Block building programming for symbolic regression

Chen Chen [a,b], Changtong Luo [a,*], Zonglin Jiang [a,b]

[a] *State Key Laboratory of High Temperature Gas Dynamics, Institute of Mechanics, Chinese Academy of Sciences, Beijing 100190, China*
[b] *School of Engineering Sciences, University of Chinese Academy of Sciences, Beijing 100049, China*

ARTICLE INFO

ABSTRACT

Symbolic regression that aims to detect underlying data-driven models has become increasingly important for industrial data analysis. For most existing algorithms such as genetic programming (GP), the convergence speed might be too slow for large-scale problems with a large number of variables. This situation may become even worse with increasing problem size. The aforementioned difficulty makes symbolic regression limited in practical applications. Fortunately, in many engineering problems, the independent variables in target models are separable or partially separable. This feature inspires us to develop a new approach, block building programming (BBP). BBP divides the original target function into several blocks, and further into factors. The factors are then modeled by an optimization engine (e.g. GP). Under such circumstances, BBP can make large reductions to the search space. The partition of separability is based on a special method, block and factor detection. Two different optimization engines are applied to test the performance of BBP on a set of symbolic regression problems. Numerical results show that BBP has a good capability of structure and coefficient optimization with high computational efficiency.

© 2017 Elsevier B.V. All rights reserved.

1. Introduction

Data-driven modeling of complex systems has become increasingly important for industrial data analysis when the experimental model structure is unknown or wrong, or the concerned system has changed [1,2]. Symbolic regression aims to find a data-driven model that can describe a given system based on observed input-response data, and plays an important role in different areas of engineering such as signal processing [3], system identification [4], industrial data analysis [5], and industrial design [6]. Unlike conventional regression methods that require a mathematical model of a given form, symbolic regression is a data-driven approach to extract an appropriate model from a space of all possible expressions S defined by a set of given binary operations (e.g. $+$, $-$, $\times$, $\div$) and mathematical functions (e.g. $\sin$, $\cos$, $\exp$, $\ln$), which can be described as follows:

$$f^* = \arg\min_{f \in S} \sum_i \left\| f(\mathbf{x}^{(i)}) - y_i \right\|, \qquad (1)$$

where $\mathbf{x}^{(i)} \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$ are sampling data. f is the target model and f^* is the data-driven model. Symbolic regression is a kind of non-deterministic polynomial (NP) problem, which simultaneously optimizes the structure and coefficient of a target model.

How to use an appropriate method to solve a symbolic regression problem is considered as a kaleidoscope in this research field [7–9].

Genetic programming (GP) [10] is a classical method for symbolic regression. The core idea of GP is to apply Darwin's theory of natural evolution to the artificial world of computers and modeling. Theoretically, GP can obtain accurate results, provided that the computation time is long enough. However, describing a large-scale target model with a large number of variables is still a challenging task. This situation may become even worse with increasing problem size (increasing number of independent variables and range of these variables). This is because the target model with a large number of variables may result in large search depth and high computational costs of GP. The convergence speed of GP may then be too slow. This makes GP very inconvenient in engineering applications.

Apart from basic GP, two groups of methods for symbolic regression have been studied. The first group focused on evolutionary strategy, such as grammatical evolution [11] and parse-matrix evolution [12]. These variants of GP can simplify the coding process. Gan et al. [13] introduced a clone selection programming method based on an artificial immune system. Karaboga et al. [14] proposed an artificial bee colony programming method based on the foraging behavior of honeybees. However, these methods are still based on the idea of biological simulation processes. This helps little to improve the convergence speed when solving large-scale problems.

* Corresponding author.
E-mail addresses: chenchen@imech.ac.cn (C. Chen), luo@imech.ac.cn (C. Luo), zljiang@imech.ac.cn (Z. Jiang).

The second branch exploited strategies to reduce the search space of the solution. McConaghy [15] presented the first non-evolutionary algorithm, fast function eXtraction (FFX), based on pathwise regularized learning, which confined its search space to generalized linear space. However, the computational efficiency is gained at the sacrifice of losing the generality of the solution. More recently, Worm [16] proposed a deterministic machine-learning algorithm, prioritized grammar enumeration (PGE). PGE merges isomorphic chromosome presentations (equations) into a canonical form. The author argues that it could make a large reduction to the search space. However, debate still remains on how the simplification affects the solving process [17–19].

In many scientific or engineering problems, the target models are separable. Luo et al. [20] presented a divide-and-conquer (D&C) method for GP. The authors indicated that detecting the correlation between each variable and the target function could accelerate the solving process. D&C can decompose a concerned separable model into a number of sub-models, and then optimize them. The separability is probed by a special method, the bi-correlation test (BiCT). However, the D&C method can only be valid for an additively/multiplicatively separable target model (see Definition 1 in Section 2). Many practical models are out of the scope of the separable model (Eqs. (6) and (7)). This limits the D&C method for further applications.

In this paper, a more general separable model that may involve mixed binary operators, namely plus ($+$), minus ($-$), times ($\times$), and division ($\div$), is introduced. In order to get the structure of the generalized separable model, a new approach, block building programming (BBP), for symbolic regression is also proposed. BBP reveals the target separable model using a block and factor detection process, which divides the original model into a number of blocks, and further into factors. Meanwhile, binary operators could also be determined. The method can be considered as a bi-level D&C method. The separability is detected by a generalized BiCT method. Numerical results show that BBP can obtain the target functions more reliably, and produce extremely large accelerations of the GP method for symbolic regression.

The presentation of this paper is organized as follows. Section 2 is devoted to the more general separable model. The principle and procedure of the BPP approach are described in Section 3. Section 4 presents numerical results, discussions, and efficiency analysis for the proposed method. In the last section, conclusions are drawn with future works.

2. Definition of separability

2.1. Examples

As previously mentioned, in many applications, the target models are separable. Below, two real-world problems are given to illustrate separability.

Example 1. When developing a rocket engine, it is crucial to model the internal flow of a high-speed compressible gas through the nozzle. The closed-form expression for the mass flow through a choked nozzle [21] is

$$\dot{m} = \frac{p_0 A^*}{\sqrt{T_0}} \sqrt{\frac{\gamma}{R} \left(\frac{2}{\gamma + 1}\right)^{(\gamma+1)/(\gamma-1)}}, \tag{2}$$

where p_0 and T_0 represent the total pressure and total temperature, respectively. A^* is the sonic throat area. R is the specific gas constant, which is a different value for different gases. $\gamma = c_p/c_v$, where c_v and c_p are the specific heat at constant volume and constant pressure. The sub-functions of the five independent variables, p_0, T_0, A^*, R, and γ are all multiplicatively separable in Eq. (2). That

is, the target model can be re-expressed as follows

$$\begin{aligned}\dot{m} &= f(p_0, A^*, T_0, R, \gamma) \\ &= f_1(p_0) \times f_2(A^*) \times f_3(T_0) \times f_4(R) \times f_5(\gamma).\end{aligned} \tag{3}$$

The target function with five independent variables can be divided into five sub-functions that are multiplied together, and each with only one independent variable. Furthermore, the binary operator between two sub-functions could be plus ($+$) or times ($\times$).

Example 2. In aircraft design, the lift coefficient of an entire airplane [22] can be expressed as

$$C_L = C_{L\alpha}(\alpha - \alpha_0) + C_{L\delta_e} \delta_e \frac{S_{\mathrm{HT}}}{S_{\mathrm{ref}}}, \tag{4}$$

where $C_{L\alpha}$ and $C_{L\delta_e}$ are the lift slope of the body wings and tail wings. α, α_0, and δ_e are the angle of attack, zero-lift angle of attack, and deflection angle of the tail wing, respectively. S_{HT} and S_{ref} are the tail wing area and reference area, respectively. Note that the sub-functions of the variable $C_{L\alpha}$, $C_{L\delta_e}$, δ_e, S_{HT}, and S_{ref} are separable, but not purely additively/multiplicatively separable. Variables α and α_0 are not separable, but their combination (α, α_0) can be considered separable. Hence, Eq. (4) can be re-expressed as

$$\begin{aligned}C_L &= f\left(C_{L\alpha}, \alpha, \alpha_0, C_{L\delta_e}, \delta_e, S_{\mathrm{HT}}, S_{\mathrm{ref}}\right) \\ &= f_1(C_{L\alpha}) \times f_2(\alpha, \alpha_0) + f_3\left(C_{L\delta_e}\right) \times f_4(\delta_e) \times f_5(S_{\mathrm{HT}}) \times f_6(S_{\mathrm{ref}}).\end{aligned} \tag{5}$$

In this example, the target function is divided into six sub-functions.

2.2. Additively/multiplicatively separable model

The additively and multiplicatively separable models introduced in [20] are briefly reviewed below.

Definition 1. A scalar function $f(\mathbf{x})$ with n continuous variables $\mathbf{x} = [x_1, x_2, \ldots, x_n]^\top$ ($f : \mathbb{R}^n \mapsto \mathbb{R}$, $\mathbf{x} \in \mathbb{R}^n$) is additively separable if and only if it can be rewritten as

$$f(\mathbf{x}) = \alpha_0 + \sum_{i=1}^m \alpha_i f_i\left(\mathbf{I}^{(i)}\mathbf{x}\right), \tag{6}$$

and is multiplicatively separable if and only if it can be rewritten as

$$f(\mathbf{x}) = \alpha_0 \cdot \prod_{i=1}^m f_i\left(\mathbf{I}^{(i)}\mathbf{x}\right). \tag{7}$$

In Eqs. (6) and (7), $\mathbf{I}^{(i)} \in \mathbb{R}^{n_i \times n}$ is the partitioned matrix of the identity matrix $\mathbf{I} \in \mathbb{R}^{n \times n}$, namely $\mathbf{I} = \begin{bmatrix} \mathbf{I}^{(1)} & \mathbf{I}^{(2)} & \ldots & \mathbf{I}^{(m)} \end{bmatrix}^\top$, $\sum_{i=1}^m n_i = n$. $\mathbf{I}^{(i)}\mathbf{x}$ is the variables set with n_i elements. n_i represents the number of variables in sub-function f_i. Sub-function f_i is a scalar function such that $f_i : \mathbb{R}^{n_i} \mapsto \mathbb{R}$. α_i is a constant coefficient.

2.3. Partially/completely separable model

Based on the definition of additive/multiplicative separability, the new separable model with mixed binary operators, namely plus ($+$), minus ($-$), times ($\times$), and division ($\div$) are defined as follows.

Definition 2. A scalar function $f(\mathbf{x})$ with n continuous variables $\mathbf{x} = [x_1, x_2, \ldots, x_n]^\top$ ($f : \mathbb{R}^n \mapsto \mathbb{R}$, $\mathbf{x} \in \mathbb{R}^n$) is partially separable if and only if it can be rewritten as

$$f(\mathbf{x}) = \alpha_0 \otimes_1 \alpha_1 f_1\left(\mathbf{I}^{(1)}\mathbf{x}\right) \otimes_2 \alpha_2 f_2\left(\mathbf{I}^{(2)}\mathbf{x}\right) \otimes_3 \ldots \otimes_m \alpha_m f_m\left(\mathbf{I}^{(m)}\mathbf{x}\right), \tag{8}$$

where $\mathbf{I}^{(i)} \in \mathbb{R}^{n_i \times n}$ is the partitioned matrix of the identity matrix $\mathbf{I} \in \mathbb{R}^{n \times n}$, namely $\mathbf{I} = \begin{bmatrix} \mathbf{I}^{(1)} & \mathbf{I}^{(2)} & \cdots & \mathbf{I}^{(m)} \end{bmatrix}^\top$, $\sum_{i=1}^{m} n_i = n$. $\mathbf{I}^{(i)}\mathbf{x}$ is the variables set with n_i elements. n_i represents the number of variables in sub-function f_i. Sub-function f_i is a scalar function such that $f_i : \mathbb{R}^{n_i} \mapsto \mathbb{R}$. The binary operator $\otimes_i$ can be plus $(+)$ and times $(\times)$. α_i is a constant coefficient.

Note that the binary operators minus $(-)$ and division $(/)$ are not included in $\otimes$ for simplicity. This does not affect much of its generality, since minus $(-)$ could be regarded as $(-) = (-1) \cdot (+)$, and sub-function could be treated as $\tilde{f}_i(\cdot) = 1/f_i(\cdot)$ if only $f_i(\cdot) \neq 0$.

Definition 3. A scalar function $f(\mathbf{x})$ with n continuous variables ($f : \mathbb{R}^n \mapsto \mathbb{R}$, $\mathbf{x} \in \mathbb{R}^n$) is completely separable if and only if it can be rewritten as Eq. (8) and $n_i = 1$ for all $i = 1, 2, \ldots, m$.

3. Block building programming

3.1. Bi-correlation test

The bi-correlation test (BiCT) method proposed in [20] is used to detect whether a concerned target model is additively or multiplicatively separable. BiCT is based on random sampling and the linear correlation method.

3.2. Block and factor detection

The additively or multiplicatively separable target function can be easily detected by the BiCT. However, how to determine each binary operator $\otimes_i$ of Eq. (8) is a critical step in BBP. One way is to recognize each binary operator $\otimes_i$ sequentially with random sampling and linear correlation techniques. For example, a given target function of six variables with five sub-functions is given below

$$f(x_1, \ldots, x_6) = \alpha_0 \otimes_1 \alpha_1 f_1(x_1) \otimes_2 \alpha_2 f_2(x_2, x_3) \otimes_3 \alpha_3 f_3(x_4)$$
$$\otimes_4 \alpha_4 f_4(x_5) \otimes_5 \alpha_5 f_5(x_6)$$
$$= \alpha_0 + \alpha_1 f_1(x_1) \times \alpha_2 f_2(x_2, x_3) + \alpha_3 f_3(x_4)$$
$$+ \alpha_4 f_4(x_5) \times \alpha_5 f_5(x_6). \tag{9}$$

The first step is to determine the binary operator $\otimes_1$. The six variables are sampled with the variable x_1 changed, and the remaining variables $x_2, x_3, \ldots, x_6$ fixed. However, it is found that the variable x_1 cannot be separable from the variables $x_2, x_3, \ldots, x_6$, since the operation order of the two binary operators plus $(+)$ and times $(\times)$ is different. This indicates that recognizing each $\otimes_i$ sequentially is difficult.

To overcome the aforementioned difficulty, block and factor detection is introduced, which helps to recognize the binary operator $\otimes_i$ more effectively. Before introducing this method, a theorem is given as follows.

Theorem 1 *Eq. (8). can be equivalently written as*

$$f(\mathbf{x}) = \beta_0 + \sum_{i=1}^{p} \beta_i \varphi_i\left(\mathbf{I}^{(i)}\mathbf{x}\right) = \beta_0 + \sum_{i=1}^{p} \beta_i \prod_{j=1}^{q_i} \psi_{i,j}\left(\mathbf{I}_j^{(i)}\mathbf{x}\right), \tag{10}$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_n]^\top \in \mathbb{R}^n$ and $f : \mathbb{R}^n \mapsto \mathbb{R}$. $\mathbf{I}^{(i)} \in \mathbb{R}^{s_i \times n}$ is the partitioned matrix of the identity matrix $\mathbf{I} \in \mathbb{R}^{n \times n}$, namely $\mathbf{I} = \begin{bmatrix} \mathbf{I}^{(1)} & \mathbf{I}^{(2)} & \cdots & \mathbf{I}^{(p)} \end{bmatrix}^\top$, $\sum_{i=1}^{p} s_i = n$. $\mathbf{I}_j^{(i)} \in \mathbb{R}^{s_{i,j} \times n}$ is the partitioned matrix of the $\mathbf{I}^{(i)}$, namely $\mathbf{I}^{(i)} = \begin{bmatrix} \mathbf{I}_1^{(i)} & \mathbf{I}_2^{(i)} & \cdots & \mathbf{I}_{q_i}^{(i)} \end{bmatrix}^\top$, $\sum_{j=1}^{q_i} s_{i,j} = s_i$. $p \geq 1$, $q_i \geq 1$, $\sum_{i=1}^{p} q_i = m$. Sub-functions φ_i and $\psi_{i,j}$ are scalar functions such that $\varphi_i : \mathbb{R}^{s_i} \mapsto \mathbb{R}$ and $\psi_{i,j} : \mathbb{R}^{s_{i,j}} \mapsto \mathbb{R}$.

Proof. See Appendix A. □

Definition 4. The sub-function $\varphi_i(\mathbf{I}^{(i)}\mathbf{x})$ is the ith block of Eq. (10), and the sub-function $\psi_{i,j}\left(\mathbf{I}_j^{(i)}\mathbf{x}\right)$ is the jth factor of the ith block.

It is observed from Eq. (9) that there are three blocks, $\varphi_1(x_1, x_2, x_3)$, $\varphi_2(x_4)$, and $\varphi_3(x_5, x_6)$. The structure of Eq. (9) is given as the following equation, based on the Theorem 1,

$$f(x_1, \ldots, x_6) = \alpha_0 + \alpha_1 f_1(x_1) \times \alpha_2 f_2(x_2, x_3) + \alpha_3 f_3(x_4)$$
$$+ \alpha_4 f_4(x_5) \times \alpha_5 f_5(x_6)$$
$$= \beta_0 + \beta_1 \varphi_1(x_1, x_2, x_3) + \beta_2 \varphi_2(x_4) + \beta_3 \varphi_3(x_5, x_6)$$
$$= \beta_0 + \beta_1 \boxed{\psi_{1,1}(x_1) \times \psi_{1,2}(x_2, x_3)} + \beta_2 \boxed{\psi_{2,1}(x_4)}$$
$$+ \beta_3 \boxed{\psi_{3,1}(x_5) \times \psi_{3,2}(x_6)}. \tag{11}$$

The first block has two factors, $\psi_{1,1}(x_1)$ and $\psi_{1,2}(x_2, x_3)$. The second block has only one factor, $\psi_{2,1}(x_4)$. The last block also has two factors, $\psi_{3,1}(x_5)$ and $\psi_{3,2}(x_6)$. Note that variables x_2 and x_3 are partially separable, while variables x_1, x_4, x_5, and x_6 are all completely separable.

From the previous discussion, it is straightforward to show that in Eq. (10), the original target function $f(\mathbf{x})$ is first divided into several blocks $\varphi_i(\cdot)$ with global constants c_i. Meanwhile, all binary plus $(+)$ operators are determined, which is based on the separability detection of the additively separable $\varphi_i(\cdot)$ by BiCT, where $i = 1, 2, \ldots, p$. Next, each block $\varphi_i(\cdot)$ is divided into several factors $\psi_{i,j}(\cdot)$. Meanwhile, all binary times $(\times)$ operators are determined, which is based on the separability detection of multiplicatively separable $\psi_{i,j}(\cdot)$ by BiCT method, where $j = 1, 2, \ldots, q_i$. It is clear that the process of block and factor detection does not require any special optimization engine.

3.3. Factor modeling

The mission of symbolic regression is to optimize both the structure and coefficient of a target function that describes an input-response system. In block building programming (BBP), after the binary operators are determined, the original target function $f(\mathbf{x})$ is divided into several factors $\psi_{i,j}(\cdot)$. In this section, we aim to find a proper way to model these factors. This problem is quite easy to be solved by an optimization algorithm, since the structure and coefficient of a factor $\psi_{i,j}(\cdot)$ can be optimized while the rest are kept fixed and unchanged.

Without the loss of generality, the factor $\psi_{1,1}\left(x_1, x_2, \ldots, x_{s_{1,1}}\right)$ in Eq. (10) illustrates the implementation of the modeling process.

1. Let the matrix $\mathbf{X}$ be a set of N sampling points for all n independent variables,

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1,n} \\ x_{21} & x_{22} & \cdots & x_{2,n} \\ \vdots & \vdots & & \vdots \\ x_{N,1} & x_{N,2} & \cdots & x_{N,n} \end{bmatrix}, \tag{12}$$

where $x_{i,j}$ represents the ith sampling point of the jth independent variable, $i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, N$.

2. Keep variables $x_1, x_2, \ldots, x_{s_{1,1}}$ being randomly sampled. Let the sampling points of the variables in local block (block 1), $x_{s_{1,1}+1}, x_{s_{1,1}+2}, \ldots, x_{s_1}$, be fixed to any two given points x_A and x_B ($\forall x_A, x_B \in [a, b]$), respectively. In addition, let the sampling points of variables in other blocks (blocks 2 to p), namely $x_{s_1+1}, x_{s_1+2}, \ldots, x_n$, be fixed to a given point x_G ($\forall x_G \in [a, b]$). We obtain

$$\mathbf{X}_1 = \begin{bmatrix} x_{1,1} & \cdots & x_{1,s_{1,1}} & x_{1,s_{1,1}+1}^{(A)} & \cdots & x_{1,s_1}^{(A)} & x_{1,s_1+1}^{(G)} & \cdots & x_{1,n}^{(G)} \\ x_{2,2} & \cdots & x_{2,s_{1,1}} & x_{2,s_{1,1}+1}^{(A)} & \cdots & x_{2,s_1}^{(A)} & x_{2,s_1+1}^{(G)} & \cdots & x_{2,n}^{(G)} \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ x_{N,1} & \cdots & x_{N,s_{1,1}} & x_{N,s_{1,1}+1}^{(A)} & \cdots & x_{N,s_1}^{(A)} & x_{N,s_1+1}^{(G)} & \cdots & x_{2,n}^{(G)} \end{bmatrix}, \tag{13}$$

and

$$\mathbf{X}_2 = \begin{bmatrix} x_{1,1} & \cdots & x_{1,s_{1,1}} & x_{1,s_{1,1}+1}^{(B)} & \cdots & x_{1,s_1}^{(B)} & x_{1,s_1+1}^{(G)} & \cdots & x_{1,n}^{(G)} \\ x_{2,2} & \cdots & x_{2,s_{1,1}} & x_{2,s_{1,1}+1}^{(B)} & \cdots & x_{2,s_1}^{(B)} & x_{2,s_1+1}^{(G)} & \cdots & x_{2,n}^{(G)} \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ x_{N,1} & \cdots & x_{N,s_{1,1}} & x_{N,s_{1,1}+1}^{(B)} & \cdots & x_{N,s_1}^{(B)} & x_{N,s_1+1}^{(G)} & \cdots & x_{2,n}^{(G)} \end{bmatrix}.$$

(14)

3. Let $\tilde{\mathbf{X}} = \mathbf{X}_1 - \mathbf{X}_2 = \begin{bmatrix} \mathbf{X}_{\text{train}} & \mathbf{0} \end{bmatrix}$. Matrix $\mathbf{X}_{\text{train}}$ is a partition of matrix $\tilde{\mathbf{X}}$. Next, let $\mathbf{f}_{\text{train}}$ be the vector of which the ith element is the function value of the ith row of matrix $\tilde{\mathbf{X}}$, namely $\mathbf{f}_{\text{train}} = f(\tilde{\mathbf{X}})$.

4. Substitute $\mathbf{f}_{\text{train}}$ and $\mathbf{X}_{\text{train}}$ into the fit models $y_{\text{train}} = \beta \cdot f^*(x_{\text{train}})$. This step could be realized by an existing optimization engine (e.g. GP). Note that, the constant β represent the fitting parameter of the function of variables $x_{s_{1,1}+1}, x_{s_{1,1}+2}, \ldots, x_{s_1}$, since these variables are unchanged during this process. We aim to obtain the optimization model f^*, and constant β will be discarded.

Other factors $\psi_{i,j}$ could be obtained in the same way. In fact, many state-of-the-art optimization engines are valid for BBP. Genetic programming methods (e.g. parse-matrix evolution (PME) [12] and GPTIPS [23]), swarm intelligence methods (e.g. artificial bee colony programming (ABCP) [14]), and global optimization methods (e.g. low-dimensional simplex evolution (LDSE) [24]) are all easy to power BBP.

3.4. Block building programming

Block-building programming (BBP) can be considered as a bi-level D&C method, that is, the separability detection involves two processes (block and factor detection). In fact, BBP provides a framework of genetic programming methods or global optimization algorithms for symbolic regression. The main process of BBP is decomposed and presented in previous sections (Sections 3.1–3.3). Despite different optimization engines for factors modeling being used, the general procedure of BBP could be described as follows.

Procedure of BBP:

Step 1. (Initialization) Input the dimension of the target function D, the set $S = \{i : i = 1, 2, \ldots, D\}$ for initial variables subscript number, sampling interval $[a, b]$, and the number of sampling points N. Generate a sampling set $\mathbf{X} \in [a, b] \subset \mathbb{R}^{N \times D}$.

Step 2. (Block detection) The information (the subscript number of the local block and variables) of each block $\varphi_i(\cdot)$, $i = 1, 2, \ldots, p$, is detected and preserved iteratively by BiCT (that is, the additively separable block).

Step 3. (Factor detection) For each block $\varphi_i(\cdot)$, the information (the subscript number of the local block, factor, and variables) of each factor $\psi_{i,j}(\cdot)$, $j = 1, 2, \ldots, q_i$, is detected and preserved iteratively by BiCT (that is, the multiplicatively separable factor in local block).

Step 4. (Factor modeling) For the jth factor $\psi_{i,j}(\cdot)$ in the ith block, set the variables in sampling set $\mathbf{X}$ in blocks $\{1, 2, \ldots, i - 1, i + 1, \ldots, p\}$ to be fixed to x_G, and the variables in factors $\{1, 2, \ldots, j - 1, j + 1, \ldots, q_j\}$ of the ith block to be fixed to x_A and x_B. Let $\tilde{\mathbf{X}}^{i,j} = \mathbf{X}_1^{i,j} - \mathbf{X}_2^{i,j} = \begin{bmatrix} \mathbf{X}_{\text{train}}^{i,j} & \mathbf{0} \end{bmatrix}$ and $\mathbf{f}_{\text{train}}^{i,j} = f(\tilde{\mathbf{X}}^{i,j})$. The optimization engine is then used.

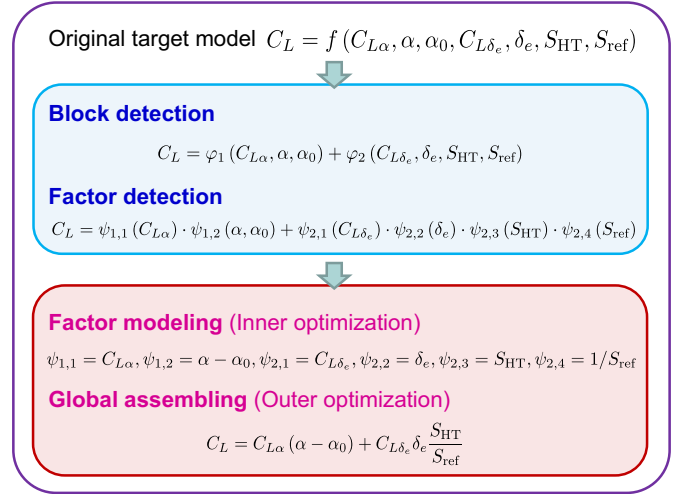Step 5. (Global assembling) Global parameter β_k, $k = 0, 1, \ldots, p$, can be linearly fitted by the equation $\mathbf{f}_{\text{train}} = \beta_0 + \sum_{i=1}^{p} \beta_i \varphi_i(\cdot) = \beta_0 + \sum_{i=1}^{p} \beta_i \prod_{j=1}^{q_i} \psi_{i,j}(\mathbf{X}_{\text{train}})$.

Fig. 1. An example procedure of BBP: the modeling of the Eq. (4).

It is clear from the above procedure that the optimization process of BBP could be divided into two parts, inner and outer optimization. The inner optimization (e.g. LDSE and GPTIPS) is invoked to optimize the structure and coefficients of each factor, with the variables of other factors being fixed. The outer optimization aims to optimize the global parameters of the target model structure. An example procedure of BBP is provided in Fig. 1.

3.5. Optimization engine

Factor modeling could be easily realized by global optimization algorithms or genetic programming. However, a few differences between these two methods should be considered.

In BBP, when using a global optimization method, the structure of the factors (function models) should be pre-established. For instance, functions that involve uni-variable and bi-variables should be set for function models. Sequence search and optimization methods is suitable for global optimization strategies. This means a certain function model will be determined, provided that the fitting error is small enough (e.g. mean square error $\leqslant 10^{-8}$).

In genetic programming, arithmetic operations (e.g. $+$, $-$, $\times$, $\div$) and mathematical functions (e.g. $\sin$, $\cos$, $\exp$, $\ln$) should be pre-established instead of function models. The search process of GP is stochastic. This makes it easily reach premature convergence. In Section 5, LDSE and GPTIPS, are both chosen to test the performance of BBP.

The choice of LDSE and GPTIPS as the optimization engine is based on the obvious fact that the factor modeling might contain the non-convex optimization process (e.g. $\psi = m_1 \sin(m_2 x + m_3)$). LDSE and GPTIPS can make our program easy to use, but may have an additional calculation cost. To further speed up the optimization process, the convex optimization algorithm [25–27] is valid provided that the factor modeling could be reduced to a linear programming, quadratic programming, and semi-definite programming problem (e.g. $\psi = m_1 x^{m_2} + m_3$).

3.6. Remarks for engineering applications

The proposed method is described with functions of explicit expressions. For practical applications, data-driven modeling problems are more common, which means the explicit expression might be unavailable. To use the proposed BBP method, one can construct a surrogate model of the black-box type to represent the underlying target function [28], and the surrogate model could be

Table 1
Uni-variable and bi-variables preseted models.

No.	Uni-variable model	Bi-variables model
1	$k(x^{m_1} + m_2)$	$k(m_1 x_1 + m_2 x_2 + m_3)$
2	$k(e^{m_1 x} + m_2)$	$k[(m_1 x_1 + m_2)/(m_3 x_2 + m_4)]$
3	$k \sin(m_1 x^{m_2} + m_3)$	$k(e^{m_1 x_1 x_2} + m_2)$
4	$k \log(m_1 x + m_2)$	$k \sin(m_1 x_1 + m_2 x_2 + m_3 x_1 x_2 + m_4)$

Table 2
Performance of LDSE-powered BBP.

Case No.	Dim	Domain	No. samples	T_d/T_{BBP}(%)
1	3	$[-3,3]^3$	300	4.35
2	3	$[-3,3]^3$	300	2.38
3	4	$[-3,3]^4$	400	4.7
4	4	$[-3,3]^4$	400	2.51
5	4	$[-3,3]^4$	400	3.32
6	5	$[1,4]^5$	300	1.98
7	5	$[-3,3]^5$	500	4.42
8	5	$[-3,3]^5$	500	1.68
9	6	$[-3,3]^6$	600	2.86
10	6	$[-3,3]^6$	600	3.38

used as the target model in the BBP method. The other modeling steps are the same as described above.

Data noise is another issue to consider for applying BBP. In practical applications, the input-response data usually involves noises such as measure errors and/or system noises. Note that during the above surrogate modeling (e.g. Kriging regression [29]) process, some noises could be suppressed. Meanwhile, the exact BiCT is now unnecessary, since the surrogate model is also not the true target function. We can use a special technique called Soft BiCT, where $|\rho|$ is set to $|\rho| = 1 - \varepsilon$ (ε is a small positive number) instead of $|\rho| = 1$ in BiCT. Multiple BiCT could also further suppress the noises, where each variable is fixed to more pairs of vectors (one pair in BiCT). Detailed discussions will be provided in future studies.

4. Numerical results and discussion

The proposed BBP is implemented in Matlab/Octave. In order to test the performance of BBP, two different optimization engines, LDSE [24] and GPTIPS [23], are used. For ease of use, a Boolean variable is used on the two selected methods. Numerical experiments on 10 cases of completely separable or partially separable target functions, as given in Appendix B, are conducted. These cases help evaluate BBP's overall capability of structure and coefficient optimization. Computational efficiency is analyzed in Section 4.3.

4.1. LDSE-powered BBP

We choose a global optimization algorithm, LDSE [24], as our optimization engine. LDSE is a hybrid evolutionary algorithm for continuous global optimization. In Table 2, case number, dimension, domain, and number of sampling points are denoted as No, Dim, Domain, and No. samples, respectively. Additionally, we record the time T_d (the block and factor detection), and T_{BBP} (the entire computation time of BBP), to test the efficiency of block and factor detection.

4.1.1. Control parameter setting

The calculation conditions are shown in Table 2. The number of sampling points for each independent variable is 100. The regions for cases 1–5 and 7–10 are chosen as $[-3,3]^3$, $[-3,3]^4$, $[-3,3]^5$, and $[-3,3]^6$ for three-dimensional (3D), 4D, 5D, and 6D problems, respectively, while case 7 is $[1,3]^5$. The control parameters

in LDSE are set as follows. The upper and lower bounds of fitting parameters are set as -50 and 50. The population size N_p is set to $N_p = 10 + 10d$, where d is the dimension of the problem. The maximum generations is set to $3N_p$. Note that the maximum number of partially separable variables in all target models is two in our tests. Hence, our uni-variable and bi-variables function library of BBP could be set as in Table 1. Recalling from Section 3.3, sequence search and optimization is used in BBP. The search will exit immediately if the mean square error is small enough (MSE $\leq \varepsilon_{target}$), and the tolerance (fitting error) is $\varepsilon_{target} = 10^{-6}$. In order to reduce the effect of randomness, each test case is executed 10 times.

4.1.2. Numerical results and discussion

Numerical results show that LDSE-powered BBP successfully recovered all target functions exactly in sense of double precision. Once the uni- and bi-variables models are pre-set, the sequence search method makes BBP easy to find the best regression model. In practical applications, more function models could be added to the function library of BBP, provided that they are needed. On the other hand, as sketched in Table 2, the calculation time of the separability detection T_d is almost negligible. This test group shows that BBP has a good capability of structure and coefficient optimization for highly nonlinear system.

4.2. GPTIPS-powered BBP

We choose a kind of genetic programming technique, GPTIPS [23], as the optimization engine. GPTIPS is a Matlab toolbox based on multi-gene genetic programming. It has been widely used in many research studies [30–32]. To provide BBP an overall evaluation of its performance for acceleration, the acceleration rate η is defined as

$$\eta = \frac{T_{GPTIPS}}{T_{BBP}}, \qquad (15)$$

where T_{GPTIPS} is the computation time of GPTIPS, and T_{BBP} is the computation time of BBP driven by GPTIPS. The full names of the notations in Table 2 are the case number (Case No.), the range of mean square error of the regression model for all runs (MSE), the average computation time for all runs (T), and remarks of BBP test.

4.2.1. Control parameter setting

Similar to Section 4.1.1, the target models, search regions, and the number of sampling points are the same as the aforementioned test group. The control parameters of GPTIPS are set as follows. The population size $N_p = 100$ and the maximum generations for re-initialization T are 100,000. To reduce the influence of randomness, 20 runs are completed for each case. The termination condition is MSE $\leq \varepsilon_{target}$, $\varepsilon_{target} = 10^{-8}$. In other words, the optimization of each factor will terminate immediately if the regression model (or its equivalent alternative) is detected, and restart automatically if it fails until generation T. The multi-gene of GPTIPS is switched off.

4.2.2. Numerical results and discussion

Table 3 shows the average performance of the 20 independent runs with different initial populations. In this test group, using the given control parameters, GPTIPS failed to obtain the exact regression model or the approximate model with the default accuracy (MSE $\leq 10^{-8}$) in almost 20 runs except case 2. This situation becomes even worse with increasing problem size (dimension of the problem). Additionally, as the result of T_{GPTIPS}/T_{BBP} shown in Table 3, the computational efficiency increases several orders of magnitude. This is because the computation time of BBP is determined by the dimension and complexity of each factor, not by the entire target model. This explains why BBP converges much faster

Table 3
Performance of GPTIPS and GPTIPS-powered BBP.

Case	Results of GPTIPS		Results of GPTIPS-powered BBP			
No.	$\mathrm{MSE_{GPTIPS}}$	$T_{\mathrm{GPTIPS}}(s)$	$\mathrm{MSE_{BBP}}$	$T_{\mathrm{BBP}}(s)$	$\eta = T_{\mathrm{GPTIPS}}/T_{\mathrm{BBP}}$	Remarks of BBP
1	$[5.1, 9.2] \times 10^{-1}$	$\gg 6.23 \times 10^3$	$\leq \varepsilon_{\mathrm{target}}$	1512.6	> 4.12	5 runs failed
2	$\leq \varepsilon_{\mathrm{target}}$	323.86	$\leq \varepsilon_{\mathrm{target}}$	3.94	82.2	Solutions are all exact
3	$[1.5, 23.6] \times 10^{-2}$	$\gg 7.41 \times 10^3$	$\leq \varepsilon_{\mathrm{target}}$	3643.3	> 2.03	11 runs failed
4	$[8.1, 16.2] \times 10^{-2}$	$\gg 6.16 \times 10^3$	$\leq \varepsilon_{\mathrm{target}}$	903.87	> 6.8	4 runs failed
5	$[4.5, 7.3] \times 10^{-1}$	$\gg 6.68 \times 10^3$	$\leq \varepsilon_{\mathrm{target}}$	26.65	> 250.65	Solutions are all exact
6	$[2.31, 9.6] \times 10^{-2}$	$\gg 6.31 \times 10^3$	$\leq \varepsilon_{\mathrm{target}}$	4416.07	> 1.67	7 runs failed
7	$[1.22, 3.96] \times 10^{-1}$	$\gg 8.52 \times 10^3$	$\leq \varepsilon_{\mathrm{target}}$	11.57	> 736.39	Solutions are all exact
8	$[2.1, 37.2] \times 10^{-1}$	$\gg 7.13 \times 10^3$	$[9.16, 32.3] \times 10^{-2}$	1.3721×10^4	None	All runs failed
9	$[5.4, 56.3] \times 10^{-2}$	$\gg 6.24 \times 10^3$	$[1.68, 12.9] \times 10^{-2}$	$\gg 6.63 \times 10^3$	None	All runs failed
10	$[5.86, 99.16] \times 10^{-1}$	$\gg 7.36 \times 10^3$	$\leq \varepsilon_{\mathrm{target}}$	11.62	> 708.26	Solutions are all exact

than the original GPTIPS. Good performance for acceleration, structure optimization, and coefficient optimization show the potential of BBP to be applied in practical applications.

4.3. Computational efficiency analysis

We compare the target functional spaces of conventional GP method (e.g. GPTIPS) and GPTIPS-powered BBP. The search space of BBP is each factor of the target model, not the entire target model. It is obvious that GPTIPS has a larger target function space, and GPTIPS-powered BBP might be considered a special case of GPTIPS. This is the reason why GPTIPS-powered BBP is more effective and efficient than conventional GPTIPS.

The computing time (t) of BBP consists of three parts:

$$t = t_1 + t_2 + t_3 \qquad (16)$$

where t_1 is for the separability detection, t_2 for factors modeling, and t_3 for global assembling. In [20], authors have demonstrated that both the separability detection and function recover processes are double-precision operations and thus cost much less time than the factor determination process. t_2 is the most expensive part used to construct a data-driven model. That is, $t \approx t_2$.

As shown in Table 3, the CPU time for determining all factors (t_2) is much less than that of the target function directly (t_d). Therefore, in practical applications, we do not consider the computation complexity of a separable target model, but each factor of it.

5. Conclusion

We established a more general separable model with mixed binary operators. In order to obtain the structure of the generalized model, a block building programming (BBP) method is proposed for symbolic regression. BBP reveals the target separable model by a block and factor detection process, which divides the original model into a number of blocks, and further into factors. The method can be considered as a bi-level divide-and-conquer (D&C) method. The separability is detected by a generalized BiCT. The factors could be easily determined by an existing optimization engine (e.g. genetic programming). Thus BBP can reduce the complexity of the optimization model, and make large reductions to the original search space. Two different optimization engines, LDSE and GPTIPS, have been applied to test the performance of BBP on 10 symbolic regression problems. Numerical results show that BBP has a good capability of structure and coefficient optimization with high computational efficiency. These advantages make BBP a potential method for modeling complex nonlinear systems in various research fields.

For future work, we plan to generalize the mathematical form of the separable function. In Definition 2, all variables appear only once in the separable function. However, in practical applications, this condition is still too strong and is not easy to satisfy. In fact, many models have quasi-separable features. For example, the flow pass of a circular cylinder is a classical problem in fluid dynamics [33]. A valid stream function for the inviscid, incompressible flow pass of a circular cylinder of radius R is

$$\psi = (V_\infty r \sin\theta)\left(1 - \frac{R^2}{r^2}\right) + \frac{\Gamma}{2\pi}\ln\frac{r}{R}, \qquad (17)$$

Eq. (17) is expressed in terms of polar coordinates, where $x = r\cos\theta$ and $y = r\sin\theta$. V_∞ is the freestream velocity. Γ is the strength of vortex flow. R is the radius of the cylinder. Eq. (17) can be considered quasi-separable. That is, some variables (e.g. variable V_∞, θ, and Γ of Eq. (17)) appear only once in a concerned target model, while other variables (e.g. variable r and R of Eq. (17)) appear more than once. This makes Eq. (17) inconsistent with the definition of the separable function. Such complicated model structures would be analyzed in further studies.

Appendix A. Proof of Theorem 1

Proof. To prove the sufficient condition, consider the binary operator $\otimes_1$. Eq. (8) could be written as

$$f(\mathbf{x}) = \alpha_0 + \alpha_1 f_1\big(\mathbf{I}^{(1)}\mathbf{x}\big)\otimes_2\alpha_2 f_2\big(\mathbf{I}^{(2)}\mathbf{x}\big)\otimes_3 \ldots \otimes_m\alpha_m f_m\big(\mathbf{I}^{(m)}\mathbf{x}\big), \qquad (A.1)$$

or

$$f(\mathbf{x}) = \alpha_0 \times \alpha_1 f_1\big(\mathbf{I}^{(1)}\mathbf{x}\big)\otimes_2\alpha_2 f_2\big(\mathbf{I}^{(2)}\mathbf{x}\big)\otimes_3 \ldots \otimes_m\alpha_m f_m\big(\mathbf{I}^{(m)}\mathbf{x}\big). \qquad (A.2)$$

Consider the position of each binary plus operator ($+$). Assume that there are $p-1$ ($p \geq 1$) binary plus operators ($+$) in Eq. (8). If $p > 1$, assume the first binary operator plus ($+$) (except $\otimes_1$) of Eq. (8) appears in the middle of the sub-functions of variables $x_1, x_2, \ldots, x_{s_1}$ (including a number of $q_1 + q_2$ sub-functions) and $x_{s_1+1}, x_{s_1+2}, \ldots, x_n$, that is

$$f(\mathbf{x}) = \alpha_0 + \tilde{\beta}_1\varphi_1(x_1, x_2, \ldots x_{s_1}) + \tilde{\beta}_2\tilde{\varphi}_2(x_{s_1+1}, x_{s_1+2}, \ldots x_n), \quad (A.3)$$

or

$$f(\mathbf{x}) = \alpha_0 \times \tilde{\beta}_1\varphi_1(x_1, x_2, \ldots x_{s_1}) + \tilde{\beta}_2\tilde{\varphi}_2(x_{s_1+1}, x_{s_1+2}, \ldots x_n). \quad (A.4)$$

Assume that the second binary plus operator ($+$) appears in the middle of the sub-functions of variables $x_1, x_2, \ldots, x_{s_1+s_2}$ (included

a number of $q_1 + q_2$ sub-functions) and $x_{s_1+s_2+1}, x_{s_1+s_2+2}, \ldots, x_n$, that is

$$f(\mathbf{x}) = \alpha_0 + \tilde{\beta}_1 \varphi_1(x_1, \ldots, x_{s_1}) + \tilde{\beta}_2 \varphi_2(x_{s_1+1}, \ldots, x_{s_1+s_2})$$
$$+ \tilde{\beta}_3 \tilde{\varphi}_3(x_{s_1+s_2+1}, \ldots, x_n), \tag{A.5}$$

or

$$f(\mathbf{x}) = \alpha_0 \times \tilde{\beta}_1 \varphi_1(x_1, \ldots, x_{s_1}) + \tilde{\beta}_2 \varphi_2(x_{s_1+1}, \ldots, x_{s_1+s_2})$$
$$+ \tilde{\beta}_3 \tilde{\varphi}_3(x_{s_1+s_2+1}, \ldots, x_n). \tag{A.6}$$

The position of the rest $p - 4$ binary plus operators $(+)$ could be determined in the same way. We obtain

$$f(\mathbf{x}) = \alpha_0 + \sum_{i=1}^{p} \tilde{\beta}_i \varphi_i(\mathbf{I}^{(i)}\mathbf{x}), \tag{A.7}$$

or

$$f(\mathbf{x}) = \alpha_0 \times \sum_{i=1}^{p} \tilde{\beta}_i \varphi_i(\mathbf{I}^{(i)}\mathbf{x}). \tag{A.8}$$

$\mathbf{I}^{(i)} \in \mathbb{R}^{s_i \times n}$ is the partitioned matrix of the identity matrix $\mathbf{I} \in \mathbb{R}^{n \times n}$, $\sum_{i=1}^{p} s_i = n$. For Eq. (A.7), $\beta_0 = \alpha_0$, $\beta_i = \tilde{\beta}_i$. For Eq. (A.8), $\beta_0 = 0$, $\beta_i = \alpha_0 \tilde{\beta}_i$. Hence, the left-hand side of Eq. (10) can be obtained.

If $p = 1$, then there is no binary plus operator $(+)$ (except $\otimes_1$) in Eq. (8). Under this condition, it is obvious that Eqs. (A.7) and (A.8) are still satisfied.

Now decide the position of each binary times operator $(\times)$. Consider the first sub-function $\varphi_1(\mathbf{I}^{(1)}\mathbf{x})$ in Eqs. (A.7) and (A.8). Assume that the first binary times operator $(\times)$ of the sub-function $\varphi_1(\mathbf{I}^{(1)}\mathbf{x})$ appears in the middle of the sub-functions of variables $x_1, x_2, \ldots, x_{s_{1,1}}$ and $x_{s_{1,1}+1}, x_{s_{1,1}+2}, \ldots, x_{s_1}$, that is

$$\varphi_1(\mathbf{I}^{(1)}\mathbf{x}) = \psi_{1,1}(x_1, x_2, \ldots, x_{s_{1,1}}) \ldots \tilde{\psi}_1(x_{s_{1,1}+1}, x_{s_{1,1}+2}, \ldots, x_{s_1}). \tag{A.9}$$

The second binary times operator $(\times)$ of the sub-function $\varphi_1(\mathbf{I}^{(1)}\mathbf{x})$ appears in the middle of the sub-functions of variables $x_1, x_2, \ldots, x_{s_{1,1}+s_{1,2}}$ and $x_{s_{1,1}+s_{1,2}+1}, x_{s_{1,1}+s_{1,2}+2}, \ldots, x_{s_1}$, that is

$$\varphi_1(\mathbf{I}^{(1)}\mathbf{x}) = \psi_{1,1}(x_1, \ldots, x_{s_{1,1}}) \cdot \psi_{1,2}(x_{s_{1,1}+1}, \ldots, x_{s_{1,1}+s_{1,2}})$$
$$\cdot \tilde{\psi}_2(x_{s_{1,1}+s_{1,2}+1}, \ldots, x_{s_1}). \tag{A.10}$$

The position of the rest binary times operators $(\times)$ of the sub-function $\varphi_1(\mathbf{I}^{(1)}\mathbf{x})$ could be determined in the same way. Then, $\varphi_1(\mathbf{I}^{(1)}\mathbf{x})$ could be rewritten as

$$\varphi_1(\mathbf{I}^{(1)}\mathbf{x}) = \prod_{j=1}^{q_1} \psi_{1,j}(\mathbf{I}_j^{(1)}\mathbf{x}), \tag{A.11}$$

where $\mathbf{I}_j^{(1)} \in \mathbb{R}^{s_{1,j} \times n}$ is the partitioned matrix of $\mathbf{I}^{(1)}$, namely $\mathbf{I}^{(1)} = \begin{bmatrix} \mathbf{I}_1^{(1)} & \mathbf{I}_2^{(1)} & \cdots & \mathbf{I}_{q_1}^{(1)} \end{bmatrix}$, $\sum_{j=1}^{q_1} s_{1,j} = s_1$. Hence, for arbitrary sub-function $\varphi_i(\mathbf{I}^{(i)}\mathbf{x})$, we have

$$\varphi_i(\mathbf{I}^{(i)}\mathbf{x}) = \prod_{j=1}^{q_i} \psi_{i,j}(\mathbf{I}_j^{(i)}\mathbf{x}), \tag{A.12}$$

where $\mathbf{I}_j^{(i)} \in \mathbb{R}^{s_{i,j} \times n}$ is the partitioned matrix of $\mathbf{I}^{(i)}$, namely $\mathbf{I}^{(i)} = \begin{bmatrix} \mathbf{I}_1^{(i)} & \mathbf{I}_2^{(i)} & \cdots & \mathbf{I}_{q_i}^{(i)} \end{bmatrix}$, $\sum_{j=1}^{q_i} s_{i,j} = s_i$, $\sum_{i=1}^{p} q_i = m$. Substituting Eq. (A.12) into Eqs. (A.7) and (A.8), the right-hand side of Eq. (10) can be obtained.

To prove the necessary condition, expand Eq. (10). Since $\sum_{i=1}^{p} q_i = m$, there are m sub-functions f_i and each function can be connected with a binary operator $\otimes_i$. Then, the Eq. (8) can be easily obtained. □

Appendix B. 10 target models of numerical experiments

The target models which are tested in Section 4 with all the blocks boxed are given as follows:

Case 1. $f(\mathbf{x}) = 1.2 + 10 * \boxed{\sin(2x_1 - x_3)} - 3 * \boxed{x_2^2}$, where $x_i \in [-3, 3], i = 1, 2, 3$.

Case 2. $f(\mathbf{x}) = 0.5 * \boxed{e^{x_3} * \sin x_1 * \cos x_2}$, where $x_i \in [-3, 3], i = 1, 2, 3$.

Case 3. $f(\mathbf{x}) = \boxed{\cos(x_1 + x_2)} + \boxed{\sin(3x_3 - x_4)}$, where $x_i \in [-3, 3], i = 1, 2, 3, 4$.

Case 4. $f(\mathbf{x}) = 5 * \boxed{\dfrac{\sin(3x_1 x_2)}{x_3 + x_4}}$, where $x_i \in [-3, 3], i = 1, 2, 3, 4$.

Case 5. $f(\mathbf{x}) = 2 * \boxed{x_1 * \sin(x_2 + x_3)} - \boxed{\cos x_4}$, where $x_i \in [-3, 3], i = 1, 2, 3, 4$.

Case 6. $f(\mathbf{x}) = 10 + 0.2 * \boxed{x_1} - 5 * \boxed{\sin(5x_2 + x_3)} + \boxed{\ln(3x_4 + 1.2)} - 1.2 * \boxed{e^{0.5x_5}}$, where $x_i \in [1, 4], i = 1, 2, \ldots, 5$.

Case 7. $f(\mathbf{x}) = 10 * \boxed{\dfrac{\sin(x_1 x_2) * x_3}{x_4 + x_5}}$, where $x_i \in [-3, 3], i = 1, 2, \ldots, 5$.

Case 8. $f(\mathbf{x}) = 1.2 + 2 * \boxed{x_4 * \cos x_2} + 0.5 * \boxed{e^{1.2x_3} * \sin 3x_1} - 2 * \boxed{\cos(1.5x_5 + 5)}$, where $x_i \in [-3, 3], i = 1, 2, \ldots, 5$.

Case 9. $f(\mathbf{x}) = 100 * \boxed{\dfrac{\cos(x_3 x_4)}{e^{x_1} * x_2^{1.2}} * \sin(1.5x_5 - 2x_6)}$, where $x_i \in [-3, 3], i = 1, 2, \ldots, 6$.

Case 10. $f(\mathbf{x}) = \boxed{\dfrac{x_1 + x_2}{x_3}} + \boxed{x_4 * \sin(x_5 x_6)}$, where $x_i \in [-3, 3], i = 1, 2, \ldots, 6$.

References

[1] J.L. Salmeron, S.A. Rahimi, A.M. Navali, A. Sadeghpour, Medical diagnosis of rheumatoid arthritis using data driven PSO-FCM with scarce datasets, Neurocomputing 232 (2017) 104–112, doi:10.1016/j.neucom.2016.09.113.

[2] P. Yan, D. Liu, D. Wang, H. Ma, Data-driven controller design for general MIMO nonlinear systems via virtual reference feedback tuning and neural networks, Neurocomputing 171 (2016) 815–825, doi:10.1016/j.neucom.2015.07.017.

[3] M. Seera, C.P. Lim, K.S. Tan, W.S. Liew, Classification of transcranial doppler signals using individual and ensemble recurrent neural networks, Neurocomputing 249 (2017) 337–344, doi:10.1016/j.neucom.2016.05.117.

[4] H.M.R. Ugalde, J.-C. Carmona, J. Reyes-Reyes, V.M. Alvarado, J. Mantilla, Computational cost improvement of neural network models in black box nonlinear system identification, Neurocomputing 166 (2015) 96–108, doi:10.1016/j.neucom.2015.04.022.

[5] C. Luo, Z. Hu, S.-L. Zhang, Z. Jiang, Adaptive space transformation: an invariant based method for predicting aerodynamic coefficients of hypersonic vehicles, Eng. Appl. Artif. Intell. 46 (2015) 93–103, doi:10.1016/j.engappai.2015.09.001.

[6] V. Parque, T. Miyashita, A method to learn high-performing and novel product layouts and its application to vehicle design, Neurocomputing 248 (2017) 41–56, doi:10.1016/j.neucom.2016.12.082.

[7] C. Chen, C. Luo, Z. Jiang, Elite bases regression: a real-time algorithm for symbolic regression, in: Proceedings of the 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery, Guilin, Guangxi, China, 2017, pp. 500–506.

[8] Y. Peng, C. Yuan, X. Qin, J. Huang, Y. Shi, An improved gene expression programming approach for symbolic regression problems, Neurocomputing 137 (2014) 293–301, doi:10.1016/j.neucom.2013.05.062.

[9] L.F. dal Piccol Sotto, V.V. de Melo, Studying bloat control and maintenance of effective code in linear genetic programming for symbolic regression, Neurocomputing 180 (2016) 79–93, doi:10.1016/j.neucom.2015.10.109.

[10] J.R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, fifth ed., MIT Press, Cambridge, MA, 1992.

[11] M. O'Neill, C. Ryan, Grammatical evolution, IEEE Trans. Evolut. Comput. 5 (4) (2001) 349–358, doi:10.1109/4235.942529.

[12] C. Luo, S.-L. Zhang, Parse-matrix evolution for symbolic regression, Eng. Appl. Artif. Intell. 25 (6) (2012) 1182–1193, doi:10.1016/j.engappai.2012.05.015.

[13] Z. Gan, T.W. Chow, W. Chau, Clone selection programming and its application to symbolic regression, Expert Syst. Appl. 36 (2) (2009) 3996–4005, doi:10.1016/j.eswa.2008.02.030.

[14] D. Karaboga, C. Ozturk, N. Karaboga, B. Gorkemli, Artificial bee colony programming for symbolic regression, Inf. Sci. 209 (2012) 1–15, doi:10.1016/j.ins.2012.05.002.

[15] T. McConaghy, FFX: Fast, Scalable, Deterministic Symbolic Regression Technology, Springer, New York, pp. 235–260.

[16] A. Worm, Prioritized grammar enumeration: a novel method for symbolic regression, Binghamton University - State University of New York, 2016 (Ph.D. thesis).

[17] D. Kinzett, M. Zhang, M. Johnston, Using Numerical Simplification to Control Bloat in Genetic Programming, Springer, Berlin, Heidelberg, pp. 493–502. doi:10.1007/978-3-540-89694-4_50.

[18] D. Kinzett, M. Johnston, M. Zhang, How online simplification affects building blocks in genetic programming, in: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, Montreal, Canada, 2009, pp. 979–986. doi:10.1145/1569901.1570035.

[19] R.K. McRee, K. Software, M. Park, Symbolic regression using nearest neighbor indexing, in: Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation, Portland, Oregon, USA, 2010, pp. 1983–1990. doi:10.1145/1830761.1830841.

[20] C. Luo, C. Chen, Z. Jiang, A divide and conquer method for symbolic regression, 2017. Preprinted on ArXiv at https://arxiv.org/abs/1705.08061.

[21] J.D. Anderson, Hypersonic and High-Temperature Gas Dynamics, second ed., American Institute of Aeronautics and Astronautics, Inc., Virginia, 2006.

[22] D.P. Raymer, Aircraft Design: A Conceptual Approach, fifth ed., American Institute of Aeronautics and Astronautics, Inc., 2012.

[23] D.P. Searson, D.E. Leahy, M.J. Willis, GPTIPS: an open source genetic programming toolbox for multigene symbolic regression, in: Proceedings of the International Multi Conference of Engineers and Computer Scientists, Hong Kong, China, 2010.

[24] C. Luo, B. Yu, Low dimensional simplex evolution a new heuristic for global optimization, J. Glob. Optim. 52 (1) (2012) 45–55, doi:10.1007/s10898-011-9678-1.

[25] Y. Wei, J. Qiu, H.R. Karimi, Quantized H_∞ filtering for continuoustime markovian jump systems with deficient mode information, Asian J. Control 171 (5) (2015) 1914–1923, doi:10.1002/asjc.1052.

[26] Y. Wei, X. Peng, J. Qiu, S. Jia, H_∞ filtering for two-dimensional continuous-time markovian jump systems with deficient transition descriptions, Neurocomputing 167 (2015) 406–417, doi:10.1016/j.neucom.2015.04.054.

[27] Y. Wei, J. Qiu, H.R. Karimi, Reliable output feedback control of discrete-time fuzzy affine systems with actuator faults, IEEE Trans. Circuits Syst. I Regul. Pap. 64 (1) (2017) 170–181, doi:10.1109/TCSI.2016.2605685.

[28] A. Forrester, A. Sobester, A. Keane, Engineering Design via Surrogate Modelling: A Practical Guide, John Wiley & Sons, 2008.

[29] S.N. Lophaven, H.B. Nielsen, J. Søndergaard, DACE–A Matlab Kriging Toolbox., Technical Report IMM-REP-2002-12, Technical University of Denmark, 2002.

[30] A. Garg, A. Garg, K. Tai, S. Sreedeep, An integrated SRM-multi-gene genetic programming approach for prediction of factor of safety of 3-D soil nailed slopes, Eng. Appl. Artif. Intel. 30 (2014) 30–40, doi:10.1016/j.engappai.2013.12.011.

[31] A.H. Alavi, H. Hasni, I. Zaabar, N. Lajnef, A new approach for modeling of flow number of asphalt mixtures, Arch. Civil Mech. Eng. 17 (2) (2017) 326–335, doi:10.1016/j.acme.2016.06.004.

[32] H. Kaydani, A. Mohebbi, M. Eftekhari, Permeability estimation in heterogeneous oil reservoirs by multi-gene genetic programming algorithm, J. Pet. Sci. Eng. 123 (2014) 201–206, doi:10.1016/j.petrol.2014.07.035.

[33] J.D. Anderson, Fundamentals of Aerodynamics, MacGraw-Hill, New York, fifth ed, 2011.

Chen Chen is currently a master candidate in the Institute of Mechanics, Chinese Academy of Sciences, Beijing, China. He received his bachelor's degree in Aircraft Design Engineering from Northwestern Polytechnical University, Xi'an, China in 2015. His research interests include fast mathematical modelling methods and their applications in aerodynamic forces and heating prediction.

Changtong Luo is an associate professor in the Institute of Mechanics, Chinese Academy of Sciences, Beijing, China. He received his Ph.D. degree from Jilin University in 2007. He has been working on Nagoya University, Japan from 2007 to 2009 as a COE researcher. His research interests include computational fluid dynamics, evolutionary computation, global optimization, and numerical algebra, and their applications in aerodynamics.

Zonglin Jiang is a professor in the Institute of Mechanics, Chinese Academy of Sciences, Beijing, China. He received his Ph.D. degree from Peking University in 1993. He received "One Hundred Person Project" of the Chinese Academy of Sciences in 1999. He had been the director of State Key Laboratory of High Temperature Gas Dynamics, Institute of Mechanics since 2001 until 2015. He was granted the Ground Testing Award 2016 by American Institute of Aeronautics and Astronautics, for his skillful leadership in developing and successful commissioning of the world's largest shock tunnel JF12. His research interests include shockwave and detonation physics, supersonic and hypersonic experiments, etc.