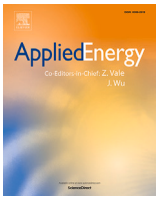
Applied Energy 392 (2025) 125893

Contents lists available at ScienceDirect

Applied Energy

journal homepage: www.elsevier.com/locate/apen



Data-driven parametric optimization for pre-calibration of internal combustion engine controls

Matteo Meli [a,1,*], Zezhou Wang [b,d,1], Stefan Sterlepper [a], Mario Picerno [c], Stefan Pischinger [a,b]

[a] Chair of Thermodynamics of Mobile Energy Conversion Systems, RWTH Aachen University, Aachen, Germany
[b] FEV Europe GmbH, Aachen, Germany
[c] Teaching and Research Area Mechatronics in Mobile Propulsion, RWTH Aachen University, Aachen, Germany
[d] Institute of Mobile Machines, Karlsruhe Institute of Technology, Karlsruhe, Germany

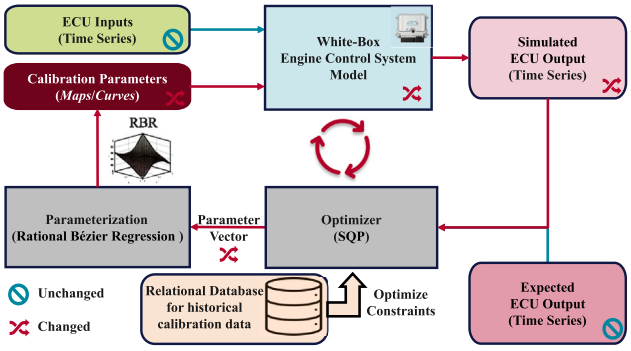
HIGHLIGHTS

- A novel engine pre-calibration method using Rational Bézier Regression (RBR) is introduced.
- Model-in-the-Loop optimization reduces hardware reliance and speeds up calibration.
- A newly defined historical calibration database enhances optimization constraints.
- Control parameter vector transforms Multi-Objective Optimization (MOO) into single-objective optimization.
- The method achieves high shape similarity in pre-calibrated lookup tables and takes 60 h in total.

ARTICLE INFO

Keywords:
Engine control
Model-based calibration
Model-in-the-Loop
Parametric optimization
Relational database
Virtual calibration

GRAPHICAL ABSTRACT



ABSTRACT

This paper presents an efficient pre-calibration method for combustion engine controls. In particular, it focuses on the initial shaping of multiple Lookup Tables (LUTs) within LUT-based Multiple-Input Single-Output (MISO) engine control systems. The approach addresses the increasing complexity of engine software, the rising number of calibration variables, and the time pressures prevalent in automotive development. Employing a white-box Model-in-the-Loop (MiL) optimization reduces the demands on hardware reliance and optimization time compared to conventional engine calibration techniques. The white-box model enables the pre-calibration of LUTs using known system inputs, expected system outputs, and the control system model structure. To optimize the white-box control system model, LUTs are parametrized through Rational Bézier Regression (RBR), facilitating Sequential Quadratic Programming (SQP) for optimization. RBR, which includes both Rational Bézier Curve Regression (RBCR) and Rational Bézier Surface Regression (RBSR), allows for flexible and smooth shaping of 1D and 2D LUTs with a unified and few number of parameters. The pre-calibration process is further improved using historical calibration data from various vehicle variants stored in a relational database. This ensures that the final outputs of the LUT-based MISO control system closely approximate the expected target outputs with high shape similarity. The proposed method is exemplified using an oil temperature control model from a state-of-the-art

* Corresponding author.
Email addresses: meli_ma@tme.rwth-aachen.de (M. Meli), zezhou.wang@kit.edu (Z. Wang), sterlepper_s@tme.rwth-aachen.de (S. Sterlepper), picerno_ma@mmp.rwth-aachen.de (M. Picerno), pischinger_s@tme.rwth-aachen.de (S. Pischinger).
[1] Matteo Meli and Zezhou Wang contributed equally to this work as co-first authors.

https://doi.org/10.1016/j.apenergy.2025.125893
Received 16 October 2024; Received in revised form 25 February 2025; Accepted 5 April 2025
0306-2619/© 2025 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

hybrid powertrain with an internal combustion engine. The results demonstrate Pearson Correlation Coefficients (PCCs) exceeding 0.8 between target and pre-calibrated LUTs, indicative of high shape similarity. Additionally, the system outputs of pre-calibrated control system models closely match expected system outputs with an R^2 value of 0.9385. This underscores the practical applicability of the proposed pre-calibration method for internal combustion engine controls.

Definitions and Acronyms

$\mathfrak{B}$	Bernstein Polynomial
C	Control Point of RBR
d_k	Iterative Step of SQP
$f(\mathbf{x})$	Target Optimization Function
$\mathbf{g}^*(\mathbf{x})$	Active Sets
H_f	Hessian Matrix of Target Optimization Function
$\mathcal{J}$	Cost Function
k	Dimensions of RBR Input Vector of X-Axis
$\mathcal{L}$	Lagrangian Function
$\mathbf{L}$	RBR Lower Bound Vector
m	Dimensions of RBR Input Vector of Y-Axis
$m\%$	Margins of RBR Initial Setups
M	Dimensions of RBR Parameter Vector
n	RBR Degree/Bernstein Polynomial Degree
$\hat{O}$	Expected ECU Output
O	Simulated ECU Output
p	RBR Parameter
$\mathbf{p}^*$	Optimal RBR Parameter Vector
$\mathfrak{Q}$	Park Transformation
$\mathbf{U}$	RBR Upper Bound Vector
w	Weight of RBR
$\mathbf{x}$	Iteration Point Vector of X-Axis
x_k	Iteration Point for SQP
x^*, y^*	Rotational X-Axis, Y-Axis
$\mathcal{X}$	Input for Global Normalization
$\mathcal{X}^*$	Globally Normalized Input
$\mathbf{Z}$	RBR Vector
$\mathcal{Z}$	Z-Axis of RBR
$\hat{\mathcal{Z}}$	Target Value of Rational Bézier Regression
$\mathcal{Z}^*$	Globally Normalized Output
ϵ	Rotational Degree of Freedom for RBSR
μ	Mean Value
σ	Standard Deviation
∇	Gradient of Selected Function
DoE	Design of Experiment
ECU	Engine Control Unit
EiL	Engine-in-the-Loop
ERD	Entity Relationship Diagram
ICE	Internal Combustion Engine
IP	Inner Point
KKT	Karush-Kuhn-Tucker
LUT	Lookup Table
MAE	Maximum Absolute Error
MBC	Model-Based Calibration
MiL	Model-in-the-Loop
MISO	Multiple-Input Single-Output
MLP	Multilayer Perceptron
MOO	Multi-Objective Optimization
NLCP	Non-Linearly Constrained Problem
nRBSR	Normal Rational Bézier Surface Regression
OP	Optimization Procedure
PCC	Pearson Correlation Coefficient
PK	Primary Key
PSO	Particle Swarm Optimization
QP	Quadratic Programming
qRBCR	Quadratic Rational Bézier Curve Regression
qRBSR	Quadratic Rational Bézier Surface Regression
RBC	Rational Bézier Curve
RBCR	Rational Bézier Curve Regression
RBR	Rational Bézier Regression
RBS	Rational Bézier Surface
RBSR	Rational Bézier Surface Regression
RMSE	Root Mean Square Error
rRBSR	Rotational Rational Bézier Surface Regression
SQL	Structured Query Language
SQP	Sequential Quadratic Programming

1. Introduction

Internal Combustion Engine (ICE) calibration is the process of optimizing engine performance to meet engineering targets by adjusting calibration data within engine control systems. The ICE calibration targets are driven by compliance with emission standards. For this, it is crucial to operate the engine near optimal operating points, particularly with regard to fuel consumption. Today's engine control systems are integrated into the Engine Control Unit (ECU), where calibration is achieved by tuning the calibration data within the ECU software. Among the most critical calibration data are the 1D and 2D Lookup Tables (LUTs), referred to as *Curves* and *Maps*. In some engine control systems, multiple *Curves* or *Maps* are interdependent, meaning that a change in one LUT's data can affect the inputs and outputs of others, potentially impacting the overall performance of the engine control system. Therefore, efficiently and accurately calibrating multiple LUTs simultaneously is a complex challenge.

Traditionally, experienced calibration engineers calibrate LUT data points requiring extensive measurements from test bench-based Engine-in-the-Loop (EiL) processes or real driving scenarios [1,2]. This process is often labor- and time-intensive. With increasing pressure to reduce time-to-market and the need to calibrate an increasing number of engine variants, the limitations of hardware-based calibration methods, such as EiL and real driving, have become more apparent. Consequently, virtual calibration methods, such as Model-Based Calibration (MBC), are increasingly being adopted to reduce reliance on physical hardware [3]. MBC enables rapid execution of numerous tests through simulation models and allows performance optimization in a virtual environment, significantly shortening development cycles [4].

MBC can be implemented using black-box surrogate models of the engine control system. For instance, the development of a surrogate model for the engine control system using a Multilayer Perceptron (MLP) offers satisfactory speed and accuracy [5]. However, MLP is only applicable to steady-state control systems, which can be modeled using discrete system inputs and outputs. If a white-box model, such as a corresponding Simulink model, is available instead, MBC can be applied. This paper explores the application of the white-box engine control system's model in MBC.

To more efficiently calibrate multiple LUTs simultaneously, several calibration methods combining the Design of Experiments (DoE) with

Model-Based Calibration (MBC) have been proposed [6]. DoE selects representative experimental points that cover a broad range of engine operating conditions while minimizing the number of experiments, thus accelerating the calibration process [7]. Afterward, Multi-Objective Optimization (MOO) techniques, such as the Pareto front, are used to optimize the discrete data points within multiple LUTs [8]. However, DoE requires calibration engineers to have an in-depth understanding of the engine control system's mechanisms, such as physical principles and engineering constraints [9]. Inappropriate selection of experimental points may negatively affect the final calibration outcome or lead to excessive testing.

Additionally, while the data points in the LUTs can be optimized in a discrete form, if there are too many data points in the LUTs, the optimization process can become significantly slower [10]. The varying number of data points across different LUTs can also complicate the integration of these discrete data points into the engine control system model.

Existing methodologies do not efficiently leverage historical calibration data to guide LUT parameter initialization and constraint definition. This gap in the literature motivates the need for a more data-driven, automated approach that seamlessly integrates historical calibration data into the LUT optimization workflow. Instead of relying solely on DoE for the same engine control system [11], historical calibration data from previous powertrain calibration projects can be utilized. This approach reduces the need for calibration engineers to have an in-depth understanding of the engine control system's internal mechanisms. It supports the calibration of new LUTs, as these historical data are based on similar operational mechanisms and have already been validated. This paper will first establish a database to store these historical calibration data. By applying a data-driven approach with the historical calibration data, the range of new LUT values can be appropriately scaled, improving overall calibration efficiency. Additionally, RBR has not been previously applied in the context of white box LUT-based ICE calibration.

Moreover, as *Curves* and *Maps* are essentially one-dimensional vectors and two-dimensional matrices, they can be modeled using parametric curves or surfaces [5]. For the simultaneous calibration of LUTs, it is highly advantageous to conduct a pre-calibration. This step ensures near-optimal initial LUT shapes, thereby minimizing the need for extensive hardware-based calibration in later stages.

This paper will adopt a novel parameterization method called Rational Bézier Regression (RBR), which models *Curves* or *Maps* using a unified set of parameters. RBR is based on rational Bézier curves or surfaces and can be divided into two sub-functions: Rational Bézier Curve Regression (RBCR) for fitting *Curves* and Rational Bézier Surface Regression (RBSR) for fitting *Maps*.

Subsequently, the selected LUTs within the engine control system will be parameterized using RBR, and the parameters controlling the shapes of these LUTs will be concatenated into one parameter vector. This approach simplifies the multi-objective optimization problem for MBC into a single-objective problem. The parameter vector that concatenates the parameterized LUTs becomes the new target for optimization, replacing the fully vectored entries. By combining efficient optimization methods and leveraging the historical calibration data from the database to optimize constraints, MBC can be performed in a Model-in-the-Loop (MiL) virtual environment for pre-calibration.

A typical LUT-based Multiple-Input Single-Output (MISO) engine control system will be used to validate the pre-calibration methodology. During the optimization process, only the system's inputs and outputs will be known. The pre-calibrated LUTs are obtained after optimization by computing the pre-calibrated parameter vectors back into LUTs using RBR.

The following sections are organized as follows. Section 2 presents the theories required for pre-calibration, including the database, RBR, and the applied optimization methods. Section 3 introduces a methodology for optimizing the constraints of the optimization problem using the calibration database. Section 4 describes the experimental setups and the core process of MBC based on MiL. The results of the pre-calibration are presented in Section 5. Finally, Section 6 concludes the paper.

2. Theories

This paper explores key theories relevant to the research. It begins with an analysis of ECU calibration data, demonstrating the efficiency of a relational database structure for data querying and manipulation. It introduces a parametric modeling approach, Rational Bézier Regression (RBR), which includes Rational Bézier Curve Regression (RBCR) and Rational Bézier Surface Regression (RBSR), detailing their formulas and characteristics. Furthermore, the paper discusses the Sequential Quadratic Programming (SQP) algorithm for solving the Non-Linearly Constrained Problem (NLCP).

2.1. Relational database for ECU calibration data

This section will use ECU calibration data as examples to design the structure of a relational database based on datasets from various existing production vehicles with internal combustion engines. Within the engine control software, there are five main data types, which are:

- *Curve*: 1D-LUT (LUT with one input and one output)
- *Map*: 2D-LUT (LUT with two inputs and one output)
- *Value*: numerical constant
- *ValueBlock*: string characters
- *AxisPts*: fixed axis points for *Curves* or *Maps*

The relational database is based on the relational model [12]. Like Excel worksheets, it uses two-dimensional tables of rows and columns to store and manage data. Each row of the data represents a "Tuple", a single item of the dataset; Each column of the data represents an "Attribute", a feature that every tuple in the dataset has; both columns and rows are organized in a table, which is called "Relation".

Each set of attributes within a table is made selectable with a unique Primary Key (PK). These PKs can connect the tables in the database. With Structured Query Language (SQL), all datasets in different tables can be queried and manipulated [13].

To explain the relationship and structure of the tables in the relational database for ECU calibration data intuitively, the Entity Relationship Diagram (ERD) is used [14]. There are three types of relationships among tables: "One-to-One", "One-to-Many", and "Many-to-Many".

Fig. 1 shows the ERD that is employed for the structure of the relational database in this paper. Due to the different data types of ECU datasets, it is not possible to store various types of data in just one table. In addition, it is suggested that the small data queries and storage can improve the efficiency of the queries [15]. Therefore, it can be seen that for each data type, multiple tables are required for storage. To avoid redundancy, the common features of each calibration data are stored in a central feature table *manual*, such as the technical specifications of the vehicles, the production information, etc. Whereas the data content of maps, curves AxisPoints, Values, and ValueBlocks are stored in individual tables, containing the data points with units, the limitations of axes, etc. It is recommended that the number of tables for all data types be consistent, which can help reduce the amount of code when querying data.

2.2. Rational Bézier regression

Rational Bézier Regression (RBR) is a unified parametric regression approach that includes Rational Bézier Curve Regression (RBCR) for one-dimensional curve fitting and Rational Bézier Surface Regression (RBSR) for two-dimensional surface fitting. For LUT fitting in the ECU, RBCR, also known as the shape-based algorithm, has been shown to outperform small neural networks and polynomials [5]. However, due to the limitation of RBCR requiring a perfect initial shape for surface fitting, the RBSR proposed in this paper will overcome this drawback.

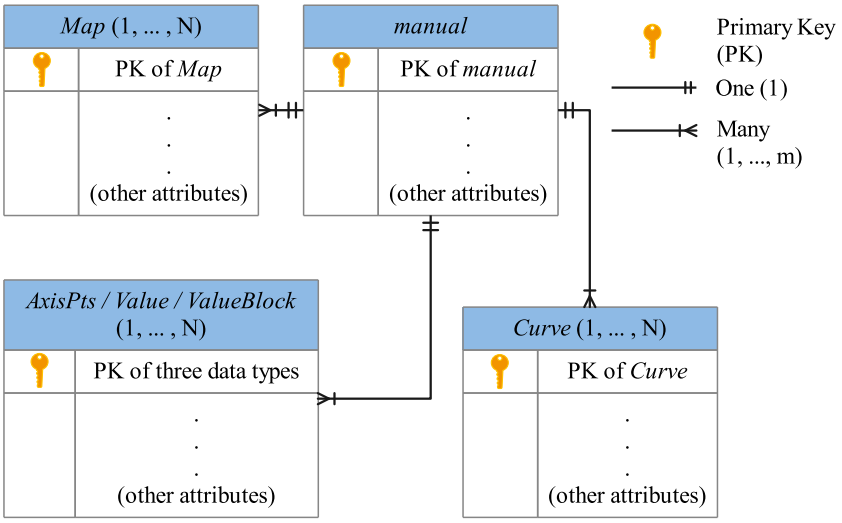


Fig. 1. The structure of the relational database for ECU calibration data.

RBCR and RBSR share the same mathematical foundation but differ in their application and formulation: RBCR is tailored for 1D LUTs (Curves), whereas RBSR extends this concept to 2D LUTs (Maps) with additional degrees of freedom to control surface adaptation. Both RBCR and RBSR maintain connectivity while distorting space, preserving the monotonicity and smoothness of parametric surfaces and curves. They offer sufficient degrees of freedom for effective shape adaptation with minimal optimization parameters, reducing computational effort. These parameters are well-defined, allowing easy setting of bounds to control the transformation behavior. Additionally, RBR excels at fitting conic sections.

Next, RBCR and RBSR will be introduced in detail, as well as their characteristics.

2.2.1. Rational Bézier curve regression

Rational Bézier Curve Regression (RBCR) is a parametric regression curve, which is based on rational Bézier curve. Bernstein polynomial is the fundamental of rational Bézier curve. Let $n, i \in \mathbb{N}$, with $i \leq n$. The ith Bernstein polynomial of degree n with the binomial coefficient $\binom{n}{i}$ is defined as below [16]:

$$\mathfrak{B}_{i,n}(x) = \binom{n}{i} x^i (1-x)^{n-i} \tag{1}$$

The Rational Bézier Curve (RBC) $\mathcal{Z}_{RBC}(x) : [0,1] \mapsto [0,1]$ is defined by:

$$x \mapsto \mathcal{Z}_{RBC}(x) = \frac{\sum_{i=0}^{n} \mathfrak{B}_{i,n}(x) C_i w_i}{\sum_{i=0}^{n} \mathfrak{B}_{i,n}(x) w_i} \tag{2}$$

where $C_i \in [0,1]$ are the control points and $w_i \in \mathbb{R}_0^+$ denote the weights associated with each control point. To simplify Eq. (2), the first control point C_0 is set to 0, and the last control point C_n is set to 1; similarly, the weight of the first control point w_0 and the weight of the last control point w_n are all set to 1. For more intuitive representation, the unknown control points C_i and its weight w_i can be replaced by parameters $p_{(2i-1)} \in [0,1]$ and $p_{(2i)} \in \mathbb{R}^+$, where $i \in \{1, 2, \dots, n-1\}$. To increase the flexibility of the parametric curve and its range, a scaling parameter $p_{(2n-1)}$ and a parameter for displacement in the direction of the z-axis $p_{(2n)}$ are added to RBCR. Thus, there are $2n$ parameters for RBCR of degree n.

Then, the general RBCR $\mathcal{Z}_{RBCR}(x) : [0,1] \mapsto \mathbb{R}$ can be written as:

$$x \mapsto \mathcal{Z}_{RBCR}(x) = p_{(2n-1)} \frac{\sum_{i=1}^{n-1} \mathfrak{B}_{i,n}(x) p_{(2i-1)} p_{(2i)} + x^n}{(1-x)^n + \sum_{i=1}^{n-1} \mathfrak{B}_{i,n}(x) p_{(2i)} + x^n} + p_{(2n)} \tag{3}$$

From Eq. (3), only four parameters are needed to control the shape of the curve with a quadratic ($n = 2$) Rational Bézier Curve Regression (qRBCR). This effect is shown in Fig. 2 with different values for parameters p_1 and p_2. As p_2 increases, parts of the y-axis will be squeezed, and the gradient of the parametric curve in the definition domain close to 1 will become larger. And as p_1 increases, the squeezed parts of the y-axis will move towards the maximum value of the range. When $p_2 = 0$, no matter how p_1 changes, the parametric curve will not change.

Algorithm A in Appendix A demonstrates $\mathbf{Z}_{RBCR} \in \mathbb{R}^k$ with input vector $\mathbf{X} \in \mathbb{R}^k$ and parameter vector $\mathbf{P} \in \mathbb{R}^{2n}$, where $k, n \in \mathbb{N}$. Algorithm $\mathfrak{B}$ is from Eq. (1).

2.2.2. Rational Bézier surface regression

Similar to RBCR, Rational Bézier Surface Regression (RBSR) is a parametric regression surface based on the Rational Bézier Surface (RBS). Bernstein polynomial is also the fundamental of RBS, which is defined in Eq. (1).

Then, RBS $\mathcal{Z}_{RBS}(x, y) : [0,1] \times [0,1] \mapsto [0,1]$ with degree n is defined by:

$$(x, y) \mapsto \mathcal{Z}_{RBS}(x, y) = \frac{\sum_{j=0}^{n} \sum_{i=0}^{n} \mathfrak{B}_{i,n}(x) \mathfrak{B}_{j,n}(y) C_{ij} w_{ij}}{\sum_{j=0}^{n} \sum_{i=0}^{n} \mathfrak{B}_{i,n}(x) \mathfrak{B}_{j,n}(y) w_{ij}} \tag{4}$$

where $C_{ij} \in [0,1]$ are the control points and $w_{ij} \in \mathbb{R}_0^+$ are the weights of each control point. To simplify Eq. (4), the first control point C_{00} is set to 0 and the last control point C_{nn} is set to 1; the weight of the first control point w_{00} and the weight of the last control point w_{nn} are all set to 1. Similar to RBCR, for a more intuitive representation, each control point $C_{ij} \in [0,1]$ and its weight $w_{ij} \in \mathbb{R}_0^+$ of RBS with n degree can be replaced with the parameters in Eq. (5):

$$C_{ij} = \begin{cases} p_{2j(n+1)+2i+1}, & (i,j) \notin \{(0,0),(n,n)\} \\ 0, & (i,j) = (0,0) \\ 1, & (i,j) = (n,n) \end{cases}, \qquad w_{ij} = \begin{cases} p_{2j(n+1)+2i+2}, & (i,j) \notin \{(0,0),(n,n)\} \\ 1, & (i,j) = (0,0) \\ 1, & (i,j) = (n,n) \end{cases}. \tag{5}$$

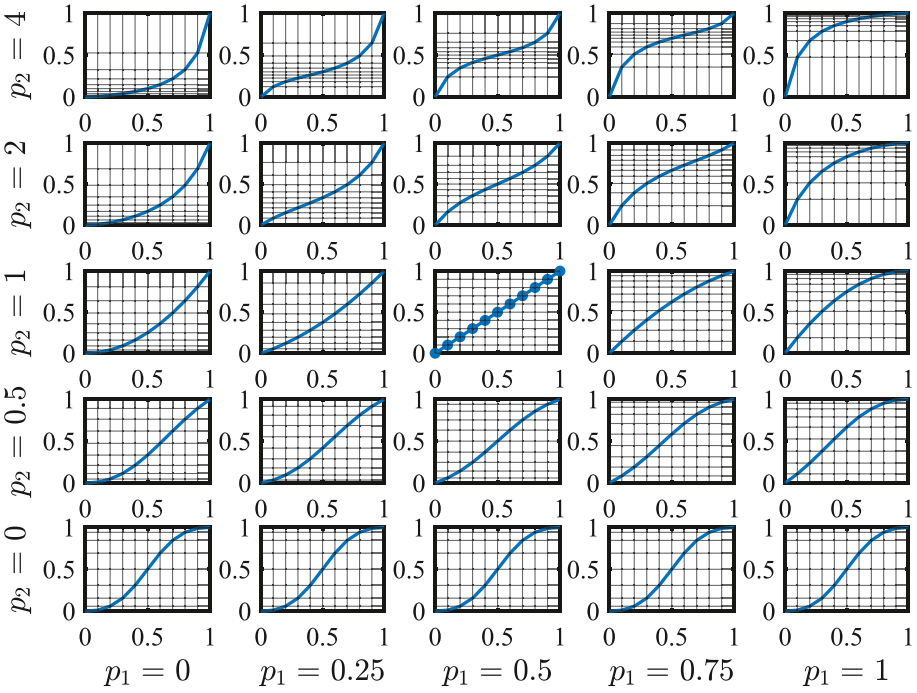


Fig. 2. The qRBCR with different p_1 and p_2, where $p_3 = 1$ and $p_4 = 0$. The horizontal axis of each subplot represents x, and the vertical axis represents $\mathcal{Z}_{qRBCR}$.

For the same purpose of RBCR parameters $p_{(2n-1)}$ and $p_{(2n)}$, a scaling parameter $p_{(2n^2+4n-1)}$ and a displacement parameter in the z-axis direction $p_{(2n^2+4n)}$ are added to RBSR.

Adding the rotation degree of freedom can improve the performance of the parametric surface fitting, as demonstrated in Section 5.1. In the following pages, RBSR will represent rotational RBSR (rRBSR), the RBSR with the rotation degree of freedom, to replace the previous normal RBSR (nRBSR), the RBSR without the rotation degree of freedom. Consequently, an extra parameter ϵ for rotational angle is added, which is represented as the parameter $p_{(2n^2+4n+1)}$ for nth RBSR. Before applying the input vectors $\mathbf{X}$ and $\mathbf{Y}$ into RBSR, $\mathbf{X}$ and $\mathbf{Y}$ are transformed with Park transformation first [17]:

$$\mathfrak{Q}(\epsilon) = \begin{bmatrix} \cos(\epsilon) & -\sin(\epsilon) \\ \sin(\epsilon) & \cos(\epsilon) \end{bmatrix}, \quad \begin{bmatrix} x^* \\ y^* \end{bmatrix} = \mathfrak{Q}(-\epsilon) \begin{bmatrix} x \\ y \end{bmatrix} = \mathfrak{Q}^{-1}(\epsilon) \begin{bmatrix} x \\ y \end{bmatrix} \tag{6}$$

Then, the rotational RBSR $\mathcal{Z}_{RBSR}(x^*, y^*) : [0,1] \times [0,1] \mapsto \mathbb{R}$ with degree n with parameters using Park transformation in Eq. (6) can be written as:

$$(x^*, y^*) \mapsto \mathcal{Z}_{RBSR}(x^*, y^*) = p_{(2n^2+4n-1)} \mathcal{Z}_{RBS}(x^*, y^*) + p_{(2n^2+4n)} \tag{7}$$

where $\mathcal{Z}_{RBS}(x^*, y^*)$ is the RBS in Eq. (4) with the parameters' replacement from Eq. (5). The parameters $p_{(2n^2+4n-1)}$ and $p_{(2n^2+4n)}$ determine the range of RBSR.

According to Eq. (7), there are 17 parameters for quadratic RBSR (qRBSR). Fig. 3 demonstrates how control points' positions and weights form a qRBSR. In this example, the positions of control points C_{10} and C_{01} are both set to 0.6, yet their weights differ, thus affecting the edges of the surface they control differently: the edge controlled by C_{10} displays a convex shape, while the edge controlled by C_{01} takes on a wavy shape. Similarly, the weights of control points C_{21} and C_{12} are the same, but their positions are different. The position of C_{21} is higher, therefore it controls a convex edge; the position of C_{12} is lower, hence it controls a concave edge.

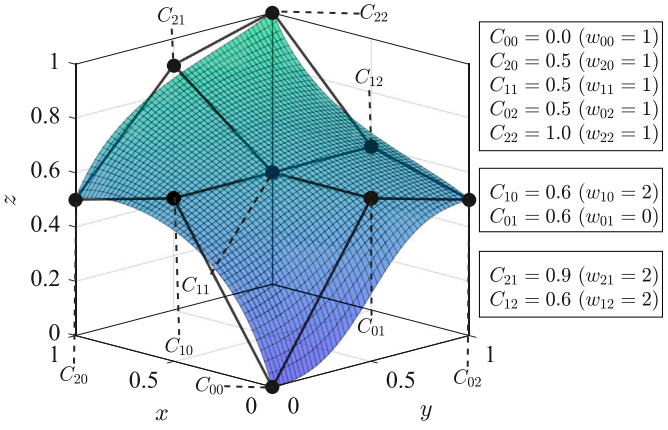


Fig. 3. A qRBSR with control points and their weights.

Algorithm B in Appendix B demonstrates $\mathbf{Z}_{RBSR} \in \mathbb{R}^{k \times m}$ with the first input vector $\mathbf{X} \in \mathbb{R}^k$, the second input vector $\mathbf{Y} \in \mathbb{R}^m$, and parameter vector $\mathbf{P} \in \mathbb{R}^{2n^2+4n+1}$, where $k, m, n \in \mathbb{N}$. Algorithm $\mathfrak{B}$ is from Eq. (1) and $\mathfrak{Q}$ is from Eq. (6).

2.2.3. General initial setups of RBR

In this paper, the RBR is limited to the range of 0 to 1. To realize this range and increase the flexibility of fitting, the general initial lower bound $\mathbf{L}_C$ and upper bound $\mathbf{U}_C$ of RBCR are defined as:

$$\mathbf{L}_{C\alpha} = 0, \mathbf{U}_{C\alpha} = \begin{cases} 1, & \alpha = 2\gamma - 1 \\ 2, & \alpha = 2\gamma \\ 1, & \alpha = 2n - 1 \\ 1, & \alpha = 2n \end{cases} \tag{8}$$

$$\forall \alpha \in \{1, 2, \ldots, (2n)\}, \ \gamma \in \{1, 2, \ldots, (n-1)\}$$

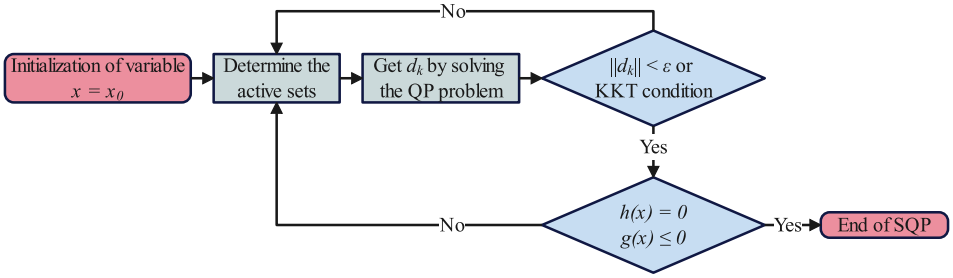


Fig. 4. The process of the SQP algorithm.

Similarly, the general initial lower bound $\mathbf{L}_S$ and upper bound $\mathbf{U}_S$ of RBSR are defined as:

$$\mathbf{L}_{S\alpha} = 0, \mathbf{U}_{S\alpha} = \begin{cases} 1, & \alpha = 2\gamma - 1 \\ 2, & \alpha = 2\gamma \\ 1, & \alpha = 2n^2 + 4n - 1 \\ 1, & \alpha = 2n^2 + 4n \\ 360°, & \alpha = 2n^2 + 4n + 1 \end{cases} \tag{9}$$

$$\forall \alpha \in \{1, 2, \ldots, (2n^2 + 4n + 1)\}, \gamma \in \{1, 2, \ldots, (n^2 + 2n - 1)\}$$

The initial neutral value is defined as the mean value of the lower and upper bounds.

2.3. Sequential quadratic programming

Sequential Quadratic Programming (SQP) is an efficient numerical optimization solution for the Non-Linearly Constrained Problem (NLCP). The NLCP is defined as [18]:

$$\begin{aligned} &\min\ f(\mathbf{x}) \\ &s.t.\ h_u(\mathbf{x}) = 0\ (u = 1, 2, \ldots, m) \\ &\quad g_v(\mathbf{x}) \leq 0\ (v = 1, 2, \ldots, p) \end{aligned} \tag{10}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$, $h : \mathbb{R}^n \mapsto \mathbb{R}^m$, $g : \mathbb{R}^n \mapsto \mathbb{R}^p$. To solve such NLCP, which means to find the optimal point x^*, the Lagrange-Newton method is often used [19]. First of all, the Lagrangian function is constructed, which can convert the constrained problem to the unconstrained problem. The Lagrangian function is defined as [20]:

$$\mathcal{L}(\mathbf{x}, \lambda, \mu) = f(\mathbf{x}) + \lambda^T \mathbf{h}(\mathbf{x}) + \mu^T \mathbf{g}^*(\mathbf{x}) = f(\mathbf{x}) + \sum_{u=1}^{m} \lambda_u h_u(\mathbf{x}) + \sum_{v=1}^{p} \mu_v g_v^*(\mathbf{x}) \tag{11}$$

In Eq. (11), $\mathbf{g}^*(\mathbf{x})$ is the active sets meeting the conditions [21]:

$$g_v(x_k) \geq 0,\ g_v^*(x_{k+1}) = 0,\ k \in \mathbb{N} \tag{12}$$

The Newton method can be used to iteratively optimize the objective function. Each iterative step d_k with the Hessian matrix $H_f(x)$ of $f(x)$ is defined as [21]:

$$d_k = x_{k+1} - x_k = -[H_f(x_k)]^{-1} \nabla f(x_k),\ k \in \mathbb{N} \tag{13}$$

For the SQP algorithm, the NLCP is defined as:

$$\begin{aligned} &\min\ f(x_k) + \nabla f(x_k)^T d_k + \frac{1}{2} d_k^T H_f(x_k) d_k \\ &s.t.\ h(x_k) + \nabla h(x_k)^T d_k = 0 \\ &\quad g^*(x_k) + \nabla g^*(x_k)^T d_k \leq 0 \end{aligned} \tag{14}$$

To solve the NLCP of SQP in Eq. (14), its Lagrangian function can be constructed from Eq. (11). Using the Karush-Kuhn-Tucker (KKT) conditions [22], the NLCP with the constructed Lagrangian function of SQP can be transformed into a new unconstrained problem, which can be simplified as:

$$\begin{bmatrix} H_f(x_k) & \nabla \mathbf{h}(\mathbf{x}) & \nabla \mathbf{g}^*(\mathbf{x}) \\ \nabla \mathbf{h}(\mathbf{x})^T & O & O \\ \nabla \mathbf{g}^*(\mathbf{x})^T & O & O \end{bmatrix} \begin{bmatrix} d_k \\ \lambda \\ \mu \end{bmatrix} = \begin{bmatrix} -\nabla f(x_k) \\ -\mathbf{h}(x_k) \\ -\mathbf{g}^*(x_k) \end{bmatrix} \tag{15}$$

Fig. 4 demonstrates the process of the SQP algorithm [23]. The active sets $\mathbf{g}^*(x_k)$ change with each iteration. The direction for the next iteration is determined by solving the constrained Quadratic Programming (QP) problem based on the current active sets as shown in Eq. (15). And the next iteration point x_{k+1} is equal $x_k + d_k$. Iteration continues until $\nabla f(\mathbf{x}) = 0$ or the norm $||x_{k+1} - x_k||$ falls below a small positive constant ϵ.

3. Data-driven RBR parameter estimation with SQP

The upper and lower bounds of RBR have been defined in Section 2.2. As mentioned before, the neutral value is the mean value of the upper and lower bounds. However, if each RBR in the control system uses this neutral value as an initial value for overall optimization, it usually results in overfitting. Section 5.2 demonstrates the overfitting phenomenon using the initial neutral value. Moreover, for complex control systems, due to the relatively large range of the initial value, the optimization convergence process of the entire system can be slow. Therefore, a method is needed to reduce the upper and lower bounds to a suitable range, so as to improve the optimization efficiency and accuracy.

Fig. 5 demonstrates an approach using the relational database introduced in Section 2.1 to estimate the RBR parameters and redefine the lower and upper bounds for the selected LUT. Since many vehicles in the database have similar engine controls and are subject to the same physical principles, there are a finite number of unique entries for each LUT. Therefore, this approach first classifies each class of the selected LUT in the database using SQL queries. Then it estimates the RBR fitting parameters of the selected LUT's every class using the SQP algorithm, whose constrained conditions are the initial lower and upper bounds. For the SQP algorithm, a quadratic derivable cost function is required for optimization, which can be defined as [24]:

$$\mathcal{J}(\mathbf{p}) = \|\hat{\mathcal{Z}} - \mathcal{Z}_{RBR}(\mathbf{p})\|_2^2 \tag{16}$$

where $\hat{\mathcal{Z}} \in \mathbb{R}^{N \times M}$ is a matrix composed of the selected LUT's data points, $\mathcal{Z}_{RBR} \in \mathbb{R}^{N \times M}$ is a matrix composed of the RBR output values for the corresponding inputs, and $\mathbf{p}$ is the parameter vector that controls RBR. Thus, the optimal parameter vector $\mathbf{p}^*$ is:

$$\mathbf{p}^* = \underset{\mathbf{p}}{\operatorname{argmin}}\ \mathcal{J}(\mathbf{p}) \tag{17}$$

Then, the RBR parameters of the selected LUT's N classes are estimated using this method. Subsequently, the maximum and minimum values of each element from these parameter vectors are filtered to form new upper and lower bounds, resulting in the new neutral value.

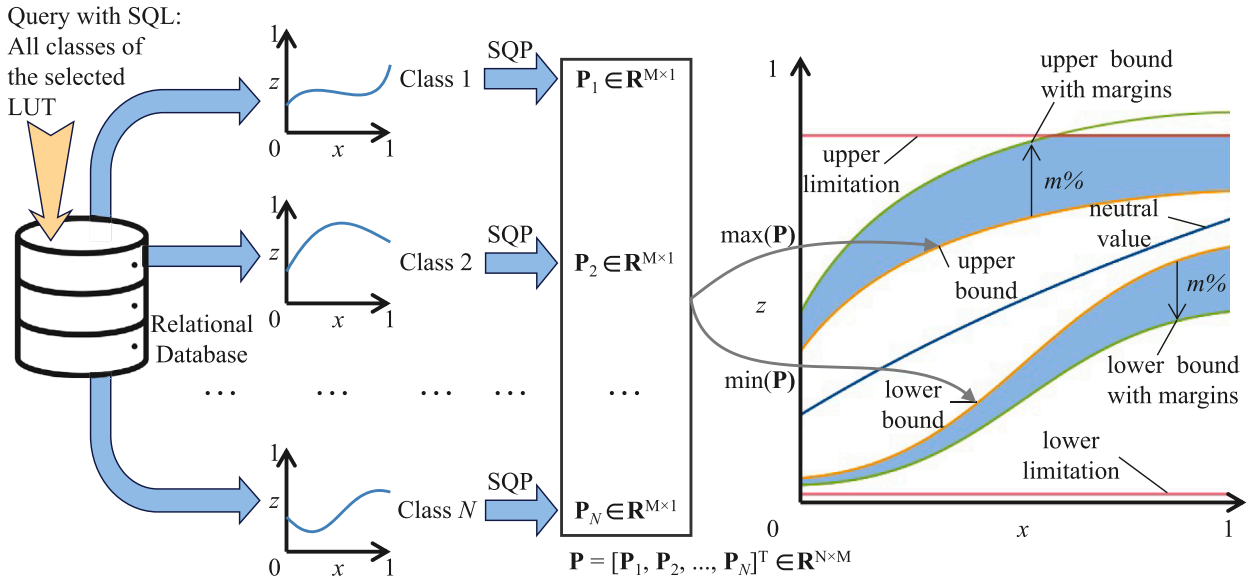


Fig. 5. Process of the data-driven RBR parameter estimation with SQP algorithm.

Adding appropriate margins to the current upper and lower bounds can improve the accuracy of optimization, as demonstrated by the results in Section 5.3. The margins $m\%$ in Fig. 5 can be considered a hyperparameter. If the margins are too small, the optimization performance may be suboptimal; if too large, overfitting may occur. Therefore, manual adjustment of the margins is necessary to achieve the best optimization effect. In addition, it is crucial to read the upper and lower limitations of the selected LUT from the database, ensuring that the bounds with margins fall within the permissible range of the LUT. The blue shaded area in Fig. 5 represents the final effective margins area.

4. Experimental setups and core processes of pre-calibration

In order to validate the methodologies of pre-calibration, a LUT-based MISO engine control system is needed. In addition, a vehicle measurement is necessary for the pre-calibration.

4.1. LUT-based MISO engine control system

The oil temperature control system in Fig. 6, which is a typical LUT-based MISO engine control system, serves as an example to demonstrate the results.

This system, characterized by diverse ECU calibration data, is well-suited to demonstrate the effectiveness of the methodologies. Furthermore, the model output of the oil temperature model is essential for component protection. Excessively high model temperatures would lead to an enrichment of the air-fuel ratio. This would lead to compromised fuel consumption and emissions and must therefore be prevented by precisely tuning the model. On the other hand, excessively low model deviations might lead to increasing engine wear or catastrophic failure. The oil temperature model is a sub-control system of the ECU used to calculate the oil temperature (T_{Oil} in °C) in the oil sump during the engine's warm state. The system first determines whether the vehicle is at the lowest engine speed or in a coasting state. If either condition is met,

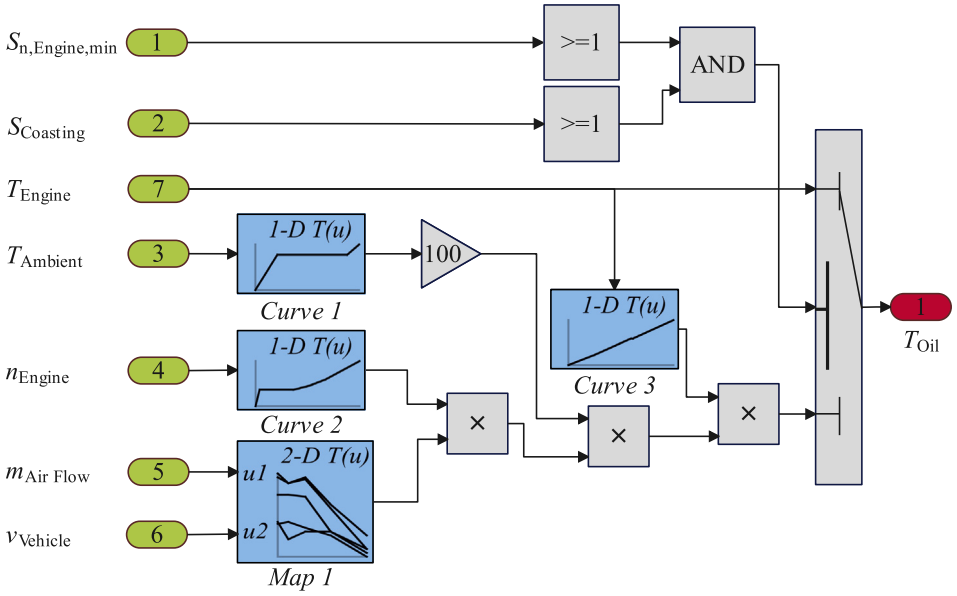


Fig. 6. Control system model of the oil temperature in the oil sump at the warm state of the engines.

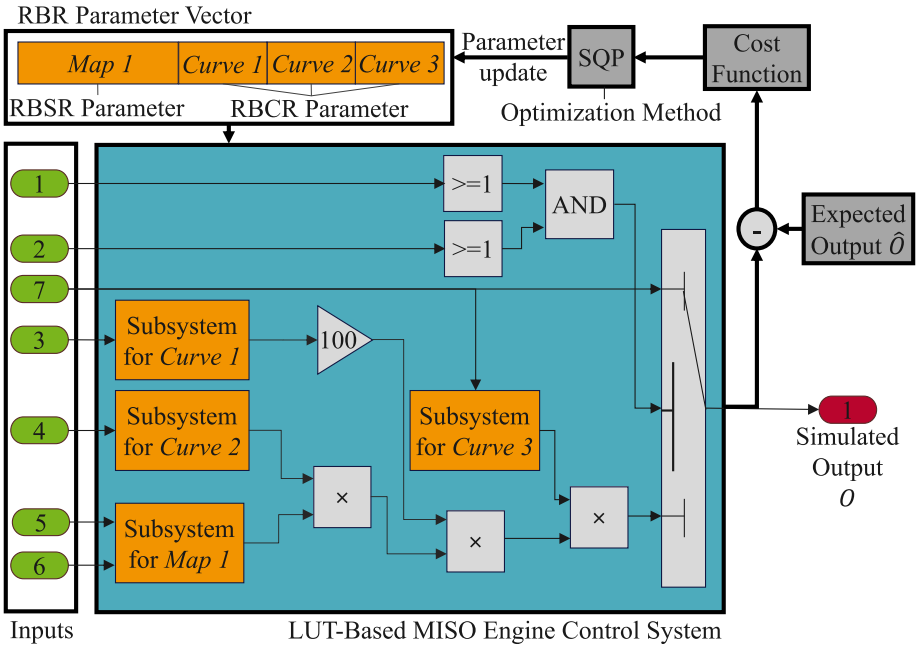


Fig. 7. Process of the MiL optimization with the control system model.

T_{Oil} is set to the engine temperature T_{Engine}; otherwise, T_{Oil} is calculated using a series of LUTs and constants. It has 7 inputs, which are:

- $S_{\text{n,Engine,min}}$: Engine idle state (possible values: 0 or 1)
- S_{Coasting}: Vehicle coasting state (possible values: 0 or 1)
- T_{Ambient}: Ambient temperature (unit: °C)
- n_{Engine}: Engine speed (unit: min^{-1})
- $m_{\text{Air Flow}}$: Air mass flow (unit: kg/h)
- v_{Vehicle}: Vehicle speed (unit: km/h)
- T_{Engine}: Engine temperature (unit: °C)

This control system contains three *Curves* and one *Map* that need to be calibrated:

- *Curve 1*: Influence factor of ambient air temperature $f_{\text{T,Ambient}}$
- *Curve 2*: Influence factor of engine speed $f_{\text{n,Engine}}$
- *Curve 3*: Influence factor of engine temperature $f_{\text{T,Engine}}$
- *Map 1*: Influence factor of load and vehicle speed $f_{\text{m,Air Flow/v,Vehicle}}$

4.2. Pre-calibration using model-in-the-loop optimization

Fig. 7 illustrates the core process of the Model-in-the-Loop (MiL) optimization with the white-box control system model from Fig. 6 [25]. This MiL optimization process relies on the control system's white-box model. However, the original LUT data points will first be parameterized using RBR. The RBR implementation, including both RBCR and RBSR subfunctions, was carried out in MATLAB [26].

As mentioned in Section 2.2.3, the RBR value range employed herein is $[0, 1]$, which may not necessarily align with the value range required by LUTs. Similarly, the definition domain of each independent variable of RBR must be $[0, 1]$, which might not correspond to the actual input range of LUTs. Ultimately, the control system does not require the RBR models, but rather the discrete points derived from the RBR.

Therefore, a subsystem is required to process the data pre- and post-RBR modeling, as shown in Fig. 8. The initial step involves normalizing all inputs to $[0, 1]$. The input $\mathcal{X}$ can be normalized using min-max normalization as:

$$\mathcal{X}^* = \frac{\mathcal{X} - \min(\mathcal{X})_{global}}{\max(\mathcal{X})_{global} - \min(\mathcal{X})_{global}} \tag{18}$$

where $\max(\mathcal{X})_{global}$ and $\min(\mathcal{X})_{global}$ are the global maximum and minimum of the input $\mathcal{X}$ quried from databases with SQL. The input 1 and 2 in Fig. 8 are normalized with this method. In comparison, if RBCR instead of RBSR is applied, then only input 1 is used. Next, these normalized inputs are modeled by RBR with the RBR parameter vector for *Maps/Curves*. After that, the scaling process for the RBR output $\mathcal{Z}$ can be seen as the denormalization, which is defined as:

$$\mathcal{Z}^* = \mathcal{Z} \cdot (\max(\mathcal{Z})_{global} - \min(\mathcal{Z})_{global}) + \min(\mathcal{Z})_{global} \tag{19}$$

where $\max(\mathcal{Z})_{global}$ and $\min(\mathcal{Z})_{global}$ are the global maximum and minimum of the RBR output $\mathcal{Z}$ from databases. Since the inputs of the subsystem itself are discrete, the RBR output is discrete as well.

By applying the above subsystem process to all subsystems in this example, the control system in Fig. 7 can be controlled by one RBSR parameter vector, and three RBCR parameter vectors. Concatenating these parameter vectors into one, it can be regarded as the final optimization target. In the same way, the upper and lower bounds corresponding to each sub-parameter vector can also be summarized into one upper-bound vector and one lower-bound vector, to obtain the constraints of the NLCP.

The cost function defined in Eq. (16) can also be applied as the target optimization function for the MiL optimization problem of the engine control system [24]. Thus, the parameter vector can be updated and optimized with the optimization method like SQP by changing $\mathcal{Z}$ in Eq. (16) to the current system output O and $\hat{\mathcal{Z}}$ to the expected system output $\hat{O}$. The optimization can be considered complete when the error between the actual system output and the expected system output converges to a suitable range or the optimization cannot be continued.

4.3. Vehicle measurement

As mentioned in Section 4.2, the defined expected system output is the prerequisite for the MiL optimization. Thus, the engine control

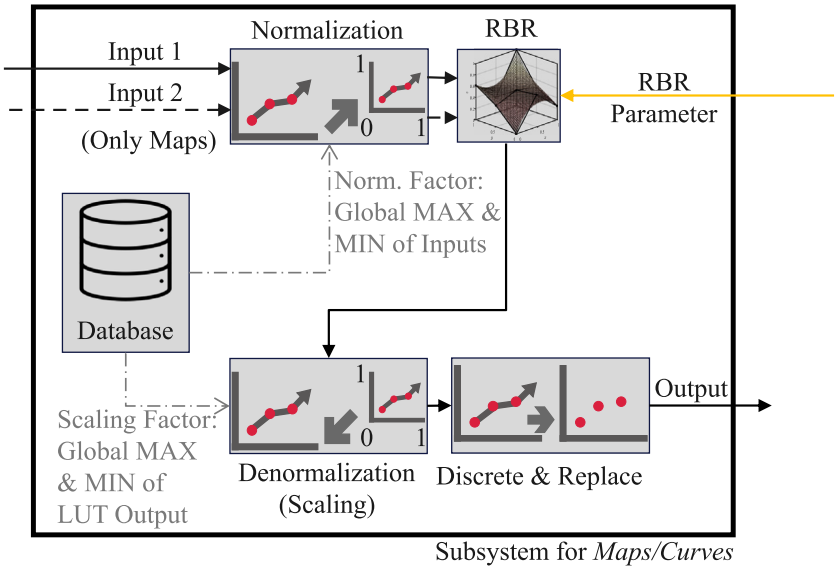


Fig. 8. Subsystem for *Maps/Curves*.

system outputs from a time series vehicle measurement are selected as the expected system outputs for the pre-calibration. For this purpose, the test vehicle was equipped with an application ECU. The measurement computer was connected to the application ECU via an ES582 - USB CAN FD interface from ETAS. The ETAS INCA software was utilized on the measurement computer to establish a connection to the calibration ECU via the CAN. By loading the program- and dataset files (.a2l and .hex) into ETAS INCA the ECU values can be interpreted and selected. This setup enabled real-time recording of the system output from the oil temperature model. The measurement grid for the ECU values was configured with a 100 ms time-synchronized interval to ensure high-resolution data capture. To monitor the actual oil temperature of the engine oil pan, an independent substitute temperature value is modeled on the ECU in addition to the real sensor signal. The permissible deviations for this substitute value are manufacturer-specific but are generally required to remain below 20 °C for production applications. This ensures reliable prevention of engine oil molecule cracking and potential engine damage in the event of an oil sensor failure. Fig. 9 shows the engine control system outputs from an exemplary Measurement A [26]. In order to reproduce the same driving scenario in the MiL, the engine control system inputs from Measurement A serve as the engine control system inputs for pre-calibration. Table 1 displays the technical specifications of the vehicle for measurement.

Given the sensitivity of the cost function in Eq. (16) to extreme values [27], it is crucial to eliminate values with excessive deviation from the mean value. In order to reduce the risk of overfitting and enhance the accuracy and robustness of the model, Six Sigma (6σ) processing is applied to the time series of the expected model outputs. This process involves selecting data points within three standard deviations σ from the mean μ of the measurement [28]. The measurement used in subsequent pre-calibration will directly employ Measurement A processed by 6σ.

4.4. Evaluation metrics

To quantitatively assess the accuracy of the pre-calibrated LUTs and the optimized system outputs, multiple evaluation metrics are employed. The Pearson Correlation Coefficient (PCC), Root Mean Square Error (RMSE), and Mean Absolute Error (MAE) are used to evaluate how well the pre-calibrated LUTs match the target LUTs. Meanwhile, the RMSE, MAE, and Coefficient of Determination (R^2) are used to assess the accuracy of the optimized system outputs, verifying how closely they align with expected system behavior [29].

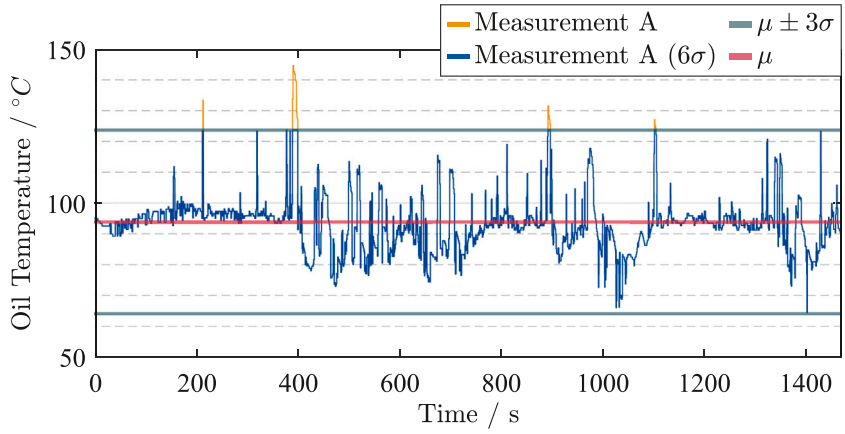


Fig. 9. Original and 6σ processed Measurement A.

Table 1
Technical specifications of the test vehicle.

Technical specifications			
Engine type:	Turbocharged four-stroke inline-six gasoline engine	Engine displacement (cc):	2998
Transmission:	Automatic (8AT)	Max. engine output (kW):	210
Powertrain:	Hybrid (PHEV)	Max. engine torque (Nm):	450
Drivetrain:	Rear-Wheel Drive (RWD)	Max. electric motor power (kW):	83
Cooling circuit	Integrated water to oil cooler	Max. electric motor torque (Nm):	265

In the following formulas, y_i and $\hat{y}_i$ denote the target values and actual values, respectively. For PCC, these correspond to the target and pre-calibrated LUT values. Additionally, $\bar{y}$ and $\bar{\hat{y}}$ represent the means of the target and actual values. N is the number of data points.

The Pearson Correlation Coefficient (PCC) evaluates the shape similarity between pre-calibrated LUTs and target LUTs [30]. It measures the linear relationship between the two sets of values and is computed as:

$$PCC = \frac{\sum_{i=1}^{N}(y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}\sqrt{\sum_{i=1}^{N}(\hat{y}_i - \bar{\hat{y}})^2}} \tag{20}$$

The absolute PCC value ranges from 0 to 1, where a value close to 1 signifies a high correlation between the datasets and a value near 0 implies a low correlation. A large PCC value between two curves or surfaces indicates that they have high shape similarity.

The Root Mean Square Error (RMSE) quantifies the magnitude of deviations between pre-calibrated and target LUT values, or between optimized and expected system outputs. It provides a measure of overall accuracy by considering the squared differences between target and actual values. RMSE is defined as:

$$RMSE = \sqrt{\frac{1}{N}\sum(y_i - \hat{y}_i)^2} \tag{21}$$

A lower RMSE indicates that the actual values are closer to the targets, reflecting better performance of the pre-calibration or optimization process.

The Mean Absolute Error (MAE) provides an alternative error measure that assigns equal weight to all deviations. It is defined as:

$$MAE = \frac{1}{N}\sum|y_i - \hat{y}_i| \tag{22}$$

A lower MAE indicates higher accuracy, meaning the pre-calibrated LUTs follow the target LUTs closely and the optimized system outputs align well with expected values.

The Coefficient of Determination (R^2) assesses how well the optimized system outputs fit the expected outputs by comparing the total variance explained by the model:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2} \tag{23}$$

A value close to 1 indicates that the optimized system accurately replicates the expected behavior, validating the effectiveness of the optimization process.

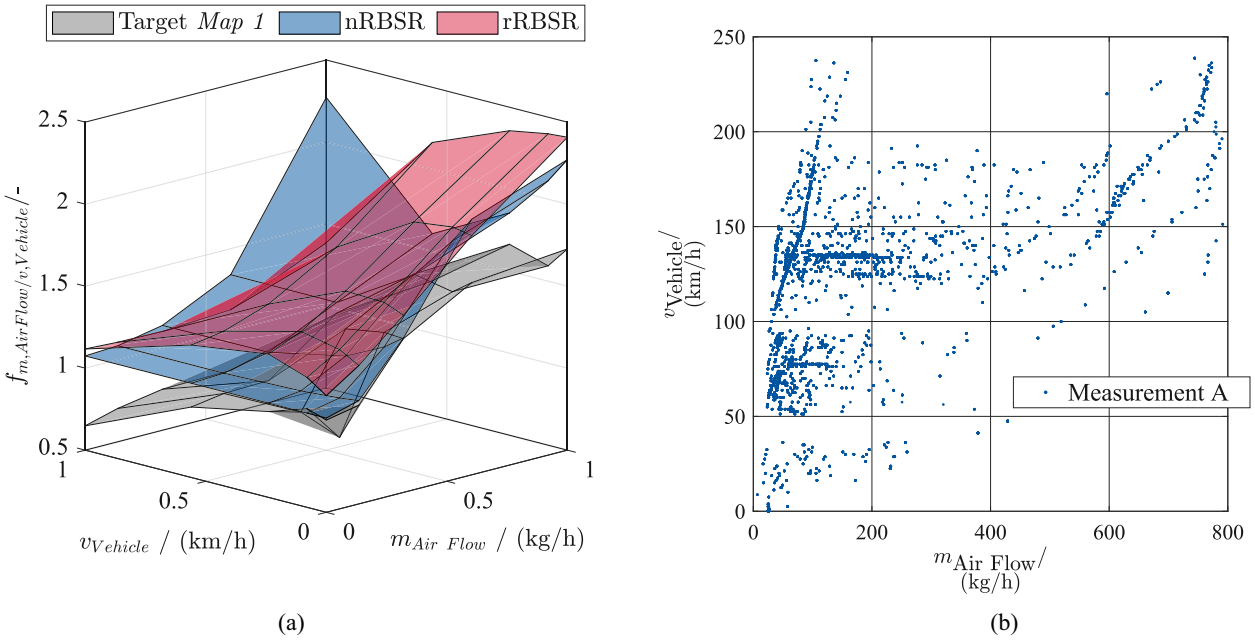
5. Results and analysis

Next, the results of the pre-calibration will be evaluated to demonstrate how various setups influence the outcomes of the pre-calibration process. The oil temperature control system described in Section 4.1 will be applied for validation.

5.1. Comparison of rotational and normal RBSR

As described in Section 2.2.2, to enhance the fitting capacity of RBSR for Maps, a coordinate rotation for transformation is applied before RBSR. Consequently, the forthcoming results will compare the outcomes of normal RBSR (nRBSR) without a coordinate rotation, and rotational RBSR (rRBSR) with a coordinate rotation.

The comparison focuses on the fitting results for the target *Map 1* of the oil temperature control system applied to Measurement A [26]. Fig. 10(a) presents the results of the comparison between rRBSR and



(a) (b)

Fig. 10. Results for the rotational or normal RBSR. (a) Comparison between the rRBSR or nRBSR and the target *Map 1*. (b) Distribution of $m_{\text{Air Flow}}$ and v_{Vehicle} in Measurement A.

Table 2
Metrics table of differences between *Maps* fitted by rRBSR or nRBSR and the target *Map 1*.

Model	PCC	RMSE/–	MAE/–
rRBSR	0.9098	0.1952	0.4668
nRBSR	0.7516	0.4388	1.0703

Table 3
Metrics for the performance of white-box model's simulated outputs and the final optimized system outputs using the general RBR initial setups.

Model	RMSE/°C	MAE/°C	R^2/–
White-Box Model	1.1	12.1	0.9965
General Setups	4.6	24.8	0.6941

nRBSR in fitting the target *Map 1*. Both rRBSR and nRBSR are trained using Measurement A defined in Fig. 9 and are based on qRBSR, whose parameters are fitted by SQP.

As shown in Fig. 10(a) and Table 2, after incorporating additional rotational degrees of freedom, rRBSR significantly outperforms nRBSR in pre-calibrating target *Map 1*. The PCC of rRBSR's pre-calibrated *Map* relative to target *Map 1* reaches 0.9098, markedly higher than nRBSR's 0.7516, which indicates higher shape similarity. Additionally, both rRBSR's RMSE and MAE are lower as well.

However, its fitting in regions with higher air mass flow $m_{\text{Air Flow}}$ and lower vehicle speed v_{Vehicle} is not strong. As the scatter plot of the sample points from Fig. 10(b) indicates, the measurement data in these specific regions of Measurement A used for pre-calibration are sparse, aligning with physical principles where typically high air mass flow is not associated with low vehicle speed. This may lead to discrepancies in pre-calibration in these areas due to the lack of data points. Nonetheless, rRBSR demonstrates superior pre-calibration capabilities compared to nRBSR, underscoring the necessity of applying a rotational transformation to the inputs of the *Map* before using RBSR.

5.2. The overfitting phenomenon using general RBR initial setups

Next, the results of pre-calibration using the general RBR initial setups introduced in Section 2.2.3 will be presented. Fig. 11 demonstrates the comparative performance between the simulated outputs of the white-box model and the final optimized system outputs using the general RBR initial setups, labeled as "General Setups" in the figure, on the original Measurement A.

It can be seen from Fig. 11 and Table 3 that the outputs from the "General Setups" are relatively close to the original system outputs. However, its pre-calibrated LUTs do not match well with the shapes from the calibration that was used to record Measurement A on the vehicle.

As shown in Fig. 12, when visualizing its pre-calibrated LUTs, *Curve 1* and *Curve 2* lack distinct shapes. This may be attributed to the optimization process focusing excessively on optimizing the system outputs, leading to a loss of LUT shapes and resulting in overfitting. The fundamental cause of overfitting might be the overly broad scope of the general RBR initial setups. Therefore, there is a need for new approaches to scale down the RBR initial setups to appropriate intervals, such as the approach mentioned in Section 3.

5.3. Performance of optimization procedures

As shown in Table 4, various Optimization Procedures (OPs) are compared together [26]. First of all, SQP is compared with the other two optimization methods, namely the Inner Point (IP) method and the Particle Swarm Optimization (PSO) method [31,32]. The PSO method underperforms compared to SQP and is the most time-consuming of the three optimization methods. For the IP method, it outperforms both the PSO and SQP algorithms at the quadratic RBR and 0 % margins (*OP 3* compared to *OP 1, OP 2*). However, when applied to cubic RBR and 75 % margins, the R^2 for IP is significantly lower compared to SQP and PSO (*OP 9* compared to *OP 7, OP 8*). This indicates a lack of stability in the IP method for this case. Thus, while the SQP may not excel in all scenarios, it offers a more stable and efficient performance compared to the performance of IP and PSO.

An increase in the RBR degree or the margins m% enhances the final optimized system outputs. As shown in Fig. 13, at margins of 100 % (*OP 10*), the overfitting phenomenon appears again in *Curve 2*, despite *OP 10* performs better than *OP 7* on final optimized system outputs. As such, this study identifies a margin of 75 % as the optimal value for the hyperparameter m%. Then, the best results are achieved in *OP 12* with 6th RBR degree.

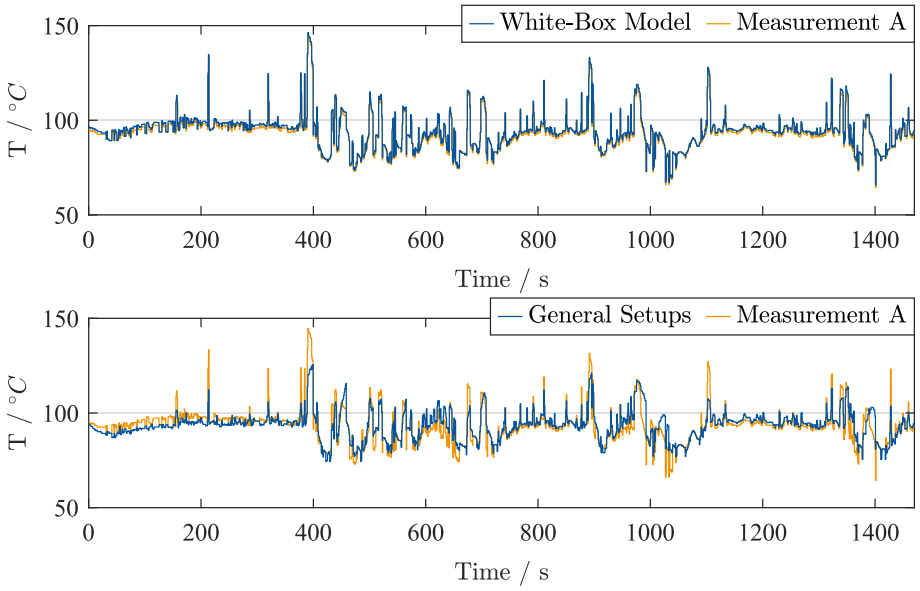


Fig. 11. Final optimized system outputs with the general RBR initial setups.

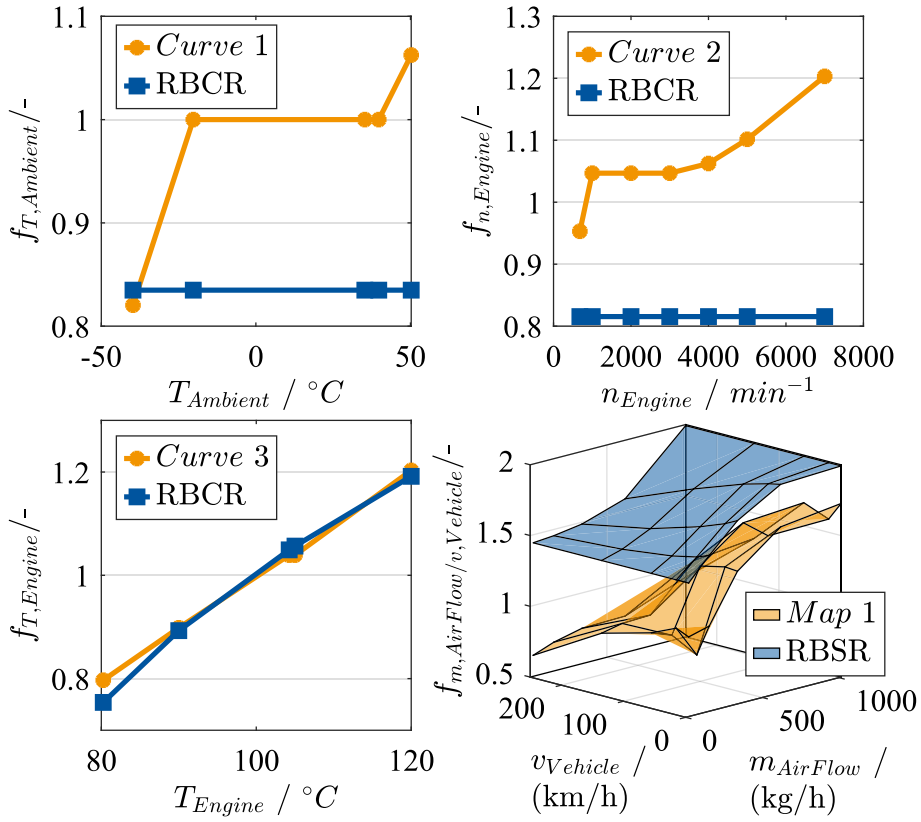


Fig. 12. Final pre-calibrated LUTs using the general RBR initial setups.

Table 4
The performance of optimization procedures (OPs).

OP	RBR degree	Optimization	Margins	Time/s	RMSE/°C	R^2/–
OP 1	2	SQP	0 %	5630	5.5	0.5872
OP 2	2	PSO	0 %	84,767	5.5	0.4899
OP 3	2	IP	0 %	6278	3.8	0.7964
OP 4	2	SQP	25 %	**3144**	4.8	0.4579
OP 5	2	SQP	50 %	7454	4.5	0.5504
OP 6	2	SQP	75 %	6470	3.4	0.8493
OP 7	3	SQP	75 %	4589	2.6	0.8987
OP 8	3	PSO	75 %	55,580	4.6	0.7881
OP 9	3	IP	75 %	7471	4.3	0.3606
OP 10	3	SQP	100 %	9282	2.4	0.9006
OP 11	4	SQP	75 %	7826	2.5	0.9119
OP 12	6	SQP	75 %	29,456	**1.9**	**0.9385**

Although achieving the best *OP* requires experimenting with various setups of hyperparameters, which can be time-consuming – for example, attempts from *OP 1* to *OP 12* in this paper take about 60 h – this method is still more efficient than traditional manual calibration. This is because the computer program will complete the pre-calibration tasks for each *OP* with a single setup of the hyperparameters. In contrast, manual calibration requires the continuous involvement of engineers. Moreover, in calibration tasks, acquiring vehicle measurements is more time-consuming than the calibration itself. The pre-calibration approach in this paper can complete the pre-calibration with just one vehicle measurement, thus helping to save overall time and enhance efficiency in the calibration process.

5.4. Evaluation of OP 12

Fig. 14 and Table 5 display the results of the final optimized system outputs for pre-calibration using *OP 12* from Table 4.

As shown in Fig. 14 and Table 5, the final optimized system outputs from the pre-calibration using initial setups generated by *OP 12* closely align with the original system outputs. Their accuracy nearly matches that of the simulated system outputs from the white-box model using target LUTs, as shown in Fig. 11.

An inspection of Fig. 15 reveals that the four LUTs pre-calibrated by *OP 12* bear a strong resemblance to the target LUTs. The high similarity of shapes can be further validated by the high PCC values of the four LUTs listed in Table 6, all exceeding 0.8 and approaching 1.

However, the pre-calibrated *Curve 2* no longer shows a broken line with a steep slope in the engine speed n_{Engine} range of 680 to 1000 min^{-1} due to measurement constraints. Since the test vehicle's n_{Engine} cannot drop below 1000 min^{-1}, this data gap prevents the final optimization from fully replicating the target *Curve 2* and affects the optimization of other LUTs.

Considering the high similarity of shapes between the pre-calibrated LUTs and the targets, it can be concluded that pre-calibration using

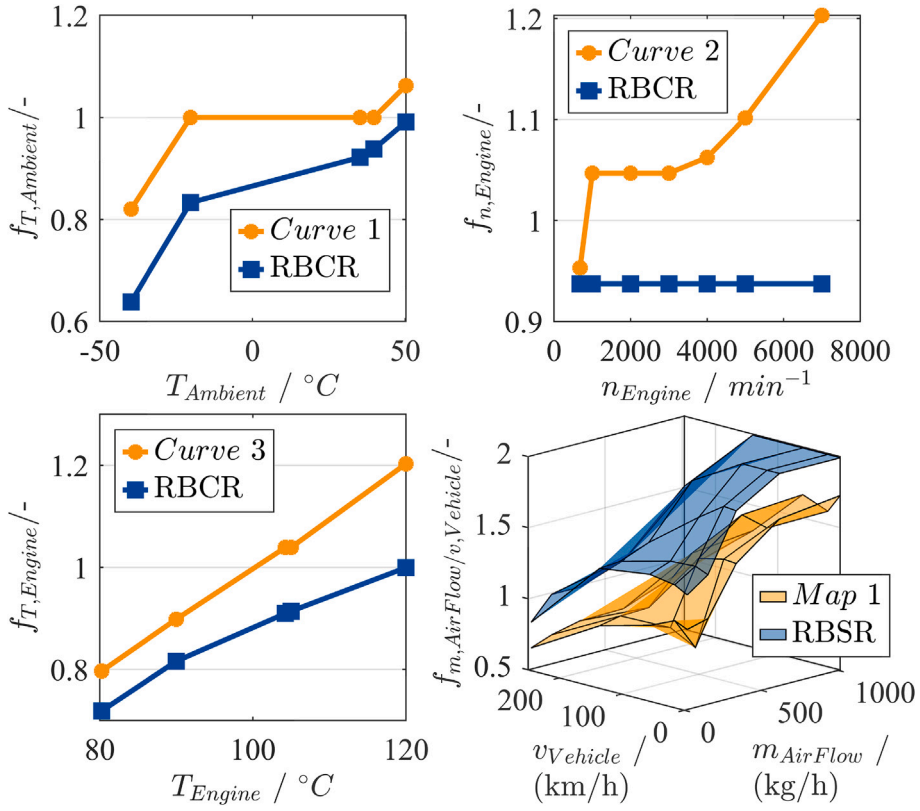


Fig. 13. Final pre-calibrated LUTs of *OP 10*.

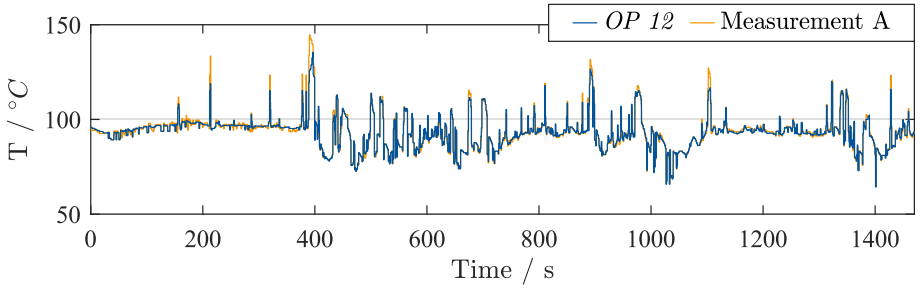


Fig. 14. Final optimized system outputs with *OP 12*.

Table 5
Metrics for the performance of white-box model's simulated outputs and the final optimized system outputs using *OP 12*.

Model	RMSE/°C	MAE/°C	R^2/–
OP 12	1.9	18.8	0.9385

OP 12 achieves near-perfect initial shapes for LUTs, which fulfill the pre-calibration target.

5.5. Applicability and limitations of the pre-calibration approach

This study focuses on an oil temperature model, it demonstrates an approach that can be adapted to other temperature-related models (e.g. intake, exhaust, or turbine temperatures) or to physical control models that exhibit similar behavior across vehicles and rely on regression for calibration, such as air or fuel path systems. The optimization method is versatile, requiring only a measurable or modellable target specification, which can originate from sensors, ECU-modeled values, or co-simulations.

The method is particularly valuable during early development stages, where physical measurements are limited, enabling pre-calibration using simulations or preliminary models. This accelerates development timelines while maintaining accuracy. Unlike traditional methods, it evaluates the calibration process against target specifications, whether derived from real data or simulations.

While the proposed calibration approach leveraging Sequential Quadratic Programming (SQP) has demonstrated effectiveness in the context of engine control systems, we acknowledge that its performance may vary depending on the specific problem and the underlying nonlinearities. The stability and efficiency of SQP are often problem-dependent, particularly in cases where the nonlinearities exhibit behaviors different from those typically encountered in engine control subsystems. The oil temperature model used in this work is primarily governed by physical laws and represents a linearized version of the practical implementation, which facilitates the applicability of SQP in this context. Furthermore, leveraging historical data from the database

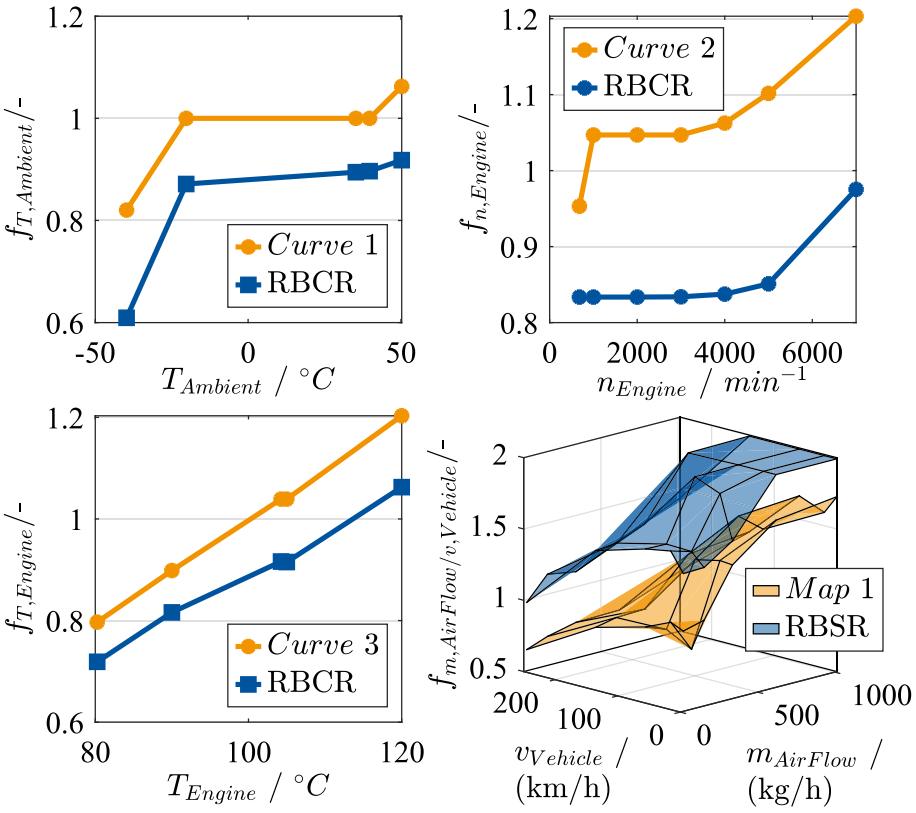


Fig. 15. Final pre-calibrated LUTs of *OP 12*.

Table 6
Metrics of pre-calibrated LUTs with *OP 12*.

LUT	PCC	RMSE/–	MAE/–
Curve 1	0.9778	0.1959	0.4531
Curve 2	0.8456	0.3561	0.3695
Curve 3	0.9983	0.2216	0.4844
Map 1	0.9168	0.2322	0.7114

to set boundary conditions and margins contributes to the stabilization of the SQP approach. However, for problems involving highly complex nonlinearities – such as those arising in co-simulation models of physical surrogate systems or in tasks lacking a well-founded database – SQP may require modifications to ensure robustness.

Furthermore, this methodology is suitable for calibrating engine control systems that are identical or just slightly modified in comparison to the LUT software revisions from the database. The verification must be ensured before using this method, otherwise the result could be strongly biased. For example, if a four-cylinder engine and a six-cylinder engine share the same control system and the four-cylinder engine has already been calibrated, its historical calibration data can be used to guide the six-cylinder engine's calibration, utilizing the pre-calibration methodology in this paper.

However, significant structural or functional changes to the engine control system, such as a change in ECU software supplier, may render this methodology inapplicable, often restricting its use to engine families with the same or similar suppliers.

6. Conclusions and future perspectives

In this paper, historical calibration data, supported by relational database technology, effectively optimizes the constraints of the NLCP problem during pre-calibration. This data-driven approach reduces the need for calibration engineers to have an in-depth understanding of the engine control system's mechanisms. It scales the LUT values to appropriate ranges. At the same time, the RBR parameterization of LUTs proves effective. It not only provides smooth modeling using a unified set of parameters but also enables parametric optimization with fewer optimization points than the original discrete LUT data points.

By correctly setting hyperparameters, such as margins, optimization methods, and RBR degrees, overfitting during pre-calibration is successfully mitigated. In the MiL optimization process using a white-box surrogate model of the MISO oil temperature control system, the key prerequisite is that the simulated system outputs closely match the expected system outputs (i.e. those from vehicle measurements). The R^2 value between the simulated and expected system outputs reaches 0.9385, with a low RMSE of 1.9 °C, indicating that the white-box engine control system with the pre-calibrated LUTs meets the prerequisite of pre-calibration. Visualizations of the pre-calibrated LUTs show strong similarity to the target LUTs, with PCC values exceeding 0.8, fulfilling the pre-calibration target of generating near-perfect initial shapes for multiple LUTs in the engine control system. This paper's pre-calibration methodology enhances calibration efficiency and reduces reliance on hardware.

Additionally, the MiL-based pre-calibration framework presented in this paper is versatile and can be adapted for LUTs with no specific requirements. For cases where specific requirements exist-such as the need for strict integer LUT values or the inclusion of physical formulas-custom modifications can be made within the proposed framework.

There are areas for further improvement. For example, if the hyperparameters of the proposed pre-calibration are automatically and optimally set, the efficiency of the virtual calibration can be further improved. In addition, exploring additional optimization methods based on the complexity of the search space of the specific engine control system

may yield better results. In scenarios where a white-box model is unavailable, black-box surrogate models, such as deep-learning models, could serve as alternatives for system modeling.

CRediT authorship contribution statement

Matteo Meli: Writing – review & editing, supervision, methodology, and conceptualization. **Zezhou Wang:** Writing – original draft, validation, supervision, methodology, data curation, and conceptualization. **Stefan Sterlepper:** Writing – review & editing. **Mario Picerno:** Writing – review & editing. **Stefan Pischinger:** Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was supported by FEV Europe GmbH through a research grant to RWTH Aachen University.

Appendix A. Pseudocode for RBCR

Algorithm 1 RBCR.

Input: $\mathbf{X} \in \mathbb{R}^k, \mathbf{P} \in \mathbb{R}^{2n}$
Output: $\mathbf{Z} \in \mathbb{R}^k$
for $i = 1$ to k **do**
  $\mathbf{t} \leftarrow \mathbf{X}_i$
  $\mathbf{m} \leftarrow \mathfrak{B}_{n,n}(\mathbf{t}), \mathbf{n} \leftarrow \mathfrak{B}_{0,n}(\mathbf{t}) + \mathfrak{B}_{n,n}(\mathbf{t})$
  for $j = 1$ to $(n-1)$ **do**
    $\mathbf{m} \leftarrow \mathbf{m} + \mathbf{P}_{(2j-1)} * \mathbf{P}_{(2j)} * \mathfrak{B}_{j,n}(\mathbf{t}), \mathbf{n} \leftarrow \mathbf{n} + \mathbf{P}_{(2j)} * \mathfrak{B}_{j,n}(\mathbf{t})$
  end for
  $\mathbf{Z}_i \leftarrow \mathbf{P}_{(2n-1)} * (\mathbf{m}/\mathbf{n}) + \mathbf{P}_{(2n)}$
end for
return $\mathbf{Z}$

Appendix B. Pseudocode for RBSR

Algorithm 2 RBSR.

Input: $\mathbf{X} \in \mathbb{R}^k, \mathbf{Y} \in \mathbb{R}^m, \mathbf{P} \in \mathbb{R}^{2n^2+4n+1}$
Output: $\mathbf{Z} \in \mathbb{R}^{k \times m}$
$\epsilon \leftarrow \mathbf{P}_{(2n^2+4n+1)}$
$\begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \end{bmatrix} \leftarrow \mathfrak{Q}^{-1}(\epsilon) \begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \end{bmatrix}$ {Park transformation}
$\mathbf{A} \leftarrow [0, 1, \dots, n, 0, 1, \dots, n, \dots, 0, 1, \dots, n]^T$ {$[0, 1, \dots, n]^T$ repeats $(n+1)$ times}
$\mathbf{B} \leftarrow [0, 0, \dots, 0, 1, \dots, 1, \dots, n, \dots, n]^T$ {Each number repeats $(n+1)$ times}
$\mathbf{A} \leftarrow \mathbf{A}_{2:(n^2+2n)}, \mathbf{B} \leftarrow \mathbf{B}_{2:(n^2+2n)}$
for $i = 1$ to k **do**
  $\mathbf{u} \leftarrow \mathbf{X}_i$
  for $j = 1$ to m **do**
    $\mathbf{v} \leftarrow \mathbf{Y}_j$
    $\mathbf{m} \leftarrow \mathfrak{B}_{n,n}(\mathbf{u}) * \mathfrak{B}_{n,n}(\mathbf{v}), \mathbf{n} \leftarrow \mathfrak{B}_{0,n}(\mathbf{u}) * \mathfrak{B}_{0,n}(\mathbf{v}) + \mathfrak{B}_{n,n}(\mathbf{u}) * \mathfrak{B}_{n,n}(\mathbf{v})$
    for $t = 1$ to $(n^2 + 2n - 1)$ **do**
      $\mathbf{m} \leftarrow \mathbf{m} + \mathbf{P}_{(2t-1)} * \mathbf{P}_{(2t)} * \mathfrak{B}_{\mathbf{A}_t,n}(\mathbf{u}) * \mathfrak{B}_{\mathbf{B}_t,n}(\mathbf{v}), \mathbf{n} \leftarrow \mathbf{n} + \mathbf{P}_{(2t)} * \mathfrak{B}_{\mathbf{A}_t,n}(\mathbf{u}) * \mathfrak{B}_{\mathbf{B}_t,n}(\mathbf{v})$
    end for
    $\mathbf{Z}_{i,j} \leftarrow \mathbf{P}_{(2n^2+4n-1)} * (\mathbf{m}/\mathbf{n}) + \mathbf{P}_{(2n^2+4n)}$
  end for
end for
return $\mathbf{Z}$

Data availability

Data will be made available on request.

References

[1] Filipi Z, Fathy H, Hagena J, Knafl A, Ahlawat R, Liu J, et al. Engine-in-the-Loop testing for evaluating hybrid propulsion concepts and transient emissions. HMMWV Case Study 2006–01–0443; 2006. doi: https://doi.org/10.4271/2006-01-0443. https://www.sae.org/content/2006-01-0443/.

[2] Jung T, Kötter M, Schaub J, Quérel C, Thewes S, Hadj-Amor H, et al. Engine-in-the-loop: a method for efficient calibration and virtual testing of advanced diesel powertrains. In: Liebl J, editor. Simulation and test 2018. Wiesbaden: Springer Fachmedien Wiesbaden; 2019. p. 209–24. Series title: Proceedings. https://doi.org/10.1007/978-3-658-25294-6_12.

[3] Atkinson CM, Allain M, Kalish Y, Zhang H. Model-based control of diesel engines for fuel efficiency optimization 2009–01–0727; 2009. doi:https://doi.org/10.4271/2009-01-0727. https://www.sae.org/content/2009-01-0727/.

[4] Di Gioia R, Papaleo D, Vicchi FM, Cavina N. Virtual GDI engine as a tool for model-based calibration 2012–01–1679; 2012. doi:https://doi.org/10.4271/2012-01-1679. https://www.sae.org/content/2012-01-1679/.

[5] Meli M, Wang Z, Bailly P, Pischinger S. Neural network modeling of black box controls for internal combustion engine calibration. SAE technical paper series. STUT, SAE International; 2024. ISSN: 2688-3627. doi:https://doi.org/10.4271/2024-01-2995.

[6] Yoshida S, Ehara M, Kuroda Y. Rapid boundary detection for model based diesel engine calibration 2011–01–0741; 2011. doi: https://doi.org/10.4271/2011-01-0741. https://www.sae.org/content/2011-01-0741/.

[7] Castagné M, Bentolila Y, Chaudoye F, Hallé A, Nicolas F, Sinoquet D. Comparison of engine calibration methods based on design of experiments (DoE). Oil Gas Sci 2008;63(4):563–82. doi:https://doi.org/10.2516/ogst:2008029. http://ogst.ifpenergiesnouvelles.fr/10.2516/ogst:2008029.

[8] Yu X, Zhu L, Wang Y, Filev D, Yao X. Internal combustion engine calibration using optimization algorithms. Appl Energy 2022;305:117894. doi:https://doi.org/10.1016/j.apenergy.2021.117894. https://linkinghub.elsevier.com/retrieve/pii/S0306261921012071.

[9] Montgomery DC, editor. Design and analysis of experiments. 8th ed. Hoboken, NJ: John Wiley & Sons, Inc.; 2013.

[10] Zhu L, Wang Y, Pal A, Zhu G. Engine calibration using global optimization methods with customization 2020–01–0270; 2020. doi:https://doi.org/10.4271/2020-01-0270. https://www.sae.org/content/2020-01-0270/.

[11] Isermann R. Modellbasierte Entwicklung von Motorsteuerungen und -regelungen. MTZ - Motortechnische Zeitschrift 2014;75(3):78–87. doi:https://doi.org/10.1007/s35146-014-0070-9.

[12] Codd EF. A relational model of data for large shared data banks. Commun ACM 1970;13(6):377–87.

[13] Buitendijk RB, van der Lek H. Direct manipulation of a data dictionary with SQL. J Inf Syst 1991;16(3):323–33.

[14] Chen PP-S. The entity-relationship model-toward a unified view of data. ACM Trans Database Syst 1976;1(1):9–36.

[15] Lipton RJ, Naughton JF, Schneider DA, Seshadri S. Efficient sampling strategies for relational database operations. Theor Comput Sci 1993;116(1):195–226.

[16] Hagen H, Bonneau G-P. Variational design of smooth rational Bezier-surfaces. In: Geometric modelling. Springer; 1993. p. 133–8.

[17] Park RH. Two-reaction theory of synchronous machines generalized method of analysis—part I. Trans Am Inst Electr Eng 1929;48(3):716–27.

[18] Boggs PT, Tolle JW. Sequential quadratic programming. Acta Numer 1995;4:1–51.

[19] Alt W, Malanowski K. The Lagrange-Newton method for nonlinear optimal control problems. Comput Optim Appl 1993;2:77–100.

[20] Nakayama H, Sayama H, Sawaragi Y. A generalized Lagrangian function and multiplier method. J Optim Theory Appl 1975;17:211–27.

[21] Jorge N, Stephen JW. Numerical optimization. Spinger; 2006.

[22] Kuhn HW, Tucker AW. Nonlinear programming. In: Traces and emergence of nonlinear programming. Springer; 2013. p. 247–58.

[23] Aji SS, Kim YS, Ahn KY, Lee YD. Life-cycle cost minimization of gas turbine power cycles for distributed power generation using sequential quadratic programming method. Energies 2018;11(12):3511.

[24] Bohn C, Stober P, Magnor O. An optimization-based approach for the calibration of lookup tables in electronic engine control. In: 2006 IEEE conference on computer aided control system design, 2006 IEEE international conference on control applications, 2006 IEEE international symposium on intelligent control. IEEE; 2006. p. 2315–20.

[25] Li Y, Shi B, Andert J. Current and torque harmonics analysis of triple three-phase permanent-magnet synchronous machines with arbitrary phase shift based on model-in-the-loop 2024–01–3025. Stuttgart, Germany; 2024. doi: https://doi.org/10.4271/2024-01-3025. https://www.sae.org/content/2024-01-3025.

[26] Meli M, Wang Z, Sterlepper S, Picerno M, Pischinger S. Datasets of paper "data-driven parametric optimization for Pre-calibration of internal combustion engine controls"; 2024. doi:https://doi.org/10.5281/zenodo.13920146.

[27] Huber PJ. Robust statistics, vol. 523. John Wiley & Sons; 2004.

[28] Pyzdek T, Keller P. Six Sigma handbook. McGraw-Hill Education; 2014.

[29] Ruan D, Baldyga J. Identification of the driver demand torque map in engine ECU based on unscented Kalman filter. IFAC-PapersOnLine 2022;55(1):217–22.

[30] Pearson K VII. Note on regression and inheritance in the case of two parents. Proc R Soc Lond 1895;58(347–352):240–2.

[31] Karmarkar N. A new polynomial-time algorithm for linear programming. In: Proceedings of the sixteenth annual ACM symposium on theory of computing; 1984. p. 302–11.

[32] Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of ICNN '95-international conference on neural networks, vol. 4. IEEE; 1995. p. 1942–8.