinear biological systems using Laguerre expansions of kernels," *Ann. Biomed. Eng.*, vol. 21, no. 6, pp. 573–589, Nov. 1993.

[15] V. Z. Marmarelis, *Nonlinear Dynamic Modeling of Physiological Systems*. Hoboken, NJ, USA: Wiley, 2004.

[16] V. Z. Marmarelis, K. H. Chon, N.-H. Holstein-Rathlou, and D. J. Marsh, "Nonlinear analysis of renal autoregulation in rats using principal dynamic modes," *Ann. Biomed. Eng.*, vol. 27, no. 1, pp. 23–31, Jan. 1999.

[17] V. Z. Marmarelis, M. Juusola, and A. S. French, "Principal dynamic mode analysis of nonlinear transduction in a spider mechanoreceptor," *Ann. Biomed. Eng.*, vol. 27, no. 3, pp. 391–402, May 1999.

[18] V. Z. Marmarelis, D. C. Shin, D. Song, R. E. Hampson, S. A. Deadwyler, and T. W. Berger, "Nonlinear modeling of dynamic interactions within neuronal ensembles using principal dynamic modes," *J. Comput. Neurosci.*, vol. 34, no. 1, pp. 73–87, Feb. 2013.

[19] V. Z. Marmarelis, D. C. Shin, M. E. Orme, and R. Zhang, "Closed-loop dynamic modeling of cerebral hemodynamics," *Ann. Biomed. Eng.*, vol. 41, no. 5, pp. 1029–1048, 2013.

[20] V. Z. Marmarelis, D. C. Shin, and R. Zhang, "Linear and nonlinear modeling of cerebral flow autoregulation using principal dynamic modes," *Open Biomed. Eng. J.*, vol. 6, pp. 42–55, Apr. 2012.

[21] V. Z. Marmarelis, D. C. Shin, D. Song, R. E. Hampson, S. A. Deadwyler, and T. W. Berger, "On parsing the neural code in the prefrontal cortex of primates using principal dynamic modes," *J. Comput. Neurosci.*, vol. 36, no. 3, pp. 321–337, Jun. 2014.

[22] V. Z. Marmarelis, D. C. Shin, M. Orme, and R. Zhang, "Time-varying modeling of cerebral hemodynamics," *IEEE Trans. Biomed. Eng.*, vol. 61, no. 3, pp. 694–704, Mar. 2014.

[23] V. Z. Marmarelis, "Modeling methology for nonlinear physiological systems," *Ann. Biomed. Eng.*, vol. 25, no. 2, pp. 239–251, Mar. 1997.

[24] V. Z. Marmarelis and X. Zhao, "Volterra models and three-layer perceptrons," *IEEE Trans. Neural Netw.*, vol. 8, no. 6, pp. 1421–1433, Nov. 1997.

[25] G. D. Mitsis, R. Zhang, B. Levine, and V. Z. Marmarelis, "Modeling of nonlinear physiological systems with fast and slow dynamics. II. Application to cerebral autoregulation," *Ann. Biomed. Eng.*, vol. 30, no. 4, pp. 555–565, Apr. 2002.

[26] G. D. Mitsis, M. J. Poulin, P. A. Robbins, and V. Z. Marmarelis, "Nonlinear modeling of the dynamic effects of arterial pressure and CO_2 variations on cerebral blood flow in healthy humans," *IEEE Trans. Biomed. Eng.*, vol. 51, no. 11, pp. 1932–1943, Nov. 2004.

[27] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 4. 1995, pp. 950–957.

[28] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA, USA: MIT Press, 2012.

[29] Y. Nourani and B. Andresen, "A comparison of simulated annealing cooling strategies," *J. Phys. A, Math. General*, vol. 31, no. 41, p. 8373, 1998.

[30] H. Ogura, "Estimation of Wiener kernels of a nonlinear system and a fast algorithm using digital Laguerre filters," in *Proc. 15th NIBB Conf.*, Okazaki, Japan, 1985, pp. 14–62.

[31] D. C. Plaut, "Experiments on learning by back propagation," Dept. Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CS-86-126, 1986.

[32] G. Schwarz, "Estimating the dimension of a model," *Ann. Statist.*, vol. 6, no. 2, pp. 461–464, 1978.

[33] D. Song, H. Wang, and T. W. Berger, "Estimating sparse Volterra models using group L1-regularization," in *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Aug./Sep. 2010, pp. 4128–4131.

[34] D. Song *et al.*, "Identification of sparse neural functional connectivity using penalized likelihood estimation and basis functions," *J. Comput. Neurosci.*, vol. 35, no. 3, pp. 335–357, Dec. 2013.

[35] P. Z. Marmarelis and K.-I. Naka, "White-noise analysis of a neuron chain: An application of the Wiener theory," *Science*, vol. 175, no. 4027, pp. 1276–1278, 1972.

[36] P. Z. Marmarelis and V. Z. Marmarelis, *Analysis of Physiological Systems: The White-Noise Approach*. New York, NY, USA: Plenum, 1978.

[37] V. Z. Marmarelis, "Signal transformation and coding in neural systems," *IEEE Trans. Biomed. Eng.*, vol. 36, no. 1, pp. 15–24, Jan. 1989.

[38] K. Geng and V. Z. Marmarelis, "Pattern recognition of Hodgkin–Huxley equations by auto-regressive Laguerre Volterra network," *BMC Neurosci.*, vol. 16, p. 156, Dec. 2015.

[39] E. Marder and A. L. Taylor, "Multiple models to capture the variability in biological neurons and networks," *Nat. Neurosci.*, vol. 14, no. 2, pp. 133–138, 2011.

[40] W. A. Catterall, I. M. Raman, H. P. C. Robinson, T. J. Sejnowski, and O. Paulsen, "The Hodgkin–Huxley heritage: From channels to circuits," *J. Neurosci.*, vol. 32, no. 41, pp. 14064–14073, 2012.

[41] C. O'Donnell and M. C. W. Van Rossum, "Systematic analysis of the contributions of stochastic voltage gated channels to neuronal noise," *Frontiers Comput. Neurosci.*, vol. 8, p. 105, 2014.

[42] R. F. Fox and Y.-N. Lu, "Emergent collective behavior in large numbers of globally coupled independently stochastic ion channels," *Phys. Rev. E*, vol. 49, no. 4, pp. 3421–3431, Apr. 1994.

[43] J. H. Goldwyn, N. S. Imennov, M. Famulare, and E. Shea-Brown, "Stochastic differential equation models for ion channel noise in Hodgkin–Huxley neurons," *Phys. Rev. E*, vol. 83, no. 4, p. 041908, Apr. 2011.

[44] M. Güler, "Stochastic Hodgkin–Huxley equations with colored noise terms in the conductances," *Neural Comput.*, vol. 25, no. 1, pp. 46–74, Jan. 2013.

[45] L. Yu and L. Liu, "Optimal size of stochastic Hodgkin–Huxley neuronal systems for maximal energy efficiency in coding pulse signals," *Phys. Rev. E*, vol. 89, no. 3, p. 032725, Mar. 2014.

[46] Q. Kang, B. Huang, and M. Zhou, "Dynamic behavior of artificial Hodgkin–Huxley neuron model subject to additive noise," *IEEE Trans. Cybern.*, vol. PP, no. 99, pp. 1–1, Aug. 2015.

[47] M. Saggar, T. Mericli, S. Andoni, and R. Miikkulainen, "System identification for the Hodgkin–Huxley model using artificial neural networks," in *Proc. Int. Joint Conf. Neural Netw.*, Orlando, FL, USA, Aug. 2007, pp. 2239–2244.

[48] W. M. Kistler, W. Gerstner, and J. L. van Hemmen, "Reduction of the Hodgkin–Huxley equations to a single-variable threshold model," *Neural Comput.*, vol. 9, no. 5, pp. 1015–1045, Jul. 1997.

[49] R. Jolivet, A. Rauch, H.-R. Lüscher, and W. Gerstner, "Predicting spike timing of neocortical pyramidal neurons by simple threshold models," *J. Comput. Neurosci.*, vol. 21, no. 1, pp. 35–49, Aug. 2006.

[50] A. A. Lazar, "Population encoding with Hodgkin–Huxley neurons," *IEEE Trans. Inf. Theory*, vol. 56, no. 2, pp. 821–837, Feb. 2010.

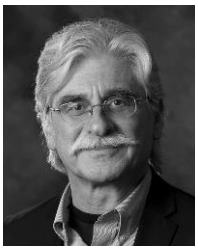
[51] R. A. Sandler, D. Song, R. E. Hampson, S. A. Deadwyler, T. W. Berger, and V. Z. Marmarelis, "Model-based assessment of an *in-vivo* predictive relationship from CA1 to CA3 in the rodent hippocampus," *J. Comput. Neurosci.*, vol. 38, no. 1, pp. 89–103, Feb. 2015.

[52] R. A. Sandler and V. Z. Marmarelis, "Understanding spike-triggered covariance using Wiener theory for receptive field identification," *J. Vis.*, vol. 15, no. 9, p. 16, Jul. 2015.

[53] Y. Kang, J. Escudero, D. C. Shin, E. Ifeachor, and V. Z. Marmarelis, "Principal dynamic mode analysis of EEG data for assisting the diagnosis of Alzheimer's disease," *IEEE J. Transl. Eng. Health Med.*, vol. 3, 2015, Art. no. 1800110.

Kunling Geng (S'15) received the B.S. degree in electrical and computer engineering from Shanghai Jiao Tong University, Shanghai, China, in 2012. He is currently pursuing the Ph.D. degree in biomedical engineering with the University of Southern California, Los Angeles, CA, USA.

His current research interests include developing compact and powerful methods based on Volterra model to analyze the complex and nonlinear physiological systems, as well as interpret and obtain the insights of the data.



Vasilis Z. Marmarelis (F'97) received the Ph.D. degree in engineering science from the California Institute of Technology, Pasadena, CA, USA, in 1976.

He is a Professor of Biomedical Engineering with the University of Southern California, Los Angeles, CA, USA, and the Co-Director of the Biomedical Simulations Resource, a research center funded by NIH since 1985. He served as the Department Chairman from 1990 to 1996. He has co-authored the seminal book entitled *Analysis of Physiological System: The White Noise Approach* (1978; Russian, 1981; Chinese, 1990) and authored the monograph entitled *Nonlinear Dynamic Modeling of Physiological Systems* (2004). He has published more than 150 journal papers and book chapters. His current research interests include dynamic nonlinear modeling of biomedical systems, neural information processing, modeling of physiological autoregulation, multimodal ultrasound tomography (MUT) for diagnostic imaging, and model-based diagnostic physiomarkers. The key application domains of interest are: cerebral hemodynamics and flow regulation, neurodegenerative and cerebrovascular disease, neurostimulation to treat brain disorders, endocrine-metabolic regulation and diabetes, and non-invasive lesion differentiation via MUT diagnostic imaging. In 2000, he invented the MUT diagnostic imaging system for early detection of breast cancer, which is currently clinically evaluated in Europe.

Dr. Marmarelis is a fellow of AIMBE. He served on the IEEE Editorial Board for many years.