Block building programming for symbolic regression

Chen Chen\textsuperscript{a,b}, Changtong Luo\textsuperscript{a,*}, Zonglin Jiang\textsuperscript{a,b}

\textsuperscript{a}State Key Laboratory of High Temperature Gas Dynamics, Institute of Mechanics, Chinese Academy of Sciences, Beijing 100190, China

\textsuperscript{b}School of Engineering Sciences, University of Chinese Academy of Sciences, Beijing 100049, China

Abstract

Symbolic regression that aims to detect underlying data-driven model has become increasingly important for industrial data analysis. For most of 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), in this paper. 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 circumstance, 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.

Keywords: Symbolic regression, Separable function, Block building programming, Genetic programming

*Corresponding author

\textit{Email addresses:} chenchen@imech.ac.cn (Chen Chen), luo@imech.ac.cn (Changtong Luo), zljiang@imech.ac.cn (Zonglin Jiang)
1. Introduction

Data-driven modeling of complex system has become increasingly important for industrial data analysis when the experimental model structure is unknown or wrong, or the concerned system has changed [20, 26]. 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 [22], system identification [24], industrial data analysis [12], industrial design [17], etc. Unlike conventional regression methods that require to provide 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., $+$, $-$, $\times$, $\div$, etc.) and mathematical functions (e.g., $\sin$, $\cos$, $\exp$, $\ln$, etc.), which can be described as follows:

$$f^* = \arg\min_{f \in S} \sum_i \|f(x^{(i)}) - y_i\|,$$  \hspace{1cm} (1)

where $x^{(i)} \in \mathbb{R}^d$ 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 NP-problem, which optimizes the structure and coefficient of a target model simultaneously. How to use an appropriate method to solve a symbolic regression problem is considered as a kaleidoscope in this research field [4, 18, 23].

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 get 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 increasing range of these variables). It is because that the target model with a large number of variables might results in large search depth and high computational cost of GP. Then the convergence speed of GP might be too slow. This makes GP very inconvenient in engineering application.

Apart from basic GP, there are two groups of methods for symbolic regression have been studied. The first group has focused on evolutionary strategy, such as grammatical evolution [16], parse-matrix evolution [14], etc, have been proposed. These variants of GP can simplify the coding process. Gan et al. [5] have introduced a clone selection programming method based on
artificial immune system. Karaboga et al. [8] have proposed a artificial bee colony programming method based on the foraging behavior of honey bees. However, these methods are still based on the idea of biological simulation process. This makes that they help little on improving the convergence speed when solving large scale problems.

The second branch has exploited strategies to reduce the search space of solution. For instance, McConaghy [15] has 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 [25] has proposed a deterministic machine learning algorithm, Prioritized Grammar Enumeration (PGE), in his thesis. The author stated that it could make a large reduction to the search space. In fact, PGE is a hybrid algorithm that combines with GP.

Fortunately, in many scientific or engineering problems, it is found that the target models are separable or partially separable. For instance, when developing a rocket engine, it is crucial to model the internal flow of a high-speed compressible gas through the nozzle [3]. The closed-form expression for the mass flow through a choked nozzle is

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

Note that the five independent variables, $p_0$, $T_0$, $A^*$, $R$ and $\gamma$ are all separable in Eq. (2). That is, the target model can be re-expressed as follows

$$\dot{m} = f(p_0, A^*, T_0, R, \gamma) = \varphi_1(p_0) \cdot \varphi_2(A^*) \cdot \varphi_3(T_0) \cdot \varphi_4(R) \cdot \varphi_5(\gamma).$$  \hspace{1cm} (3)

We can see that the target function with five independent variables can be divided into five sub-functions multiplied together, and each sub-function has only one independent variable. Furthermore, the binary operator between two sub-functions could be plus (+) or times (×). For example, in aircraft design [19], the lift coefficient of a whole airplane can be expressed as

$$C_L = C_{La} (\alpha - \alpha_0) + C_L\delta_e \frac{S_{HT}}{S_{ref}},$$  \hspace{1cm} (4)
where the variable $C_{L\alpha}$, $C_{L\delta_e}$, $\delta_e$, $S_{HT}$ and $S_{ref}$ are separable. Variables $\alpha$ and $\alpha_0$ are not separable, but their combination $(\alpha, \alpha_0)$ can be considered separable. Hence, Eq. (4) can be re-expressed as

$$C_L = f (C_{L\alpha}, \alpha, \alpha_0, C_{L\delta_e}, \delta_e, S_{HT}, S_{ref}) = \varphi_1 (C_{L\alpha}) \cdot \varphi_2 (\alpha, \alpha_0) + \varphi_3 (C_{L\delta_e}) \cdot \varphi_4 (\delta_e) \cdot \varphi_5 (S_{HT}) \cdot \varphi_6 (S_{ref}).$$ (5)

Here, the target function is divided into six sub-functions.

Luo et al. [11] suggest using the separability to accelerate GP for symbolic regression. The authors have proposed a divide and conquer (D&C) method, which can decompose a concerned separable target function, e.g., Eq. (3), into a number of sub-functions and then optimize them. The separability is probed by a special technique, bi-correlation test (BiCT). However, the D&C method can only be valid for a additively or multiplicatively separable target model. In many practical problems, the separable functions might be constructed with mixed binary operators, namely plus (+), minus (−), times (×) and division (÷), e.g., Eq. (5). This limits D&C method for further applications.

In this paper, a new approach, block building programming (BBP), for symbolic regression is proposed. BBP is an improved version of D&C method [11], which is capable of dealing with hybrid binary operators (+, −, × and ÷). By detecting the separability, the original target function is divided into several blocks, and further into factors. Meanwhile, binary operators could also be determined. The block and factor detection is based on BiCT technique. Numerical results show that BBP can get the target functions with complex forms more reliably, and can produce extremely large acceleration of GP method for symbolic regression.

The presentation of this paper is organized as follows: Fundamentals including the definition of separable function and bi-correlation test (BiCT) technique are briefly reviewed in Section 2. The principle and procedure of block building programming approach are described in Section 3. Section 4 presents numerical results, discussions and efficiency analyses for the proposed method. In the last section, conclusions are drawn with remarking the future work.

2. Fundamentals

In this section, fundamentals including the definition of separable function and bi-correlation test (BiCT) technique introduced in [11] are briefly
reviewed. The BiCT technique is originally proposed to determine whether a concerned target function has an additively or multiplicatively separable form. This step is the foundation of our proposed approach, block building programming. Before describing the procedure of BiCT, the definition of separable function proposed in [11] is introduced first.

2.1. Separable function

Definition 1. (Partially separable function) A scalar function \( f(x) \) with \( n \) continuous variables \( x = [x_1, x_2, \cdots, x_n]^T \) (\( f : \mathbb{R}^n \mapsto \mathbb{R}, \ x \in \mathbb{R}^n \)) is said to be partially separable if and only if it can be rewritten as

\[
 f(x) = c_0 \otimes_1 \varphi_1 (I^{(1)}x) \otimes_2 c_2 \varphi_2 (I^{(2)}x) \otimes_3 \cdots \otimes_m c_m \varphi_m (I^{(m)}x) \tag{6}
\]

where \( I^{(i)} \in \mathbb{R}^{n_i \times n} \) is the partitioned matrix of the identity matrix \( I \in \mathbb{R}^{n \times n} \), namely \( I = [I^{(1)} \ I^{(2)} \ \cdots \ I^{(m)}]^T \), \( \sum_{i=1}^{m} n_i = n \). \( I^{(i)}x \) is the variables set with \( n_i \) elements. \( n_i \) represents the number of variables in sub-function \( \varphi_i \).

Sub-function \( \varphi_i \) is a scalar function such that \( \varphi_i : \mathbb{R}^{n_i} \mapsto \mathbb{R} \). The binary operator \( \otimes_i \) could be plus (+) and times (\( \times \)). \( c_i \) is constant coefficient.

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

Definition 2. (Completely separable function) A scalar function \( f(x) \) with \( n \) continuous variables (\( f : \mathbb{R}^n \mapsto \mathbb{R}, \ x \in \mathbb{R}^n \)) is said to be completely separable if and only if it can be rewritten as Eq. (6) and \( n_i = 1 \) for all \( i = 1, 2, \cdots, m \).

2.2. Bi-correlation test

The BiCT technique is conducted by random sampling and linear correlation method. To give a brief introduction, we just consider the separability of the first \( n_1 \) variables \( I^{(1)}x = (x_1, x_2, \cdots, x_{n_1}) \) in Eq. (6) are discussed (\( f : \mathbb{R}^n \mapsto \mathbb{R}, \ x \in \Omega \subset \mathbb{R}^n \), and \( \Omega = [a, b]^n \)).

1. Let the matrix \( X \) be a set of \( N \) sampling points for all \( n \) independent variables,

\[
 X = \begin{bmatrix}
 x_{11} & x_{12} & \cdots & x_{1,n} \\
 x_{21} & x_{22} & \cdots & x_{2,n} \\
 \vdots & \vdots & \ddots & \vdots \\
 x_{N,1} & x_{N,2} & \cdots & x_{N,n}
\end{bmatrix}, \tag{7}
\]
2. Let the sampling points of variables $x_{n_1+1}, x_{n_1+2}, \cdots, x_n$ in Eq. (7) (the $(n_1 + 1)$-th, $(n_1 + 2)$-th, $\cdots$, $n$-th columns of matrix $X$) be fixed to any two given points $x_A$ and $x_B$, $x_A, x_B \in [a, b]$, respectively. Then we obtain

$$X^A = \begin{bmatrix} x_{11} & \cdots & x_{1,n_1} & x_{1,n_1}^{(A)} & \cdots & x_{1,n}^{(A)} \\ x_{22} & \cdots & x_{2,n_1} & x_{2,n_1}^{(A)} & \cdots & x_{2,n}^{(A)} \\ \vdots & & \vdots & & \vdots & \vdots \\ x_{N,1} & \cdots & x_{N,n_1} & x_{N,n_1}^{(A)} & \cdots & x_{N,n}^{(A)} \end{bmatrix} \quad (8)$$

and

$$X^B = \begin{bmatrix} x_{11} & \cdots & x_{1,n_1} & x_{1,n_1}^{(B)} & \cdots & x_{1,n}^{(B)} \\ x_{22} & \cdots & x_{2,n_1} & x_{2,n_1}^{(B)} & \cdots & x_{2,n}^{(B)} \\ \vdots & & \vdots & & \vdots & \vdots \\ x_{N,1} & \cdots & x_{N,n_1} & x_{N,n_1}^{(B)} & \cdots & x_{N,n}^{(B)} \end{bmatrix} \quad (9)$$

3. Let the sampling points of variables $x_1, x_2, \cdots, x_{n_1}$ in Eq. (7) (the first, second, $\cdots$, $n_1$-th columns of matrix $X$) be fixed to any two given points $x_C$ and $x_D$, $x_C, x_D \in [a, b]$, respectively. Then we obtain

$$X^C = \begin{bmatrix} x^{(C)}_{11} & \cdots & x^{(C)}_{1,n_1} & x_{1,n_1+1}^{(C)} & \cdots & x_{1,n}^{(C)} \\ x^{(C)}_{21} & \cdots & x^{(C)}_{2,n_1} & x_{2,n_1+1}^{(C)} & \cdots & x_{2,n}^{(C)} \\ \vdots & & \vdots & & \vdots & \vdots \\ x^{(C)}_{N,1} & \cdots & x^{(C)}_{N,n_1} & x_{N,n_1+1}^{(C)} & \cdots & x_{N,n}^{(C)} \end{bmatrix} \quad (10)$$

and

$$X^D = \begin{bmatrix} x^{(D)}_{11} & \cdots & x^{(D)}_{1,n_1} & x_{1,n_1+1}^{(D)} & \cdots & x_{1,n}^{(D)} \\ x^{(D)}_{21} & \cdots & x^{(D)}_{2,n_1} & x_{2,n_1+1}^{(D)} & \cdots & x_{2,n}^{(D)} \\ \vdots & & \vdots & & \vdots & \vdots \\ x^{(D)}_{N,1} & \cdots & x^{(D)}_{N,n_1} & x_{N,n_1+1}^{(D)} & \cdots & x_{N,n}^{(D)} \end{bmatrix} \quad (11)$$

4. Let $f^A$ be the vector of which the $i$-th element is the function value of the $i$-th row of matrix $X^A$, namely $f^A = f\left( X^A \right)$. Vectors $f^B$, $f^C$ and $f^D$ are similarly obtained.

5. If the components of two function-value vectors, $f^A$ and $f^B$, differ by a constant. Meanwhile, the components of two function-value vectors,
\( f^C \) and \( f^D \), differ by a constant too. The sub-function \( \varphi_1 \) of variables \( x_1, x_2, \cdots, x_n \) are separable from the sub-function \( \varphi_2 \) of variables \( x_{n+1}, x_{n+2}, \cdots, x_n \), with the binary operator \( \otimes_1 \) being plus (+). That is

\[
f(x) = \varphi_1(x_1, x_2, \cdots, x_n) + \varphi_2(x_{n+1}, x_{n+2}, \cdots, x_n).
\tag{12}
\]

If the two function-value vectors, \( f^A \) and \( f^B \), are linearly dependent. Meanwhile, the components of two function-value vectors, \( f^C \) and \( f^D \), are linearly dependent too. The sub-function \( \varphi_1 \) of variables \( x_1, x_2, \cdots, x_n \) are separable from the sub-function \( \varphi_2 \) of variables \( x_{n+1}, x_{n+2}, \cdots, x_n \), with the binary operator \( \otimes_1 \) being times (\( \times \)). That is

\[
f(x) = \varphi_1(x_1, x_2, \cdots, x_n) \cdot \varphi_2(x_{n+1}, x_{n+2}, \cdots, x_n).
\tag{13}
\]

The separability detection of variables \( I(i)x \) in sub-functions \( \varphi_i \), \( i = 1, 2, \cdots, m \), share the same way. The aforementioned BiCT method is established by two sufficient and necessary conditions, which are given below.

**Theorem 1.** Let \( \tilde{I}^{(i)} = [ I^{(1)} I^{(2)} \cdots I^{(i-1)} I^{(i+1)} \cdots I^{(m)} ] \in \mathbb{R}^{(n-n_i) \times n} \). Then, a given completely separable or partially separable function \( f(x) \) with \( n \) continuous variables \( (f : \mathbb{R}^n \mapsto \mathbb{R}, x \in \mathbb{R}^n) \) can be expressed as

\[
f(x) = k_0 + \sum_{i=1}^{m} k_i \varphi_i(I, x),
\tag{14}
\]

if and only if both of the following statements are true.

1. The components of any two output function-value vectors of sampling matrix \( X \) (Eq. (7)) with the variables \( I^i x \) being fixed, differ by a constant for all \( i = 1, 2, \cdots, m \);
2. The components of any two output function-value vectors of sampling matrix \( X \) (Eq. (7)) with the variables \( \tilde{I}^i x \) being fixed, differ by a constant for all \( i = 1, 2, \cdots, m \).

**Proof.** This theorem could be easily obtained from theorem 1 in [11].

**Theorem 2.** Let \( \tilde{I}^{(i)} = [ I^{(1)} I^{(2)} \cdots I^{(i-1)} I^{(i+1)} \cdots I^{(m)} ] \in \mathbb{R}^{(n-n_i) \times n} \). Then, a given completely separable or partially separable function \( f(x) \) with \( n \) continuous variables \( (f : \mathbb{R}^n \mapsto \mathbb{R}, x \in \mathbb{R}^n) \) can be expressed as

\[
f(x) = k_0 \prod_{i=1}^{m} \varphi_i(I, x),
\tag{15}
\]

if and only if both of the following statements are true.
(1) Any two output function-value vectors of sampling matrix $X$ (Eq. (7)) with the variables $I^{(i)}x$ being fixed are linearly dependent for all $i = 1, 2, \cdots, m$;

(2) Any two output function-value vectors of sampling matrix $\bar{X}$ (Eq. (7)) with the variables $\bar{I}^{(i)}x$ being fixed are linearly dependent for all $i = 1, 2, \cdots, m$.

Proof. This theorem could be easily obtained from theorem 1 in [11].

Based on the above two theorems, the procedure of the BiCT can be outlined as follows.

Procedure of BiCT:

Step 1. Initialize: Input dimension of problem $D$, test set of all independent variables $X = \{x_i : i = 1, 2, \cdots, D\}$, sampling interval $[a, b]$, number of sampling points $N$.

Step 2. Sample: Generate a sampling matrix $X \in \mathbb{R}^{N \times D}$, with $N$ random sampling points in $[a, b]$ for $D$ independent variables.

Step 3. Detect separability: If the test set $X$ is not empty, perform the following steps repeatedly. For $i = 1, 2, \cdots, D$, create a variables set $X^{(i)}$ which consists of all possible combinations of the set $X$ taken $i$ elements at a time.

\begin{enumerate}
  
  (3.1) Fix variables: Let the variables $X^{(i)}$ in matrix $X$ be fixed to $x_A, x_B \in [a, b]$, to obtain $X^A$ and $X^B$, respectively. Then, get two vectors of function values $f^A = f(X^A)$ and $f^B = f(X^B)$. Meanwhile, let the variables $\bar{X}^{(i)} = \bar{C}_X X^{(i)}$ in matrix $\bar{X}$ be fixed to $x_C, x_D \in [a, b]$, to obtain $X^C$ and $X^D$, respectively. Then, get two vectors of function values $f^C = f(X^C)$ and $f^D = f(X^D)$.

  (3.2) Test correlation: If the components of two vectors $f^A$ and $f^B$ are all constants, meanwhile $f^C$ and $f^D$ are all constants too. Then variable set $X^{(i)}$ is additively separable; If vector $f^A$ and vector $f^B$ are linearly dependent, meanwhile vector $f^C$ and vector $f^D$ are linearly dependent too. Then variable set $X^{(i)}$ is multiplicatively separable.

  (3.3) Preserve and update: Preserve the separable variable set $X^{(i)}$. Update the test variable set $X := \bar{C}_X X^{(i)}$. Then return to step 3.
\end{enumerate}
3. Block building programming

Our new approach is named as block building programming (BBP), which has two main steps:

1. Separability detection. In this step, a designed method, block and factor detection, is introduced. It detects the separability of the original target function, and decomposes the target function into several sub-functions. Meanwhile, binary operators (+, −, ×, and ÷) could also be determined.

2. Optimization and modeling. In this step, optimization engine, such as genetic programming method or global optimization algorithm, could be used to optimize the structure and coefficients of the sub-functions. Then, these optimized sub-functions are assembled into a regression model properly.

The aforementioned two steps will be detailly described in Section 3.1-3.3, then the procedure of our proposed approach, block building programming (BBP), will be given in Section 3.4.

3.1. Block and factor detection

The additively or multiplicatively separable target function can be easily detected by BiCT technique. However, how to determine each binary operator $\otimes_i$ of Eq. (6) is a critical step in BBP approach. One way is to recognize each binary operator $\otimes_i$ sequentially with random sampling and linear correlation techniques (similar to BiCT). For example, a given target function of six variables with five sub-functions is given below

$$f(x_1, \cdots, x_6) = \varphi_1(x_1) \otimes_1 \varphi_2(x_2, x_3) \otimes_2 \varphi_3(x_4) \otimes_3 \varphi_4(x_5) \otimes_4 \varphi_5(x_6)$$

$$= \varphi_1(x_1) \times \varphi_2(x_2, x_3) + \varphi_3(x_4) + \varphi_4(x_5) \times \varphi_5(x_6).$$  (16)

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

To overcome the aforementioned difficulty, a designed method, 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 3. Eq. (16) can be equivalently written as

\[ f(x) = k_0 + \sum_{i=1}^{p} k_i \omega_i \left( I^{(i)} x \right) = c_0 + \sum_{i=1}^{p} c_i \prod_{j=1}^{q_i} \varphi_{i,j} \left( I^{(i)} x \right), \]  

(17)

where \( x = [x_1, x_2, \cdots, x_n]^T \), \( x_i \in \mathbb{R}^n \) and \( f : \mathbb{R}^n \mapsto \mathbb{R} \). \( I^{(i)} \in \mathbb{R}^{n_i \times n} \) is the partitioned matrix of the identity matrix \( I \in \mathbb{R}^{n \times n} \), namely \( I = \left[ I^{(1)} \ I^{(2)} \ \cdots \ I^{(p)} \right]^T \), \( \sum_{i=1}^{p} n_i = n \). \( I^{(i)}_j \in \mathbb{R}^{n_{i,j} \times n} \) is the partitioned matrix of the \( I^{(i)} \), namely \( I^{(i)} = \left[ I^{(i)}_1 \ I^{(i)}_2 \ \cdots \ I^{(i)}_{q_i} \right] \), \( \sum_{j=1}^{q_i} n_{i,j} = n_i \). Sub-functions \( \omega_i \) and \( \varphi_{i,j} \) are scalar functions such that \( \omega_i : \mathbb{R}^{n_i} \mapsto \mathbb{R} \) and \( \varphi_{i,j} : \mathbb{R}^{n_{i,j}} \mapsto \mathbb{R} \).

Proof. See Appendix A. \( \square \)

Based on the theorem 3, the concepts of block and factor can be given below.

Definition 3. (Block and factor) The sub-function \( \omega_i \left( I^{(i)} x \right) \) is named as the \( i \)-th block of the Eq. (17), and the sub-function \( \varphi_{i,j} \left( I^{(i)} x \right) \) is named as the \( j \)-th factor of the \( i \)-th block.

Recalling from Eq. (16) that there are three blocks in it, that is \( \omega_1 \left( x_1, x_2, x_3 \right) \), \( \omega_2 \left( x_4 \right) \) and \( \omega_3 \left( x_5, x_6 \right) \). The first block has two factors, namely \( \varphi_{11} \left( x_1 \right) \) and \( \varphi_{12} \left( x_2, x_3 \right) \). The second block has only one factor, \( \varphi_{21} \left( x_4 \right) \). The last block also has two factors, namely \( \varphi_{31} \left( x_5 \right) \) and \( \varphi_{32} \left( x_6 \right) \). Note that variables \( x_2, x_3 \) are partially separable, while variables \( x_1, x_4, x_5, x_6 \) are all completely separable.

From the above discussion, it is straightforward to show that in Eq. (17), the original target function \( f(x) \) is first divided into several blocks \( \omega_i \left( \cdot \right) \) with global constants \( k_i \). Meanwhile, all binary operators plus (+) are determined, which is based on the separability detection of additively separable \( \omega_i \left( \cdot \right) \) by BiCT technique, where \( i = 1, 2, \cdots, p \). Then, each block \( \omega_i \left( \cdot \right) \) is divided into several factors \( \varphi_{i,j} \left( \cdot \right) \) with global constants \( c_j \). Meanwhile, all binary operators times (×) are determined, which is based on the separability detection of multiplicatively separable \( \varphi_{i,j} \left( \cdot \right) \) by BiCT method, where \( j = 1, 2, \cdots, q_i \). It is clear that the process of block and factor detection does not require any special optimization engine.
3.2. Factor modelling

The mission of symbolic regression is to optimize both the structure and coefficient of a target function that describe an input-response system. In block building programming (BBP) approach, after the binary operators are determined, the original target function \( f(\mathbf{x}) \) is divided into several factors \( \varphi_{i,j}(\cdot) \). So, 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 \( \varphi_{i,j}(\cdot) \) can be optimized while the rest are kept fixed and unchanged.

Without the loss of generality, the factor \( \varphi_{11}(x_1, x_2, \cdots, x_{n,1}) \) in Eq. (17) is taken as an example to illustrate the implementation of the modelling process. The determination of other factors \( \varphi_{i,j}(\cdot) \) share the same way.

1. Let the matrix \( \mathbf{X} \) be a set of \( N \) sampling points for all \( n \) variables, then Eq. (7) is obtained.

2. Keep variables \( x_1, x_2, \cdots, x_{n,1} \) being changed, and let the sampling points of the variables in local block (block 1), \( x_{n,1+1}, x_{n,1+2}, \cdots, x_{n,1} \), be fixed to any two given points \( x_A \) and \( x_B \) \((\forall x_A, x_B \in [a, b])\), respectively. Meanwhile, let the sampling points of variables in other blocks (blocks 2 to \( p \)), namely \( x_{n+1}, x_{n+2}, \cdots, x_{n} \), be fixed to a given points \( x_G \) \((\forall x_G \in [a, b])\). Then, we obtain

\[
\mathbf{X}_1 = \begin{bmatrix}
  x_{11} & \cdots & x_{1,n_1} & x_{1,1,n_1+1} & \cdots & x_{1,n_1}^{(A)} & x_{1,n_1+1}^{(A)} & \cdots & x_{1,n_1}^{(G)} & x_{1,n}^{(G)} \\
  x_{22} & \cdots & x_{2,n_1} & x_{2,1,n_1+1} & \cdots & x_{2,n_1}^{(A)} & x_{2,n_1+1}^{(A)} & \cdots & x_{2,n_1}^{(G)} & x_{2,n}^{(G)} \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
  x_{N,1} & \cdots & x_{N,n_1} & x_{N,1,n_1+1} & \cdots & x_{N,n_1}^{(A)} & x_{N,n_1+1}^{(A)} & \cdots & x_{N,n_1}^{(G)} & x_{2,n}^{(G)} \\
\end{bmatrix},
\]

and

\[
\mathbf{X}_2 = \begin{bmatrix}
  x_{11} & \cdots & x_{1,n_1}^{(B)} & x_{1,1,n_1+1}^{(B)} & \cdots & x_{1,n_1}^{(G)} & x_{1,n_1+1}^{(G)} & \cdots & x_{1,n}^{(G)} \\
  x_{22} & \cdots & x_{2,n_1}^{(B)} & x_{2,1,n_1+1}^{(B)} & \cdots & x_{2,n_1}^{(G)} & x_{2,n_1+1}^{(G)} & \cdots & x_{2,n}^{(G)} \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
  x_{N,1} & \cdots & x_{N,n_1}^{(B)} & x_{N,1,n_1+1}^{(B)} & \cdots & x_{N,n_1}^{(G)} & x_{N,n_1+1}^{(G)} & \cdots & x_{2,n}^{(G)} \\
\end{bmatrix},
\]

3. Let \( \mathbf{\tilde{X}} = \mathbf{X}_1 - \mathbf{X}_2 = [\mathbf{X}_{\text{train}} \ 0] \). Matrix \( \mathbf{X}_{\text{train}} \) is a partition of matrix \( \mathbf{\tilde{X}} \). Then, let \( \mathbf{f}_{\text{train}} \) be the vector of which the \( i \)-th element is the function value of the \( i \)-th row of matrix \( \mathbf{\tilde{X}} \), namely \( \mathbf{f}_{\text{train}} = f(\mathbf{\tilde{X}}) \).
4. Substitute $f_{train}$ and $X_{train}$ into the fit models $y_{train} = k \cdot f^*(x_{train})$. This step could be realized by an existing optimization engine (e.g., GP). Note that, the constant $k$ represent the fitting parameter of the function of variables $x_{n1+1}, x_{n1+2}, \cdots, x_{n1}$, since these variables are set unchanged during this process. We aim to get the optimization model $f^*$, and constant $k$ will be discarded.

Other factors $\varphi_{i,j}$ 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) [14], GPTIPS [21], etc.), swarm intelligence methods (e.g., artificial bee colony programming (ABCP) [8], etc.) and global optimization methods (e.g., low dimensional simplex evolution (LDSE) [13], etc.) are all easy to power BBP.

3.3. Optimization engine

Factors modelling aims to optimize both the structure and coefficients of a factor. This process could be easily realized by global optimization algorithm 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 which involve uni-variable (e.g., $k e^{m_1 x}$, $k (x^m + m_2)$, etc.) and bi-variables (e.g., $k \sin (m_1 x_1 x_2 + m_2)$, $k (e^{m_1 x_1 x_2} + m_2)$, etc.) should be set for function models. Note that, constants $k$ in all function models is the fitting parameter, which are generated by fixed variables of local block (see Section 3.2). Sequence search and optimization method is suitable for global optimization strategy. This means a certain function model will be determined, provided that the fitting error is small enough (e.g., mean square error $\leq 10^{-8}$).

In genetic programming, arithmetic operations (e.g., $+$, $-$, $\times$, $\div$, etc.) and mathematical functions (e.g., $\sin$, $\cos$, $\exp$, $\ln$, etc.) should be pre-established instead of function models. The search process of GP is stochastic. This makes it easy premature convergence. In fact, we can not say which of the two strategies is a better match for BBP. In section 3.4, LDSE and GPTIPS, are both chose to test the performance of BBP.

3.4. Block building programming

Block building programming (BBP) is an extended version of D&C method given by Luo et al. [11]. In fact, it 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 (Section 3.1 to 3.3). Despite different optimization engines for factors modelling might be 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 $S = \{i : i = 1, 2, \cdots, D\}$ for initial variables subscript number, sampling interval $[a, b]$ and the number of sampling points $N$. Generate a sampling set $X \in [a, b] \subset \mathbb{R}^{N \times D}$.

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

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

Step 4. (Factor modelling) For the $j$-th factor $\varphi_{i,j}(\cdot)$ in the $i$-th block, set the variables in sampling set $X$ in blocks $\{1, 2, \cdots, i-1, i+1, \cdots, p\}$ to be fixed to $x_G$, and the variables in factors $\{1, 2, \cdots, j-1, j+1, \cdots, q_j\}$ of the $i$-th block to be fixed to $x_A$ and $x_B$, respectively. Let $\bar{X}^{i,j} = X^i - X^j = \begin{bmatrix} X^{i,j}_{\text{train}} & 0 \end{bmatrix}$ and $f^{i,j}_{\text{train}} = f(\bar{X}^{i,j})$. Then, optimization engine is used.

Step 5. (Global assembling) Global parameter $m_k$, $k = 0, 1, \cdots, p$, can be linearly fitted by equation $f_{\text{train}} = m_0 + \sum_{i=1}^{p} m_i \omega_i(\cdot) = m_0 + \sum_{i=1}^{p} m_i \prod_{j=1}^{q_i} \varphi_{i,j}(X_{\text{train}})$.

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

The proposed method BBP is implemented in MATLAB. In this section, in order to test the performance of BBP, two different optimization engines, LDSE \[13\] and GPTIPS \[21\] are used to power BBP. For the sake of easy use, a Boolean variable is used to turn to 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 us evaluate BBP’s overall capability of ‘structure and coefficient optimization’. Furthermore, computational efficiency is analyzed in Section 4.2.2.

4.1. Driven by LDSE

In this part, we choose a kind of global optimization algorithm, LDSE \[13\], as our optimization engine. LDSE is a hybrid evolutionary algorithm for continuous global optimization. In Table 2, case number, dimension, domain, number of sampling points are denoted as No, Dim, Domain, No.samples, respectively. Additionally, we record time $T_d$ (the block and factor detection), and $T_{BBP}$ (the whole 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 is set as $-50$ and $50$. The population size $N_p$ is set to $N_p = 10 + 10^d$, 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 ($\text{MSE} \leq \varepsilon_{\text{target}}$), and the tolerance (fitting error) is $\varepsilon_{\text{target}} = 10^{-6}$. In order to reduce the effect of randomness, each test case is executed 10 times.

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

| No. | Uni-variable model | Bi-variables model |
|-----|-------------------|--------------------|
| 1   | $k(x^{m_1} + m_2)$ | $k(m_1x_1 + m_2x_2 + m_4)$ |
| 2   | $k(e^{m_1x} + m_2)$ | $k((m_1x_1 + m_2)/(m_3x_2 + m_4))$ |
| 3   | $k\sin (m_1x^{m_2} + m_3)$ | $k(e^{m_1x_1x_2} + m_2)$ |
| 4   | $k\log (m_1x + m_2)$ | $k\sin (m_1x_1 + m_2x_2 + m_3x_1x_2 + m_4)$ |

4.1.2. Numerical results and discussion

Numerical results show that LDSE powered BBP has successfully recovered all the target functions exactly in sense of double precision. Once the uni- and bi-variables models are preseted, 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, we can see that the calculation time of separability detection $T_d$ is almost negligible. This test group show that BBP has a good capability of ‘structure and coefficient optimization’ for highly nonlinear system.

4.2. Driven by GPTIPS

In this part, we choose a kind of genetic programming technique, GPTIPS [21], as the optimization engine. GPTIPS is a MATLAB toolbox based on multi-gene genetic programming. It has been widely used in many researches.
Table 2: Performance of BBP (by LDSE).

| 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              |

To give BBP an overall evaluation of its performance for acceleration, a concept ‘acceleration rate’, $\eta$ is defined as

$$\eta = \frac{T_{GPTIPS}}{T_{BBP}},$$

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 the Section 4.1.1, the target models, search regions and the number of sampling points are set 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 $T$ is set to 100,000. To reduce the influence of randomness, 20 runs are carried out for each case. The termination condition is $\text{MSE} \leq \varepsilon_{\text{target}}$, $\varepsilon_{\text{target}} = 10^{-8}$. 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 get the exact regression model or the approximate model with the default accuracy ($\text{MSE} \leq 10^{-8}$) in almost 20 runs.
except case 2. This situation become even worse with increasing problem size (dimension of the problem). However, the detection of separability helps BBP almost give exact fitting problems for almost all target functions, except case 8 and 9 for its complex coefficients optimization (e.g., the factor $\sin (1.5x_5 - 2x_6)$). Additionally, as the result of $T_{GPTIPS}/T_{BBP}$ shown in Table 3, the computational efficiency raises several orders of magnitude. It is because that the computation time of BBP is determined by the dimension and complexity of each factor, not by the whole target model. That explains why BBP converges much faster than original GPTIPS. Good performance for acceleration, ‘structure optimization’ and ‘coefficient optimization’ show the potential of BBP to be applied in practical applications.

| Case No. | Results of GPTIPS | $T_{GPTIPS}(s)$ | Results of BBP | $T_{BBP}(s)$ | $\eta = T_{GPTIPS}/T_{BBP}$ | Remarks of BBP |
|----------|------------------|----------------|----------------|--------------|----------------------------|----------------|
| 1        | $[5.1, 9.2] \times 10^{-1}$ | $\gg 6.23 \times 10^3$ | $\leq \varepsilon_{\text{target}}$ | $1.512.6$ | $> 4.12$ | 5 runs failed |
| 2        | $\leq \varepsilon_{\text{target}}$ | $323.86$ | $\leq \varepsilon_{\text{target}}$ | $3.94$ | $> 82.2$ | Solutions are all exact |
| 3        | $[1.5, 23.6] \times 10^{-2}$ | $\gg 7.41 \times 10^3$ | $\leq \varepsilon_{\text{target}}$ | $364.3$ | $> 2.03$ | 11 runs failed |
| 4        | $[8.1, 16.2] \times 10^{-2}$ | $\gg 6.16 \times 10^3$ | $\leq \varepsilon_{\text{target}}$ | $903.87$ | $> 6.8$ | 4 runs failed |
| 5        | $[4.5, 7.3] \times 10^{-1}$ | $\gg 6.68 \times 10^3$ | $\leq \varepsilon_{\text{target}}$ | $26.65$ | $> 250.65$ | Solutions are all exact |
| 6        | $[2.31, 9.6] \times 10^{-2}$ | $\gg 6.31 \times 10^3$ | $\leq \varepsilon_{\text{target}}$ | $4416.07$ | $> 1.67$ | 7 runs failed |
| 7        | $[1.22, 3.96] \times 10^{-1}$ | $\gg 8.52 \times 10^3$ | $\leq \varepsilon_{\text{target}}$ | $11.57$ | $> 736.39$ | Solutions are all exact |
| 8        | $[2.1, 37.2] \times 10^{-1}$ | $\gg 7.13 \times 10^3$ | $[9.16, 32.3] \times 10^{-2}$ | $\gg 1.3721 \times 10^4$ | None | All runs failed |
| 9        | $[5.4, 56.3] \times 10^{-2}$ | $\gg 6.24 \times 10^3$ | $[1.68, 12.9] \times 10^{-2}$ | $\gg 6.63 \times 10^4$ | None | All runs failed |
| 10       | $[5.86, 99.16] \times 10^{-1}$ | $\gg 7.36 \times 10^3$ | $\leq \varepsilon_{\text{target}}$ | $11.62$ | $> 708.26$ | Solutions are all exact |

5. Conclusion

A new approach, block building programming (BBP), for symbolic regression has been presented. The method is an improved version of D&C method [11]. BBP divides the original complex target model into several blocks, and further into factors. To detect the separability, a special method, block and factor detection, is also developed. 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 modelling complex nonlinear systems in various research fields.
As a future work, it is planned to generalize the mathematical form of the separable function. Here, we give an example of the flow over a circular cylinder, which is a classical problem in fluid dynamics [2]. A valid stream function for the inviscid, incompressible flow over a circular cylinder of radius \( R \) is

\[
\psi = (V_\infty r \sin \theta) \left( 1 - \frac{R^2}{r^2} \right) + \frac{\Gamma}{2\pi} \ln \frac{r}{R}
\]  \hspace{1cm} (21)

Note that, in Eq. (21), partially separable variables \((r, R)\) appear twice in Eq. (21). In other words, variables \((r, R)\) have two sub-functions, namely \( \varphi_1 (r, R) = 1 - R^2/r^2 \) and \( \varphi_2 (r, R) = \ln (r/R) \). This makes Eq. (21) inconsistent with the definition of the separable function. Such complicated condition would be analyzed in future studies.

Acknowledgements

This work has been supported by the National Natural Science Foundation of China (Grant No. 11532014).

Appendix A. Proof of Theorem 3

Proof. Considering that the binary operator \( \otimes \) are only plus (+) and times (\( \times \)), without significant loss of generality, constant \( c_i \) in Eq. (6) is not discussed in this proof.

Firstly, consider the position of each binary operator plus (+). Without loss of generality, we assume that, in Eq. (6), the first binary operator plus (+) appears in the middle of the sub-functions of variables \( x_1, x_2, \cdots, x_{n_1} \) and \( x_{n_1+1}, x_{n_1+2}, \cdots, x_n \), that is

\[
f (\mathbf{x}) = \omega_1 (x_1, x_2, \cdots, x_{n_1}) + \tilde{\omega}_1 (x_{n_1+1}, x_{n_1+2}, \cdots, x_n).
\]  \hspace{1cm} (A.1)

The second binary operator plus (+) appears in the middle of the sub-functions of variables \( x_1, x_2, \cdots, x_{n_1+n_2} \) and \( x_{n_1+n_2+1}, x_{n_1+n_2+2}, \cdots, x_n \), that is

\[
f (\mathbf{x}) = \omega_1 (x_1, \cdots, x_{n_1}) + \omega_2 (x_{n_1+1}, \cdots, x_{n_1+n_2}) + \tilde{\omega}_2 (x_{n_1+n_2+1}, \cdots, x_n).
\]  \hspace{1cm} (A.2)

The position of the rest binary operators plus (+) could be determined in the same way. Then, we obtain

\[
\tilde{f} (\mathbf{x}) = \sum_{i=1}^{m} \omega_i (I^{(i)} \mathbf{x}),
\]  \hspace{1cm} (A.3)
where $\mathbf{I}^{(i)} \in \mathbb{R}^{n_i \times n}$ is the partitioned matrix of the identity matrix $\mathbf{I} \in \mathbb{R}^{n \times n}$, \( \sum_{i=1}^{m} n_i = n \). Provided that the global constant $k_i$ is considered, the left-hand side of the Eq. (17) can be obtained.

Secondly, we decide the position of each binary operator times ($\times$). Consider the first sub-function $\omega_1 (\mathbf{I}^{(1)} \mathbf{x})$ in Eq. (17). Without loss of generality, the global constants are not considered here. We assume that, the first binary operator times ($\times$) appears in the middle of the sub-functions of variables $x_1, x_2, \cdots, x_{n_1}$ and $x_{n_1+1}, x_{n_1+2}, \cdots, x_{n_1}$, that is

$$\omega_1 (\mathbf{I}^{(1)} \mathbf{x}) = \varphi_{11} (x_1, x_2, \cdots, x_{n_1}) \cdot \tilde{\varphi}_1 (x_{n_1+1}, x_{n_1+2}, \cdots, x_{n_1}). \quad (A.4)$$

The second binary operator times ($\times$) appears in the middle of the sub-functions of variables $x_1, x_2, \cdots, x_{n_1+n_2}$ and $x_{n_1+n_2+1}, x_{n_1+n_2+2}, \cdots, x_{n_1}$, that is

$$\omega_1 (\mathbf{I}^{(1)} \mathbf{x}) = \varphi_{11} (x_1, \cdots, x_{n_1}) \cdot \varphi_{12} (x_{n_1+1}, \cdots, x_{n_1+n_2}) \cdot \tilde{\varphi}_2 (x_{n_1+n_2+1}, \cdots, x_{n_1}). \quad (A.5)$$

The position of the rest binary operators times ($\times$) could be determined in the same way. Then, $\omega_1 (\mathbf{I}^{(1)} \mathbf{x})$ could be rewritten as $\omega_1 (\mathbf{I}^{(1)} \mathbf{x}) = \prod_{j=1}^{q_i} \varphi_{i,j} (\mathbf{I}_j^{(1)} \mathbf{x})$, where $\mathbf{I}_j^{(1)} \in \mathbb{R}^{n_{i,j} \times n}$ is the partitioned matrix of the $\mathbf{I}^{(1)}$, namely $\mathbf{I}^{(1)} = \begin{bmatrix} \mathbf{I}_1^{(1)} & \mathbf{I}_2^{(1)} & \cdots & \mathbf{I}_{q_i}^{(1)} \end{bmatrix}$, $\sum_{j=1}^{q_i} n_{1,j} = n_1$. Hence, for arbitrary sub-function $\omega_i (\mathbf{I}^{(i)} \mathbf{x})$, we have

$$\omega_i (\mathbf{I}^{(i)} \mathbf{x}) = \prod_{j=1}^{q_i} \varphi_{i,j} (\mathbf{I}_j^{(i)} \mathbf{x}), \quad (A.6)$$

where $\mathbf{I}_j^{(i)} \in \mathbb{R}^{n_{i,j} \times n}$ is the partitioned matrix of the $\mathbf{I}^{(i)}$, namely $\mathbf{I}^{(i)} = \begin{bmatrix} \mathbf{I}_1^{(i)} & \mathbf{I}_2^{(i)} & \cdots & \mathbf{I}_{q_i}^{(i)} \end{bmatrix}$, $\sum_{j=1}^{q_i} n_{i,j} = n_i$. Substitute Eq. (A.6) into Eq. (A.3). Provided that the global constant $c_i$ is considered, the right-hand side of the Eq. (17) can be obtained.

Appendix B. 10 target models of numerical experiments

The target models tested in Section 4 are given as follows:
Case 1.  $f(x) = 1.2 + 10 \sin(2x_1 - x_3) - 3x_2^2$, where $x_i \in [-3, 3], i = 1, 2, 3$.

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

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

Case 4.  $f(x) = \frac{5\sin(3x_1x_2)}{x_3 + x_4}$, where $x_i \in [-3, 3], i = 1, 2, 3, 4$.

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

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

Case 7.  $f(x) = \frac{10\sin(x_1x_2) - x_3}{x_4 + x_5}$, where $x_i \in [-3, 3], i = 1, 2, \cdots, 5$.

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

Case 9.  $f(x) = \frac{100\cos(x_3x_4)}{e^{1x_2+2}} \sin (1.5x_5 - 2x_6)$, where $x_i \in [-3, 3], i = 1, 2, \cdots, 6$.

Case 10.  $f(x) = \frac{x_1 + x_2}{x_3} + x_4 \sin (x_5x_6)$, where $x_i \in [-3, 3], i = 1, 2, \cdots, 6$.

References

[1] A.H. Alavi, H. Hasni, I. Zaabar, N. Lajnef, A new approach for modeling of flow number of asphalt mixtures, Arch. Civ. Mech. Eng. 17 (2017) 326-335.

[2] J.D. Anderson, Fundamentals of Aerodynamics, Fifth Edition, MacGraw-Hill, New York, USA, 2011.

[3] J.D. Anderson, Hypersonic and High-Temperature Gas Dynamics, Second Edition, American Institute of Aeronautics and Astronautics, Inc., Virginia, 2006.

[4] X. Dong, W. Dong, Y. Yi, Y. Wang, X. Xu, The Recent Developments and Comparative Analysis of Neural Network and Evolutionary Algorithms for Solving Symbolic Regression, Intelligent Computing, Springer, 2015, pp. 703-714.
[5] Z. Gan, T.W.S. Chow, W.N. Chau, Clone selection programming and its application to symbolic regression, Exp. Syst. Appl. 36 (2) (2009) 3996-4005.

[6] 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.

[7] I. Icke, J. C. Bongard, Improving genetic programming based symbolic regression using deterministic machine learning, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE-CEC, 2013, pp. 1763-1770.

[8] D. Karaboga, C. Ozturk, N. Karaboga, B. Gorkemli, Artificial bee colony programming for symbolic regression, Inform. Sci. 209 (2012) 1-15.

[9] H. Kaydani, A. Mohebbi, M. Eftekhar, Permeability estimation in heterogeneous oil reservoirs by multi-gene genetic programming algorithm, J. Petrol. Sci. Eng. 123 (2014) 201-206.

[10] J.R. Koza, Genetic programming: on the programming of computers by means of natural selection, MIT Press, Cambridge, MA, USA, 1992.

[11] C. Luo, C. Chen, Z. Jiang, A divide and conquer method for symbolic regression, arXiv: 1705.08061.

[12] C. Luo, Z. Hu, S. Zhang, Z. Jiang, Adaptive space transformation: An invariant based method for predicting aerodynamic coefficients of hypersonic vehicles, Eng. Appl. Artif. Intel. 46 (2015) 93-103.

[13] C. Luo, B. Yu, Low dimensional simplex evolution a new heuristic for global optimization, J. Global Optim. 52 (1) (2012) 45-55.

[14] C. Luo, S. Zhang, Z. Jiang, Parse-matrix evolution for symbolic regression, Eng. Appl. Artif. Intel. 25 (6) (2012) 1182-1193.

[15] T. McConaghy, FFX: Fast, scalable, deterministic symbolic regression technology, Genetic Programming Theory and Practice IX, Springer, 2011, pp. 235-260.
[16] M. O’Neil, C. Ryan, Grammatical evolution, IEEE Evolut. Comput. 5 (2001) 349-358.

[17] 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.

[18] 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.

[19] D.P. Raymer, Aircraft Design: A Conceptual Approach, Fifth Edition, American Institute of Aeronautics and Astronautics, 2012.

[20] J.L. Salmeron, S.A. Rahimi, A.M. Navali, A. Sadeghpour, Medical diagnosis of Rheumatoid Arthritis using data driven PSOF CM with scarce datasets, Neurocomputing 232 (2017) 104-112.

[21] 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 MultiConference of Engineers and Computer Scientists, 2010, Hong Kong, China.

[22] 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.

[23] L.F. dal P. 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.

[24] 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.

[25] A. Worm, Prioritized Grammar Enumeration: A novel method for symbolic regression, Binghamton University - State University of New York, 2016 Ph.D. thesis.

[26] 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.