2018 IEEE Conference on Decision and Control (CDC)
Miami Beach, FL, USA, Dec. 17-19, 2018

Fast Adaptive Hinging Hyperplanes

Qinghua Tao[1,2], Jun Xu[3], Johan A.K. Suykens[2] *IEEE Fellow* and Shuning Wang[1]

***Abstract*— This paper proposes a fast algorithm for the training of adaptive hinging hyperplanes (AHH), which is a popular and effective continuous piecewise affine (CPWA) model consisting of a linear combination of basis functions. The original AHH incrementally generates new basis functions by simply traversing all the existing basis functions in each dimension with the pre-given knots. Meanwhile, it also incorporates a backward procedure to delete redundant basis functions, which avoids over-fitting. In this paper, we accelerate the procedure of AHH in generating new basis functions, and the backward deletion is replaced with Lasso regularization, which is robust, requires less computation, and manages to prevent over-fitting. Besides, the selection of the splitting knots based on training data is also discussed. Numerical experiments show that the proposed algorithm significantly improves the efficiency of the existing AHH algorithm even with higher accuracy and it also enhances robustness in the given benchmark problems.**

I. INTRODUCTION

System identification works by building a mathematical model, which describes the intrinsic relationship with given data. In applications, the relationships among the data are usually described with different levels of nonlinearity. Therefore, the choice of the mathematical model becomes crucial, since the chosen model has to be flexible enough to describe the given data. At the same time, the identification method to estimate the parameters of the model is also required to be effective with respect to efficiency, accuracy and robustness.

In nonlinear systems, continuous piecewise affine (CPWA) functions are popular for modeling, since any continuous function can be arbitrarily approximated by a CPWA function in a compact set. A CPWA function $f(\mathbf{x})$ equals different affine function $f_k(\mathbf{x})$ in different sub-region Ω_k within $\Omega = \bigcup \Omega_k$, $\mathring{\Omega}_{k_1} \bigcap \mathring{\Omega}_{k_2} = \emptyset$, $\forall k_1 \neq k_2$, where $\mathring{\Omega}_k$ is the interior of Ω_k with $f_{k_1}(\mathbf{x}) = f_{k_2}(\mathbf{x}), \forall \mathbf{x} \in \Omega_{k_1} \bigcap \Omega_{k_2}$ [1]. However, this formula is ineffective to apply. Therefore, compact representations for CPWL have been proposed. They are usually described in a linear combination of basis functions,

$$f(\mathbf{x}) = \sum_{m=1}^{M} a_m B_m(\mathbf{x}), \tag{1}$$

where $f(\mathbf{x})$ and $B_m(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ is also CPWA.

A series of CPWA models have been established in recent years [2], [3], [4], [5]. Among these models, hinging hyperplane (HH) has attracted extensive attention due to the model simplicity and explicit geometric meaning [2], [6]. However, HH cannot represent all the CPWA functions in more than two dimensions. In [4], Wang proposed the generalized hinging hyperplane (GHH) model, which can represent any CPWA function in all dimensions. The strong representation ability of GHH leads to complicated structure. To make a tradeoff between model complexity and flexibility, adaptive hinging hyperplanes (AHH) was proposed, and it has shown excellent performance in function approximation, system identification, predicative control, etc [5], [7].

AHH was originally constructed as an analogy to multivariate adaptive regression splines (MARS), which is proposed for flexible regression modeling and can be regarded as a generalization of recursive partitioning regression [8]. It has been shown that AHH has more flexibility than MARS with piecewise linear splines. AHH can also be regarded as a special case of GHH with simpler model structure.

Analogous to MARS, the existing AHH is identified through recursively partitioning the domain, and mainly consists of two parts, which are the forward and backward steps. In the forward procedure, the domain is progressively partitioned and new basis functions are generated incrementally. Specifically, it sequentially chooses one of the existing basis functions as the root, and then traverses the candidate knots to select the one with the largest decrease in fitting error to further split the domain, generating two new basis functions. To avoid over-fitting, the backward step tentatively checks all the basis functions and deletes the redundant ones.

In this paper, we develop a fast adaptive hinging hyperplanes (FAHH) algorithm, which effectively reduces the traverse in the forward procedure and replaces the backward procedure with Lasso regularization to further improve the efficiency and also enhance the accuracy and robustness. Besides, the knots selection is also discussed. Numerical experiments show that FAHH ouperforms the existing AHH in efficiency while help to approach higher accuracy in both function approximation and dynamic system identification.

The rest of this paper is organized as follows. In Section 2, the preliminaries are presented. Section 3 introduces the proposed FAHH algorithm, which is tested in Section 4 with

*This work is jointly supported by the National Natural Science Foundation of China (61473165, 61134012), Chinese Scholarship Council (CSC) and Science and Technology Innovation Committee of Shenzhen Municipality (JCYJ2017-0811-155131785). Johan Suykens acknowledges support by Research Council KU Leuven CoE PFV/10/002 (OPTEC), FWO projects: G0A4917N, G.088114N, ERC Advanced Grant E-DUALITY (787960).

[1,2]Qinghua Tao is with TNList, Department of Automation, Tsinghua University, Beijing, 100084, China, and with STADIUS, ESAT, KU Leuven, Leuven, 3001, Belgium taoqh14@mails.tsinghua.edu.cn

[3]Jun Xu is with School of Mechanical Engineering and Automation, Harbin Institute of Technology, Shenzhen, 518055, China xujunqgy@hit.edu.cn

[2]Johan A.K. Suykens is with STADIUS, ESAT, KU Leuven, Leuven, 3001, Belgium johan.suykens@esat.kuleuven.be

[1]Shuning Wang is with TNList, Department of Automation, Tsinghua University, Beijing, 100084, China swang@mail.tsinghua.edu.cn

numerical experiments. Section 5 concludes the paper.

II. PRELIMINARIES

A. Model of Adaptive Hinging Hyperplanes

MARS was first introduced by Friedman in [8], and can be considered as a generalization of recursive partitioning regression. The basic idea of MARS is to partition the entire domain recursively, assigning a basis function for each sub-region. Then the approximation model of MARS is constructed as a linear combination of these basis functions, which are expressed in terms of the product of truncated power spline function $[\pm(\mathbf{x}(v)-\beta)]_+^q$, where $\mathbf{x}\in\mathbb{R}^n$, $[\cdot]_+$ denotes the positive part of the expression, q is the order of univariate spline functions, $\mathbf{x}(v)$ is the splitting variable located in the v-th dimension and β is the splitting knot.

The order q of the spline functions in MARS is usually set as $q=1$. When $q=1$, MARS is formulated as

$$f(\mathbf{x}) = a_0 + \sum_{m=1}^{M} a_m \prod_{k=1}^{K_m} [s_{mk}\cdot(\mathbf{x}(v_{mk})-\beta_{mk})]_+, \quad (2)$$

where $s_{mk}=\pm 1$, K_m is the number of the splits that give rise to $B_m(\mathbf{x})$ and $K_m \le n$, since the splitting variables are required to be distinctive. The truncated power spline function of MARS is equivalent to $\max\{0,\pm(\mathbf{x}(v_{mk})-\beta)\}$, which resembles the HH models. Inspired by this, model (2) is extended to obtain a generalized hinging function based on nonadditive MARS model, i.e., AHH. The model of AHH is constructed by converting product operator "$\prod$" to "min", hence the basis function $B_m(\mathbf{x})$ of AHH is

$$\min_{k\in\{1,\dots,K_m\}} \max\{0, s_{mk}\cdot(\mathbf{x}(v_{mk})-\beta_{mk})\}. \quad (3)$$

It is shown that AHH can partition the domain into more subregions than MARS and AHH also achieves higher testing accuracy in [5]. A toy example is conducted for $y=\sin(0.83\pi x_1)\cos(1.25\pi x_2)$ in Fig 1, where the identified AHH model brings more flexibility than MARS.

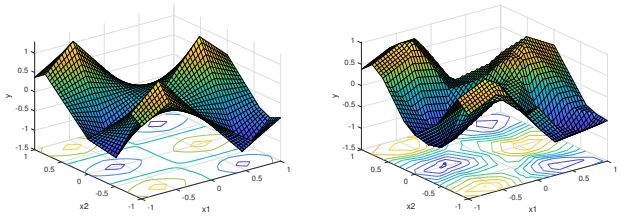

Fig. 1. The identification results of MARS (left figure) and AHH (right figure) based on the fixed settings with the same identification algorithm.

Although AHH is derived based on MARS, it still belongs to the family of CPWA with hinging basis, which is basically formulated as hinge-shaped functions (HH), say

$$\pm\max\{0, L_m(\mathbf{x})\}, m=1,\dots,M \quad (4)$$

where L_m is a linear function [2]. To improve the representation ability of HH, GHH model is proposed in [4] with global representation ability in $\mathbb{R}^n$. In fact, AHH is equivalent to a special case of GHH model, whose basis function is

$$\pm\max\{L_{m,1}(\mathbf{x}),\dots,L_{m,k_m}(\mathbf{x})\}, m=1,...,M, \quad (5)$$

where k_m is the number of linear functions in $B_m(\mathbf{x})$.

AHH has been proven to be able to approximate any continuous function in a compact set. Due to these properties, it is worthy of shading more light on the research of AHH.

B. Identification of AHH

The model of AHH is formulated as follows

$$f(\mathbf{x}) = a_0 + \sum_{m=1}^{M} a_m \min_{k\in\{1,\dots,K_m\}} \max\{0, s_{mk}(\mathbf{x}(v_{mk})-\beta_{mk})\}, \quad (6)$$

where $\mathbf{x}\in\mathbb{R}^n$ and $K_m \le n$. Similar to MARS, the identification can be interpreted as a tree where each node corresponds to a basis function. Meanwhile, the split of the tree (domain) is used to further partition the corresponding node (sub-region) and incrementally generate new basis functions which yield the best fit. The measurement of fit in MARS and AHH is a modified form of the generalized cross validation (GCV) criterion originally proposed in [9]

$$\mathrm{LOF}(\hat{f}_{M_0}) = \frac{1}{N}\frac{\sum_{i=1}^{N}(y_i-\hat{f}_{M_0}(\mathbf{x}_i))^2}{(1-\tilde{C}(M_0)/N)^2}, \quad (7)$$

where M_0 is the current number of basis functions, $\mathbf{B}\in\mathbb{R}^{M_0\times N}$ is data matrix [8], $\tilde{C}(M_0)=\mathrm{trace}(\mathbf{B}(\mathbf{B}^T\mathbf{B})\mathbf{B}^T)+1+dM_0$, and d is the smoothing parameter depending on the problems. The same as [5], we set $d=2$ in this paper.

In the forward procedure, all the existing basis functions are traversed to be taken as the parental basis, and then all the pre-given candidate knots in each dimension are searched to find the one with the lowest LOF to generate two new basis functions. To prevent over-fitting, the backward step is then introduced to tentatively delete the basis functions to reduce redundancy. The details are given in Algorithm 1.

III. FAST ADAPTIVE HINGING HYPERPLANES

AHH model shows great flexibility in performance, but its identification procedure is prohibitive in efficiency due to the exhaustive search. In the proposed FAHH algorithm, inspired by [11], the exhaustively complete search is modified to be more concise: only the most promising trials are kept and some potentially redundant searches can be omitted to reduce computational burden. Meanwhile, the backward procedure is replaced with Lasso regularization to further increase the efficiency with robustness. The data-based knots selection method is also discussed.

A. Reduction In Traverse

Given the data $\{\mathbf{x}_i\}_{i=1}^N$ and $M=2I-1$ where I is the number of iterations and $\{a_m\}_{m=1}^M$ is obtained by LS method whose computation is proportional to $C_{LS}=NM^2$. We may assume that there are J candidate knots in each dimension.

Algorithm 1: Adaptive Hinging Hyperplanes

Input: $M_{\max}$; $\beta_k \in \{\beta_1, ..., \beta_{J_k}\}$, $k = 1, ..., n$.
Output: The identified AHH model $\hat{f}(\mathbf{x})$.
• Forward Stepwise
Step 1: Set $B_1(\mathbf{x}) = \alpha_1$ and $M = 1$.
Step 2: Sequentially select $B_m(\mathbf{x})$ and v from

Loop 1: $B_m(\mathbf{x}), m \in \{1, \ldots, M\}$
Loop 2: $v \in \{1, \ldots, n\}$ & $v \notin \{v_{mk} | k = 1, \ldots, K_m\}$

with candidate knot $\beta_k \in \{\beta_1, ..., \beta_{J_k}\}$. Let

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{M} a_i B_i(\mathbf{x}) + a_{M+1} B_{M+1}(\mathbf{x}) + a_{M+2} B_{M+2}(\mathbf{x}),$$

and obtain $\{a_i\}_1^{M+2}$ by least squares (LS) method.
Step 3: Choose $B_m^*(\mathbf{x})$, $\mathbf{x}(v^*)$ and β^* with the lowest LOF*, such that
$B_{M+1}(\mathbf{x}) = \min\{B_m^*(\mathbf{x}), \max\{0, \mathbf{x}(v^*) - \beta^*\}\}$
$B_{M+2}(\mathbf{x}) = \min\{B_m^*(\mathbf{x}), \max\{0, -(\mathbf{x}(v^*) - \beta^*)\}\}$
Step 4. $M = M + 2$. If $M < M_{\max}$, go to step 2.
• Backward Stepwise
Step 1: $J^* = \{1, ..., M\}$; LOF*.
Step 2: Sequentially select M_0 from $\{M, ..., 2\}$, and then sequentially delete one basis function from $\{B_2(\mathbf{x}), ..., B_{M_0}(\mathbf{x})\}$. Record the lowest LOF* with corresponding basis functions indexed as J^*.
Step 3: $\hat{f}(\mathbf{x}) = a_1\alpha_1 + \sum_{i \in J^*} a_i B_i(\mathbf{x})$

All the candidate knots $1 \le k \le J$ in each dimension $\mathbf{x}(v)$ based on the existing basis function $B_m(\mathbf{x})$, i.e.,

$$\begin{array}{ll} \text{Loop 1:} & m \in \{1, \ldots, M\} \\ \text{Loop 2:} & v \in \{1, \ldots, n\}, \end{array} \tag{8}$$

are traversed. Thus, in the I-th iteration, the computation is proportional to $nJMC_{LS} = nJNM^3$. For a large-scale data set (large nNJ) and an adequate complexity of the model (large M), the training time of AHH becomes prohibitive. Inspired by [11], the core idea is to alter Loop 1 and Loop 2 from the exhaustive search to a more concise way.

1) Priority queue among basis functions: In Loop 1, the existing AHH algorithm sequentially chooses one of the existing basis functions $\{B_m(\mathbf{x})\}_{m=1}^M$ as the parental basis to generate two new basis functions $B_{M+1}(\mathbf{x})$ and $B_{M+2}(\mathbf{x})$ with the minimal $\mathrm{LOF}_M(m)$. Hence, $\mathrm{LOF}_M(m)$ is computed for all the M existing basis functions with $m^* = \arg\min_{1 \le m \le M} \mathrm{LOF}_M(m)$, where the optimal splitting variable and knot are denoted as $\mathbf{x}(v^*)$ and β^*.

With iterations going on, $\mathrm{LOF}_M(m)$ does not change dramatically by adding new basis functions. Thus, in later iterations the basis functions resulting in lower LOF are more likely to bring a better approximation in the next iteration. The idea is then to construct the parental basis by selecting some of such existing basis functions. Consider $\{\mathrm{LOF}_M(k)\}_{k=1}^M$ sorted in ascending order and let $R_M(m)$ be the ranking order of $\mathrm{LOF}_M(k)$ in the sorted list,

$$R_M(m) = \underset{\mathrm{LOF}_M(m)}{\mathrm{Sort}} \{\mathrm{LOF}_M(k)\}_{k=1}^M. \tag{9}$$

Smaller $R_M(m)$ means that the corresponding basis function $B_m(\mathbf{x})$ has more potential to bring a better result when being chosen as the parental basis. Differently, only the first K basis functions ranked in this list are selected to be the parental basis. For $B_{M+1}(\mathbf{x})$ and $B_{M+2}(\mathbf{x})$, we set $\mathrm{LOF}_M(M+1) = \mathrm{LOF}_M(M+2) = -\infty$ to ensure that the new basis functions are selected as the parental basis at least once, and then update $\{R_M(m) \leftarrow R_M(m) - 2\}_{m=1}^M$.

However, the basis functions with high LOF_M might lose the chance to be tried as the parental basis, while still having some potentials to bring a better result. This problem can be solved by introducing an aging factor β in the priority queue,

$$P_M(m) = R_M(m) + \beta(I - I_m) \tag{10}$$

where I_m is the iteration at which $\mathrm{LOF}_M(m)$ was last computed. Thus it can provide some opportunities for these basis functions to be selected. Hence, Loop 1 becomes

$$\begin{array}{lll} \text{Loop 1}^*\text{:} & \text{If } M < K, & m \in \{1, \ldots, M\} \\ & \text{If } M \ge K, & m \in \{\mathrm{Sort}_i\{P_M(k)\}_{k=1}^M\}_{i=1}^K. \end{array} \tag{11}$$

2) Splitting saving among iterations: In Loop 2, the candidate knots have to be recomputed in each iteration over all dimensions. In Section III-A.1, the underlying assumption motivating the priority queue is that the approximation does not dramatically change with iterations going on. Thus, the optimal splitting variable $\mathbf{x}(v^*)$ of the parental basis $B_{m^*}(\mathbf{x})$ should not substantially change either.

This idea can be achieved with two parameters (l_m, v_m). l_m is the last iteration at which the knots are computed for parental basis $B_m(\mathbf{x})$ over all dimensions $1 \le v \le n$ with $v^* = v_m$. For the first K functions in $P_M(m)$, the exhaustive search over all dimensions $1 \le v \le n$ only conducts if

$$I - l_m > h \tag{12}$$

where h is the interval between two iterations at which the complete search over all dimensions is conducted. Then we update $l_m \leftarrow I$ and $v_m \leftarrow v^*$. If condition (12) does not hold, the computation is conducted only in dimension $v = v_m$. Hence, Loop 2 becomes

$$\begin{array}{lll} \text{Loop 2}^*\text{:} & \text{If } I - l_m > h, & v \in \{1, \ldots, n\} \\ & \text{If } I - l_m \le h, & v = v_m. \end{array} \tag{13}$$

The index h quantifies the frequency of the complete search over all dimensions. The complete traverse is done only if it has been more than h iterations since it was last conducted with complete traverse. For the new basis functions, we set $l_{M+1} = l_{M+2} = -\infty$ so that the complete search can be calculated in the next $(I+1)$-th iteration, and then update $M = M + 2$ and $I = I + 1$ [11]. Larger h and smalled K bring less computation. When $h = 1$ and $K = M$, it degenerates to the original AHH.

B. Lasso Regularization

To some extent, the greedy incremental design of basis functions is to deliberately over-fit the data with an excessively large model. Then, a backward stepwise deletion is incorporated to trim the model back to a proper size, which can also be achieved by introducing Lasso regularization.

Lasso is a shrinkage and selection method for regression. It minimizes the sum of squared errors with a bound on the sum of the absolute values of the coefficients, which can induce coefficients to be exactly 0 [12]. By introducing Lasso regularization, the redundant basis functions can be removed more efficiently. Then, the problem is formulated as

$$\min \frac{1}{2}\sum_{i=1}^{N}\Big(y_i - \sum_{m=1}^{M} a_m B_m(\mathbf{x})\Big)^2 + \lambda \sum_{m=1}^{M} |a_m|, \quad (14)$$

where $B_m(\mathbf{x})$ is the generated basis function and $\lambda > 0$. With Lasso regularization, the whole model is optimized with robustness. The strategy of reducing traverse in Section III-A may bring a slight decrease in accuracy due to the incomplete searching mechanism, while Lasso regularization can provides possibilities improve accuracy.

The problem with Lasso regularization can be efficiently solved with the alternating direction method of multipliers (ADMM) which is designed to solve convex optimization problems by breaking them into smaller pieces [13], [14]. In the existing backward procedure, the computational burden increases evidently with the increase of model complexity and data dimension, while ADMM in our approach does not.

C. Knots Selection

The existing AHH algorithm uniformly selects the splitting knots in each dimension. In FAHH, we provide an alternative to utilize the training data, i.e., the candidate knots can be

$$\beta_1 = \mathbf{x}_j(k), \ldots, \beta_{J_k} = \mathbf{x}_{j+w(J_k-1)}(k), \quad (15)$$

where w is the sampling interval among the data. Equation (15) requires $B_m(\mathbf{x}_j) > 0$ [8], [10]. With identification going on, the number of qualified splitting knots with $B_m(\mathbf{x}_j) > 0$ decreases, which also accelerates the identification.

In summary, the framework of the proposed FAHH algorithm is presented in Algorithm 2.

D. Computation analysis

The computation reduced in FAHH is mainly controlled by the reduced traverse and Lasso regularization. Similarly to the analysis in [11], we introduce the proportion that reduces the total computation to compare with the existing AHH.

In reducing the traverse of parental basis functions and dimensions, the computation is controlled by K and h. When $K = \infty$ and $h = 1$, it degenerates to the original AHH.

For the first K elements in the priority queue, the optimizations over all dimensions are performed with probability $1/h$. The expected computation can be reduced proportionally from Kn to $3n+(K-3(1-1/h+n/h))$, which brings the average computational improvement ratio

$$C(h,K) = (3 + (K-3)((1-1/h)/n + 1/h))/K. \quad (16)$$

Algorithm 2: Fast Adaptive Hinging Hyperplanes

Input: $M_{\max}$; K; h; λ; $\beta_k \in \{\beta_1, \ldots, \beta_{J_k}\}$ where β_k is from the training data, $k = 1, \ldots, n$.
Output: The identified AHH model $\hat{f}(\mathbf{x})$.
• Forward Stepwise
Step 1: Set $B_1(\mathbf{x}) = \alpha_1$ and $M = 1$.
Step 2: Loop $1^* \rightarrow$ sequentially select $B_m(\mathbf{x})$.
Step 3: Loop $2^* \rightarrow$ sequentially select v with β_k. Similar with AHH, apply LS method to obtain $\{a_i\}_{i=1}^{M+2}$.
Step 4: Choose $B_m^*(\mathbf{x})$, $\mathbf{x}(v^*)$ and β^* with the lowest LOF* obtain B_{M+1} and B_{M+2} analogously. Step5: Update information in Loop 1^* and Loop 2^*.
Step 6. $M = M + 2$, If $M < M_{\max}$, go to step 2.
• Lasso Regularization
Apply ADMM to equation (14) and obtain $\hat{f}(\mathbf{x})$.

As mentioned in Section III-A, in each iteration I, the computation of LS method is proportional to $nJNM^3$. Thus, the total computation of I iterations is proportional to

$$W_0 = nJN\big(\tfrac{M(M+1)}{2}\big)^2. \quad (17)$$

Hence, for $m \le K$, the computation proportion W_1 is the same as that for the original AHH, i.e.,

$$W_1 = nJN\big(\tfrac{K(K+1)}{2}\big)^2. \quad (18)$$

When $m > K$, the expected computation becomes proportional to $nJNM^2K \cdot C(h,K)$ according to equation (16). For $m > K$, the computation is proportional to

$$\begin{aligned} W_2 = \quad & nJNK \cdot C(h,K)(M(M+1)(2M+1)/6 \\ & -K(K+1)(2K+1)/6). \end{aligned} \quad (19)$$

Thus, the total computation of FAHH is reduced by ratio $R = (W_1 + W_2)/W_0$ compared to AHH. The computation in FAHH increases with M^3 instead of M^4 in AHH. Decreasing K or increasing h manages to relieve computational burden, but it may affect the accuracy.

In AHH, the backward procedure repeats $(M-1)+(M-2)\ldots+1$ operations of LS method, which makes the total computation proportional to $W_B = N((M-1)M/2)^2 - N((M-1)M(2M-1)/6)$. In FAHH, the backward procedure is replaced with Lasso regularization, which is solved efficiently with ADMM algorithm. Since the computation of ADMM algorithm differs in various problems, we compare the computational cost with numerical tests.

For knots selection based on the training data, the restriction $B_m(\mathbf{x}_j) > 0$ shrinks the number of qualified knots during the identification process. Thus, the computation varies with different data sets. A toy example is illustrated in Section IV-A.

IV. NUMERICAL TESTS

In this section, we evaluate FAHH on problems of function approximation and dynamic system identification. The numerical experiments compare FAHH with the existing AHH and the popular CPWA models of HH and GHH, which are

with the same settings in [5]. All experiments are performed on Matlab R2017 with Intel i7-3770, 3.40GHz CPU, 16G RAM and Windows 10.

A. Evaluation of the proposed strategies

We first present a toy example to evaluate the strategies in FAHH. Consider the test function $y_1 = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + 0x_6, x \in [-1,1]^6$, tested by [1], [5], [15]. The relative squared sum of error (RSSE) is used to evaluate the performance i.e., $\mathrm{RSSE} = \sum_{t=1}^{N}(y(t)-\hat{y}(t))^2 / \sum_{t=1}^{N}(y(t)-\overline{y}(t))^2$. The running time is denoted as $T(s)$, and N is the size of data. We set $M = 40$. Each test is repeated 50 times, and the average is shown. The training and testing data are randomly generated with $N = 2000$. The results of AHH are RSSE_0 and T_0. To enhance the performance, all the data are normalized in this paper.

We try different K and h to do the test. Table I shows that smaller K and large h can relieve computation burden, where $R_E = \mathrm{RSSE}\,/\,\mathrm{RSSE}_0$ and $R_T = T/T_0$, but may bring lower accuracy. When the efficiency is considered as priority, this strategy outperforms the existing AHH.

TABLE I

PERFORMANCE ON y_1.

	$K=30,h=1$	$K=M,h=10$	$K=30,h=10$
R_E	0.0065/0.0064	0.0069/0.0064	0.0070/0.0064
R_T	4.85/5.51	2.76/5.51	2.58/5.51

We apply ADMM to solve FAHH with Lasso regularization, with $K = 30, h = 10$ and $\lambda \in \{0.1, 1, 5, 10\}$. We only compare the backward deletion T_{Back} to Lasso T_{Lasso}.

TABLE II

PERFORMANCE WITH LASSO REGULARIZATION ON y_1.

	$M=40$	
	$N=1000$	$N=2000$
$\mathrm{RSSE}\,/\,\mathrm{RSSE}_0$	0.0106/0.0172	0.0058/0.0064
$T_{\mathrm{Lasso}}/T_{\mathrm{Back}}$	0.04/0.27	0.04/0.33

Table II illustrates that Lasso regularization improves the accuracy and efficiency. Compared to Table I ($M = 40, N = 2000$), the accuracy sacrificed due to the traverse reduction is compensated with Lasso. Besides, we use data-based method to select knots with $w = 100$. RSSE decreases to 0.0053, and T to 1.62s. This strategy provides another improvement, but this property cannot be always guaranteed in all cases.

B. Tests on function approximation

The tests on function approximation are from [1], [5], say $y1$ and $y2$, where $y1$ is the explained in Section IV-A and

$$y_2 = \frac{e^{v_1(\mathbf{x})}}{1+e^{v_1(\mathbf{x})}} + \frac{e^{v_2(\mathbf{x})}}{1+e^{v_2(\mathbf{x})}} + \frac{e^{v_3(\mathbf{x})}}{1+e^{v_3(\mathbf{x})}},\ \mathbf{x} \in [0,1]^{10}.$$

The details of $v_1(\mathbf{x})$, $v_2(\mathbf{x})$ and $v_3(\mathbf{x})$ can be found in [5]. $K = 30, h = 10$ are set. The values in bold have the highest accuracy, and the underlined ones hold the lowest running time. Table III illustrates that FAHH performs with the highest accuracy and significantly improves the efficiency of AHH. The performance of HH and GHH may be prohibitive, since its gradient-based algorithm greatly relies on a good initial point and the learning step. To enhance the experiment, we take different M and N to compare AHH and FAHH.

TABLE III

PERFORMANCE OF $\mathrm{RSSE}(T)$ WITH $M = 50$ AND $N = 2000$.

	FAHH	AHH	HH	GHH
y_1	**0.0050**(2.23)	0.0061(10.24)	0.1290(14.61)	0.1173(24.15)
y_2	**0.1807**(2.91)	0.1819(11.85)	0.2131(2.78)	0.2118(16.86)

TABLE IV

PERFORMANCE (FAHH/AHH) WITH DIFFERENT M AND N.

	$M=30$	$M=40$	$M=50$
	RSSE		
y_1	**0.0059**/0.0065	**0.0053**/0.0064	**0.0050**/0.0061
y_2	**0.1942**/0.1981	**0.1822**/0.1903	**0.1807**/0.1819
	T		
y_1	1.21/2.56	1.62/5.51	2.23/10.24
y_2	1.38/2.92	2.25/6.36	2.91/11.85
	$N=500$	$N=1000$	$N=2000$
	RSSE		
y_1	**0.0149**/0.0198	**0.0078**/0.0122	**0.0053**/0.0064
y_2	**0.2218**/0.2661	**0.1967**/0.2090	**0.1822**/0.1903
	T		
y_1	0.39/3.17	0.75/4.28	1.62/5.51
y_2	0.42/3.38	0.95/4.64	2.25/6.36

Table IV shows that the efficiency of FAHH becomes more significant with M increasing. It also illustrates that the accuracy advantage of the proposed FAHH algorithm is more evident with less training data .

C. Application to dynamic system identification

This section compares the performance of the algorithms in dynamic system identification with benchmark nonlinear dynamic systems of NARX model [16] and the real data from coupled electric drive data set [19], [20].

System 1: Consider the nonlinear system [5], [17], [18]

$$y(t) = \frac{y(t-1)y(t-2)y(t-3)u(t-2)(y(t-3)-1)}{1+y^2(t-2)+y^2(t-3)} + \frac{u(t-1)}{1+y^2(t-2)+y^2(t-1)}.$$

The system is excited by a random signal $u(t)$ $(1 \le t \le 800)$ uniformly distributed in $[-1,1]$ with a noise from $\mathcal{N}(0, 0.05^2)$. The regression vector is set as $\mathbf{x} = [y(t-1)\ y(t-2)\ y(t-3)\ u(t-1)\ u(t-2)]^T$. We set $M = 50$, $k = 30$, $h = 1$ and $w = 100$. Fig 2 shows the simulated outputs of the round at which AHH and FAHH both achieve preferable estimation.

In some of the tests, AHH shows instability in prediction, where the simulated output fluctuates heavily in some regions. A more comprehensive comparison is the conducted

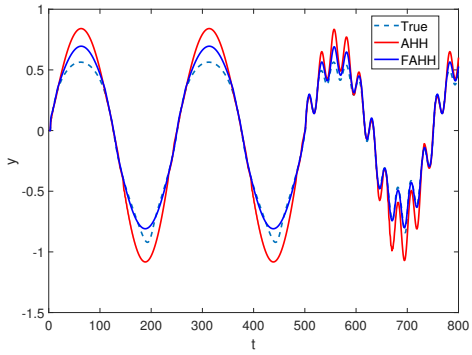

Fig. 2. For AHH and FAHH, the RSSE is 0.0895 and 0.0132 respectively, while the running time T is 3.45s and 0.66s.

with 300 repeats in different settings. FAHH maintains stability but AHH has 5 unstable runs, which mainly owns to the introduction of Lasso regularization.

System 2: The CE8 coupled electric drives [19] consists of two electric motors that drive a pulley using a flexible belt. We use the uniform data from [20], which consists of 500 evaluations. We train FAHH model with the first 300 evaluations and test it with the rest data. The performance is shown in Fig 3, where $M = 10$ and $w = 80$. We choose $\mathbf{x} = [y(t-1) \cdots y(t-5)\ u(t-1) \cdots u(t-5)]^T$ as the regression vector. When we reset $M = 15$, FAHH still outperforms AHH, where the corresponding RSSE is 0.0034 and 0.0042, and running time T is 0.23s and0.81s for FAHH and AHH respectively. Another point to be noticed is that AHH and FAHH can select the variables, which help to explore proper regression vector. In this test, it shows that only with $\mathbf{x} = [y(t-1)\ y(t-2); y(t-3)\ u(t-3)\ u(t-4)\ u(t-5)]^T$ FAHH and AHH still achieve similar results, where the RSSE is 0.0034 and 0.0047 respectively.

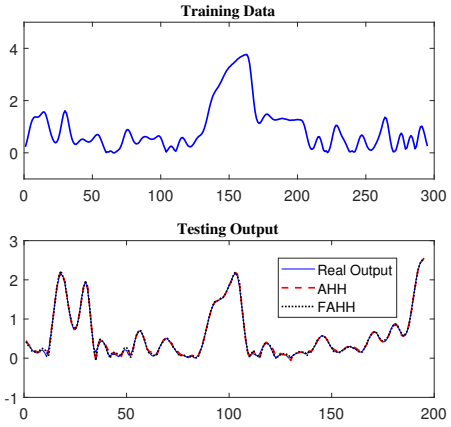

Fig. 3. For AHH and FAHH, the RSSE is 0.0046 and 0.0041, while the running time T is 0.26s and 0.06s respectively.

V. CONCLUSIONS

AHH is a popular CWPA model consisting of a linear combination of basis functions, which are incrementally generated by the complete traverse in forward procedure combined with backward deletion. In this paper, we propose FAHH algorithm to accelerate the traverse in the forward procedure by skipping the potentially redundant searches with a heuristic strategy, while the backward deletion is replaced with Lasso regularization to further improve efficiency and guarantee accuracy with robustness. Besides, the data-based selection of the splitting knots is also discussed. Numerical experiments verify that the proposed FAHH algorithm significantly improves the efficiency of existing AHH with higher accuracy and it also enhances robustness.

REFERENCES

[1] X. Huang, J. Xu, and S. Wang, "Nonlinear system identification with continuous piecewise linear neural network", *Neurocomputing*, vol. 77, no. 1, pp. 167-177, 2012.

[2] L. Breiman, "Hinging hyperplanes for regression, classification, and function approximation", *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 999-1013, 1993.

[3] P. Julian, A. Desages, and O. Agamennoni, "High-level canonical piecewise linear representation using a simplicial partition", *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 46, no. 4, pp. 463-480, 1999.

[4] S. Wang and X. Sun, "Generalization of hinging hyperplanes", *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4425-4431, 2005.

[5] J. Xu, X. Huang, and S. Wang, "Adaptive hinging hyperplanes and its applications in dynamic system identification", *Automatica*, vol. 45, no. 10, pp. 2325-2332, 2009.

[6] X. Huang, S. Mehrkanoon, and J. A. K. Suykens, "Support vector machines with piecewise linear feature mapping", *Neurocomputing*, vol. 117, pp. 118-127, 2013.

[7] J. Xu, X. Huang, X. Mu, and S. Wang, "Model predictive control based on adaptive hinging hyperplanes model", *Journal of Process Control*, vol. 22, no. 10, pp. 1821-1831, 2012.

[8] J. H. Friedman, "Multivariate adaptive regression splines", *The annals of statistics*, pp.1-67, 1991.

[9] P. Craven and G. Wahba, "Smoothing noisy data with spline functions", *Numerische Mathematik*, vol. 31, pp. 377-403, 1979.

[10] J. H. Friedman and B. W. Silverman, "Flexible parsimonious smoothing and additive modeling", *Technometrics*, vol. 31, no. 1, pp. 3-21, 1989.

[11] J. H. Friedman, "Fast MARS", Technical Report, Stanford University, 1993.

[12] R. Tibshirani, "Regression shrinkage and selection via the lasso", *Journal of the Royal Statistical Society*, Series B (Methodological), pp. 267-288, 1996.

[13] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers", *Foundations and Trends in Machine learning*, vol. 3, no. 1, pp. 1-122, 2011.

[14] N. Parikh and S. Boyd, "Proximal algorithms", *Foundations and Trends in Optimization*, vol. 1, no. 3, pp. 127-239, 2014.

[15] V. Cherkassky, D. Gehring and F. Mulier, "Comparison of adaptive methods for function estimation from samples", *IEEE Transactions on Neural Networks*, vol. 7, no. 4, pp. 969-984, 1996.

[16] J. Sjöberg, Q Zhang, L. Ljung, et al, "Nonlinear black-box modeling in system identification", *Automatica*, vol. 31, no. 12, pp. 1671-1724, 1995.

[17] K. S. Narendra, and K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks", *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4-27, 1990.

[18] J. Roll, A. Nazin and L. Ljung, "Nonlinear system identification via direct weight optimization", *Automatica*, vol. 41, no. 3, pp. 475-490, 2005.

[19] P. E. Wellstead, "Introduction to physical system modeling", Academic Press, London, UK, 1979.

[20] T. Wigren and M. Schoukens, "Coupled Electric Drives Data Set and Reference Models", Technical Report, Department of Information Technology, Uppsala University, 2017.