12th IFAC International Symposium on
Advanced Control of Chemical Processes
July 14-17, 2024. Toronto, Canada

# Piecewise Linear Plus Quadratic Surrogate Model for Real-Time Optimization ⋆

**Duo Zhang** * **Xiang Li** *,** **Kody Kazda** ** **Zhijiang Shao** *

* *College of Control Science and Engineering, Zhejiang University, Hangzhou, 310027, China (e-mail: szj@zju.edu.cn)*
** *Department of Chemical Engineering, Queen's University, Kington, ON K7L 3N6, Canada (e-mail: xiang.li@queensu.ca)*

**Abstract:** Surrogate models are important in real-time optimization (RTO) when the first-principles model is unavailable or computationally challenging for online optimization. Among different surrogate models, the continuous piecewise linear (CPWL) model enjoys the universal approximation ability and potential computational benefits. However, the CPWL surrogate model poses three challenges to current RTO algorithms. First, the solution of a CPWL model is always located on the boundary of a polytopic subregion, while the plant optimum may be in the interior of a subregion. Second, the CPWL model is nonsmooth, which cannot be handled by RTO methods that rely on gradient matching. Third, the resulting nonsmooth optimization subproblems are hard to solve. This paper addresses the difficulties by adding a quadratic function to the CPWL surrogate model, extending a classical RTO method to accommodate nonsmoothness, and exploiting the difference-of-convex structure of the surrogate model for efficient solution. The advantages of the proposed method are demonstrated through a benchmark problem in the RTO literature.

*Keywords:* real-time optimization, piecewise linear approximation, surrogate, modifier adaptation, difference-of-convex

## 1. INTRODUCTION

With the widespread application of advanced control technology, real-time optimization (RTO) is regarded as an important tool that can further reduce operational costs and enhance competitiveness. Usually, RTO is executed based on a first-principles model derived from physical and chemical knowledge. However, there are two situations where RTO has to use a surrogate model. One is that the first-principles model is overly complex, and the solution takes a long time or easily fails, making online calculation difficult. Compared to the first-principles model, a surrogate model established using simulation data often has a simpler function structure. It also has a lower dimension because it does not include intermediate variables that do not explicitly appear in the RTO scheme (Chen et al., 2011; Biegler et al., 2014). Another situation is the lack of a ready-made first-principles model due to an insufficient understanding of the process. Alternatively, a data-driven model is derived from regression of measured data (Yang and Kelly, 2019).

Various surrogate models are used in real-time optimization or control, such as kriging (Gomes et al., 2008), neural network (Li et al., 2023), and continuous piecewise linear (CPWL) model (Gunnerud and Foss, 2010; Yang and Kelly, 2019). Among them, the CPWL model enjoys the universal approximation ability as well as potential computational benefits. Every continuous nonlinear function can be approximated on any compact set by a CPWL with an arbitrary accuracy (Gorokhovik et al., 1994). The CPWL model is especially beneficial when the plant or objective function has piecewise characteristics or the optimization problem already involves integer variables.

The use of surrogate models, inaccuracy of the original first-principles model, and changes in plant characteristics can lead to plant-model mismatch. As a result, the optimal values calculated based on the surrogate model usually differ from the plant's optimal values. The RTO literature has developed different approaches to compensating for model mismatch (Srinivasan and Bonvin, 2019). One popular and straightforward way is modifier adaptation (MA) (Marchetti et al., 2009). The idea is to adapt the model in each iteration according to the plant's first-order local information so that the plant's KKT conditions match that of the model upon convergence. However, there are three challenges when applying the modifier adaptation technique to CPWL surrogate models:

1) The minimum of a CPWL function is achieved on the boundary of a polytopic subregion. If the plant optimum is located in the interior of a subregion, the solutions of the CPWL surrogate will not converge to the plant optimum, and most likely, they will oscillate near the plant optimum.

2) A CPWL model is nonsmooth. In each iteration of the MA algorithm, the model is adapted according to plant measurements so that its gradient equals the plant's at the

⋆ We gratefully acknowledge financial support from National Natural Science Foundation of China with grant numbers 62120106003, 62173301 and China Scholarship Council with award number 202006320176.

current operating point. If the model is not differentiable, how to adapt it appropriately is unknown.

3) A CPWL model leads to nonsmooth optimization subproblems, which can be hard to solve, especially for real-time applications.

In this paper, we overcome the above challenges by adding a convex quadratic function to the CPWL surrogate (Section 2), extending the standard MA to a nonsmooth MA (Section 3), and exploiting the difference-of-convex structure of CPWL function in order to locate an optimum efficiently using the difference-of-convex algorithm (Section 4). We demonstrate the advantages of the proposed surrogate model and the solution method through a benchmark problem in Section 5 and provide concluding remarks in Section 6.

## 2. PIECEWISE LINEAR PLUS QUADRATIC SURROGATE

### *2.1 Continuous Piecewise linear function*

*Definition 1, CPWL function (Gorokhovik et al., 1994)* Let $D \subseteq \mathbb{R}^n$ be a compact set. A function $f : D \to \mathbb{R}$ is a CPWL function if and only if there exists $\{w_i\}_{i=1}^k \subseteq \mathbb{R}^n$, $\{v_i\}_{i=1}^k \subseteq \mathbb{R}$, and a polyhedral partition $\{P_i\}_{i=1}^k$ of $D$ such that:

$$f^{CPWL}(x) = x^T w_i + v_i, \quad \forall x \in P_i, \quad i = 1, \cdots, k. \quad (1)$$

CPWL function has many nice properties, and an important one is the universal approximation ability. (Gorokhovik et al., 1994). In other words, the space of CPWL functions is dense in the space of continuous functions.

According to Theorem 3.1 in (Gorokhovik et al., 1994), every CPWL function $f$ on a compact set $D \subseteq \mathbb{R}^n$ can be written as the difference of two convex CPWL functions as in Eq. (2). In other words, it is a difference-of-convex (DC) function. The superscripts $+$ and $-$ indicate the positive and negative components of the DC function, respectively. $p^+$ and $p^-$ are the numbers of linear pieces in the two convex components. The order of a DC CPWL function is $(p^+, p^-)$.

$$\begin{aligned} f^{CPWL}(x) = &\max_{i \in \{1,\cdots,p^+\}} \{x^T w_i^+ + v_i^+\} - \\ &\max_{i \in \{1,\cdots,p^-\}} \{x^T w_i^- + v_i^-\}. \end{aligned} \quad (2)$$

The DC representation of the CPWL function can be exploited for efficient optimization (see Section 4 for details).

### *2.2 Adding a Quadratic Function*

By adding a quadratic function to a CPWL function, we get a continuous piecewise linear plus quadratic (CPWL-Q) function as follows:

$$f^{CPWL-Q}(x) = f^{CPWL}(x) + f^Q(x), \quad (3)$$

$$f^Q(x) = x^T A x + b^T x + c, \quad (4)$$

where $f^{CPWL}(x)$ is the CPWL part of the CPWL-Q function and $f^Q(x)$ is the quadratic part of the CPWL-Q function. The CPWL-Q surrogate also has the universal approximation property. We require that $f^Q(x)$ be strictly convex such that no additional nonconvexity is added into the model. In addition, this implies that the CPWL-Q function has positive-definite second-order derivatives almost everywhere, which is beneficial for the convergence property of RTO algorithms such as the MA (Marchetti et al., 2009). The optimum of a CPWL function is always located on the boundary of a polytopic subregion, and this restriction prevents the CPWL optimum from reaching the plant optimum when the latter is located in the interior of a subregion. After adding $f^Q(x)$ to the CPWL function, this restriction is not present, and the solutions of the CPWL-Q surrogate are less likely to oscillate near the plant optimum.

### *2.3 Fitting CPWL-Q surrogate*

We follow the following three steps to construct the CPWL-Q surrogate.

(1) Generate a dataset from simulation data of the first-principles model or historical measurement data. Specify the surrogate model's error tolerance $\epsilon_s$.
(2) Approximate a convex quadratic model by the method in (Rosen and Marcia, 2004). This step determines $f^Q$ in Eq. (3).
(3) Calculate the residue errors between the data and the convex quadratic model. Fit a CPWL function to the residue data by the method in (Kazda and Li, 2023) with error tolerance $\epsilon_s$. This step determines $f^{CPWL}$ in Eq. (3).

In the CPWL fitting step, problem (5) is solved iteratively, where $s_l^+$ and $s_l^-$ are slack variables indicating violation of error bounds. The objective function will equal zero if the approximation error is within the tolerance. The approximation algorithm starts from a low-order DC CPWL function with $(p^+, p^-) = (1, 1)$ and gradually increases the order of the DC CPWL function until the desired approximation error $\epsilon_s$ is achieved.

$$\begin{aligned} \min \quad & \sum_{l \in \{1,\ldots,n_q\}} (s_l^+ + s_l^-) \\ \text{s.t.} \quad & f^{TRUE}(x_l) \le f(x_l) + \epsilon_s + s_l^-, \quad \forall l \in \{1, ..., n_q\}, \\ & f^{TRUE}(x_l) \ge f(x_l) - \epsilon_s - s_l^+, \quad \forall l \in \{1, ..., n_q\}, \\ & s_l^+, s_l^- \ge 0, \quad \forall l \in \{1, ..., n_q\}, \\ & f \in \text{DC CPWL of order } (p^+, p^-). \end{aligned} \quad (5)$$

Fig. 1 illustrates the progressive fitting method by a bivariate function. With the order of DC CPWL increasing, the polytopic partition is adjusted and refined, and the maximum gap between the true function and its approximator decreases. The process continues until the maximum error is below the tolerance.

We chose the method in (Kazda and Li, 2023) to generate the CPWL part because of its several advantages. (1) It handles multivariate models and yields CPWL with relatively few pieces compared to other methods that are based on hyperrectangle and triangulation. (2) It can approximate the nonlinear function to any predefined precision. (3) It directly generates CPWL functions in the DC form in Eq. (2), which favors efficient online calculation.

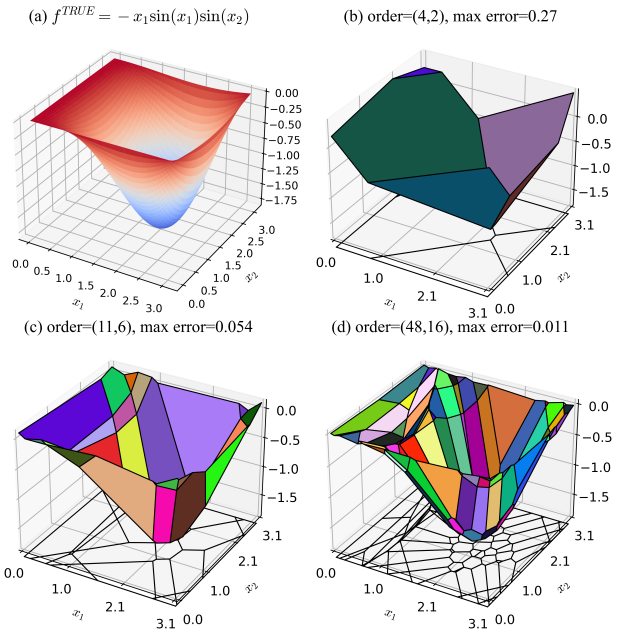


Fig. 1. DC CPWL approximation with increasing orders

## 3. NONSMOOTH MODIFIER ADAPTATION

This paper discusses the RTO algorithm that aims to find the most profitable steady state of a process. We consider the plant optimization problem in the form of Eq. (6), where $u$ is the decision variable. $\phi_p$ is the objective function, and $g_p$ is the inequality constraint function. Considering notational simplicity, we assume there is only one inequality constraint. The situation with multiple inequality constraints is similar.

$$\min_u \ \phi_p(u) \quad \text{s.t. } g_p(u) \leq 0. \tag{6}$$

Since we do not know the exact information of the plant, the optimization problem that is actually solved in RTO is the following model optimization problem:

$$\min_u \ \phi_m(u) \quad \text{s.t. } g_m(u) \leq 0, \tag{7}$$

where $\phi_m$ and $g_m$ come from the surrogate model of the system. Assume that the nonlinearity of the problem comes solely from the nonlinear process and the CPWL-Q surrogate model is used to approximate the process, then $\phi_m$ and $g_m$ can be expressed in the following DC form:

$$\begin{aligned} \phi_m(u) &= \phi_m^+(u) - \phi_m^-(u) + u^T A_\phi u + b_\phi^T u + c_\phi, \\ g_m(u) &= g_m^+(u) - g_m^-(u) + u^T A_g u + b_g^T u + c_g, \end{aligned} \tag{8}$$

where $\phi_m^+$, $\phi_m^-$, $g_m^+$, $g_m^-$ are convex CPWL functions.

Due to model mismatch caused by model approximation and/or process uncertainty, the optimum of problem (7) is often not optimal for the plant. Therefore, the model $\phi_{m,k}$ and $g_{m,k}$ are adapted in each iteration $k$ to match the plant at the current input $u_k$. Hopefully, the optimum of problem (9) coincides with the plant's at the convergence of the algorithm.

$$\min_u \ \phi_{m,k}(u) \quad \text{s.t. } g_{m,k}(u) \leq 0. \tag{9}$$

### 3.1 Modifier adaptation algorithm

There are many ways to update the model $\phi_{m,k}$ and $g_{m,k}$. Among them, MA is one of the simplest in handling structural plant-model mismatches and has successful applications in the industry. The model is adapted at each iteration to be a good first-order approximation of the plant. Given a continuously differentiable model, the modifier adaptation algorithm is described as follows.

*Algorithm 1: Modifier adaptation algorithm (Marchetti et al., 2009)*

**Step 0: Initialization.** Choose an starting point $u_0$ and filter coefficient $0 < K < 1$. $k \leftarrow 0$.

**Step 1: Get plant measurements.** Implement $u_k$ to the plant and obtain $\phi_p(u_k)$, $\nabla\phi_p(u_k)$, $g_p(u_k)$, $\nabla g_p(u_k)$.

**Step 2: Gradient modifier calculation.** Calculate the output modifiers $\epsilon^{\cdot}$ gradient modifiers $\lambda^{\cdot}$ by Eq. (10).

$$\begin{aligned} \epsilon_k^\phi &= \phi_p(u_k) - \phi_m(u_k), \\ \epsilon_k^g &= g_p(u_k) - g_m(u_k), \\ \lambda_k^\phi &= \nabla\phi_p(u_k) - \nabla\phi_m(u_k), \\ \lambda_k^g &= \nabla g_p(u_k) - \nabla g_m(u_k). \end{aligned} \tag{10}$$

**Step 3: Model adaptation.** Adapt the model according to Eq. (11).

$$\begin{aligned} \phi_{m,k}(u) &= \phi_m(u) + \epsilon_k^\phi + (u - u_k)^T \lambda_k^\phi, \\ g_{m,k}(u) &= g_m(u) + \epsilon_k^g + (u - u_k)^T \lambda_k^g. \end{aligned} \tag{11}$$

**Step 4: Step calculation.** Solve the model optimization problem (9) and denote the solution as $u_k^*$.

**Step 5: Input filtering.** Filter the input by Eq. (12).

$$u_{k+1} = u_k + K(u_k^* - u_k). \tag{12}$$

$k \leftarrow k + 1$. Go back to Step 1.

The optimization begins at the initial point $u_0$. In Step 1, the local plant information at $u_k$ is obtained. Steps 2 and 3 calculate the adapted model, which is the original model plus a linear function. $\lambda_k^{\cdot}$ is called a gradient modifier because it modifies the model's gradient. The updated model is a first-order local approximation of the plant. Step 4 calculates the optimal input. Step 5 filters the input to enhance convergence. Theorem 1 in (Marchetti et al., 2009) proves that MA can reach a KKT point of the plant if the algorithm converges despite structural plant-model mismatch under assumptions A1 and A2.

**Assumption A1**. $\phi_p$ and $g_p$ are continuously differentiable.

**Assumption A2**. $\phi_m$ and $g_m$ are continuously differentiable.

However, assumption A2 excludes the CPWL or CPWL-Q surrogate. This paper relaxes this assumption and extends Algorithm 1 for nonsmooth surrogate models.

*3.2 Extension using the Clarke generalized gradient*

Like in MA, our basic idea is that in each iteration, the zeroth- and the first-order information of model objective and constraint functions match that of the plant. One major difference is that $\nabla\phi_m$ and $\nabla g_m$ are replaced by the Clarke generalized gradient.

*Clarke generalized gradient* (Clarke, 1981) Let $f:\mathbb{R}^n\to\mathbb{R}$ be a locally Lipschitz continuous function. The Clarke generalized gradient of $f$ at a point $x\in\mathbb{R}^n$ is denoted as $\partial f(x)$ and is defined by Eq. (13):

$$\partial f(x)=\left\{\xi\in\mathbb{R}^n \mid \xi^T v\le f^\circ(x,v),\quad \forall v\in\mathbb{R}^n\right\} \tag{13}$$

, where $f^\circ(x,v)$ is defined by Eq. (14).

$$f^\circ(x,v)=\limsup_{y\to x,h\downarrow 0}\frac{f(y+hv)-f(y)}{h} \tag{14}$$

For convex functions, a Clarke generalized gradient is a subdifferential. For differentiable functions, it reduces to a gradient. With the definition of Clarke generalized gradient, Algorithm 1 can be extended to the nonsmooth models that satisfy the following Assumption A3.

**Assumption A3**. $\phi_m$ and $g_m$ are continuous, and their Clarke generalized gradient exists.

*Algorithm 2: Nonsmooth modifier adaptation*

**Step 0: Initialization.** Choose an starting point $u_0$ and filter coefficient $0<K<1$. $k\leftarrow 0$.

**Step 1: Get plant measurements.** Implement $u_k$ to the plant and obtain $\phi_p(u_k)$, $\nabla\phi_p(u_k)$, $g_p(u_k)$, $\nabla g_p(u_k)$.

**Step 2: Step calculation.** Minimize problem (15) and denote the solution as $u_k^*$.

$$\begin{aligned}\min_{u,\lambda^\phi,\lambda^g}\ & \phi_{m,k}(u)=\phi_m(u)+\epsilon_k^\phi+(u-u_k)^T\lambda^\phi\\ s.t.\ & g_{m,k}(u)=g_m(u)+\epsilon_k^g+(u-u_k)^T\lambda^g\le 0\\ & \lambda^\phi\in\nabla\phi_p(u_k)-\partial\phi_m(u_k)\\ & \lambda^g\in\nabla g_p(u_k)-\partial g_m(u_k)\end{aligned} \tag{15}$$

**Step 3: Input filtering.** Filter the input by Eq. (12). $k\leftarrow k+1$. Go back to Step 1.

Steps 0 and 1 in the nonsmooth MA are the same as those in Algorithm 1. Step 2 is a combination of Steps 2-4 in Algorithm 1. Since the Clarke generalized gradient is a set, it needs to be determined which element is chosen to represent the first-order information of the model. By incorporating gradient modifier calculation into the optimization program, the "best" one is selected to calculate the first-order modifier that leads to the maximum optimization improvement. Step 3 filters the input to reduce the step size and improves global convergence.

It can be shown that under A1 and A3, Algorithm 2 preserves the upon-convergence-optimality property of MA. The proof is provided in another paper that is under preparation. If the model satisfies assumption A2, Algorithm 2 reduces to Algorithm 1 because problem (15) has only one feasible $\lambda$ that can be trivially computed by Eq. (10). Therefore, Algorithm 2 is a generalization of the classic modifier adaptation algorithm.

For a CPWL function, fundamental result 8 by Clarke (1981) guarantees that the Clarke generalized gradient at a point can be represented by the convex hull of the gradients of the active pieces at that point. So the Clarke generalized gradient of $f$ in Eq. (1) can be written as

$$\partial f(x)=co\{w_i:\ i\in I_f(x)\}, \tag{16}$$

where *co* denotes the convex hull, and $I_f(x)$ is the index set for active pieces of $f$ at $x$. With this result, the last two constraints in problem (15) can be replaced by constraints (17), which include new variables $\gamma$.

$$\begin{aligned}\lambda^\phi&=\sum_{j\in I_{\phi_m}(u_k)}\gamma_{\phi,j}w_{\phi,j}\\ \lambda^g&=\sum_{j\in I_{g_m}(u_k)}\gamma_{g,j}w_{g,j}\\ 0\le\gamma_{\phi,j}&\le 1,\quad j\in I_{\phi_m}(u_k)\\ 0\le\gamma_{g,j}&\le 1,\quad j\in I_{g_m}(u_k)\end{aligned} \tag{17}$$

## 4. SOLVING THE MODEL OPTIMIZATION PROBLEM

Traditionally, the optimization problem with the CPWL function is formulated as a mixed-integer program (Yang and Kelly, 2019). With the integer variable representing the activity of pieces, problem (15) becomes a mixed-integer nonlinear program. This approach has two disadvantages. First, solving mixed-integer nonlinear programs can be too slow for RTO. Second, it takes work to determine an appropriate value for the big-M parameters in the mixed-integer nonlinear program to ensure unbiasedness.

The difference-of-convex algorithm (DCA) proposed by Le Thi and Pham Dinh (2018) is another approach to solving problem (15), taking advantage of the difference-of-convex decomposition of surrogate models. Standard DCA solves the inequality-constrained problem where the objective and constraint functions are both DC functions. Such functions are vast in engineering (Le Thi and Pham Dinh, 2018), including the CPWL and CPWL-Q surrogates. Compared to the mixed-integer approach, DCA only finds a local minimum. However, it is usually enough for RTO applications, and the multi-start technique can enhance the solution quality. The procedure of solving problem (15) by DCA is described in Algorithm 3. Due to the page limit, we skip the derivation details and only provide the final algorithm here.

*Algorithm 3: DCA algorithm for problem* (15)

**Step 0: Initialization.** Choose convergence tolerance $\epsilon_{dca}>0$. $l\leftarrow 0$. Find a feasible starting point $u_{k,l}$ for problem (15).

**Step 1: Solving DCA subproblem.** Compute $\alpha_\phi\in\partial\phi_m^-(u_{k,l})$. Compute $\alpha_g\in\partial g_m^-(u_{k,l})$. Solve DCA subproblem (18). Denote the solution as $u_{k,l+1}$.

$$\begin{aligned}\min_{u,\lambda^\phi,\lambda^g,\beta_\phi^+,\beta_g^+} \quad & \beta_\phi^+ - \beta_\phi^-(u) + \beta_\phi^q(u) + \epsilon_k^\phi + (u-u_k)^T\lambda^\phi \\ \text{s.t. } & \beta_g^+ - \beta_g^-(u) + \beta_g^q(u), + \epsilon_k^g + (u-u_k)^T\lambda^g \le 0 \\ & \beta_\phi^+ \ge u^T w_{\phi,j}^+ + v_{\phi,j}^+, j = 1,\cdots,p_\phi^+, \\ & \beta_g^+ \ge u^T w_{g,j}^+ + v_{g,j}^+, j = 1,\cdots,p_g^+, \\ & \text{constraints (17)},\end{aligned} \tag{18}$$

where

$$\begin{aligned}\beta_\phi^-(u) &:= \alpha_\phi^T(u-u_{k,l}) + \phi_m^-(u_{k,l}), \\ \beta_\phi^q(u) &:= u^T A_\phi u + b_\phi^T u + c_\phi, \\ \beta_g^-(u) &:= \alpha_g^T(u-u_{k,l}) + g_m^-(u_{k,l}), \\ \beta_g^q(u) &:= u^T A_g u + b_g^T u + c_g.\end{aligned} \tag{19}$$

**Step 2: Convergence check.** If $\|u_{k,l+1} - u_{k,l}\| < \epsilon_{dca}$, optimum found and return $u_k = u_{k,l+1}$. Otherwise, $l \leftarrow l+1$ and go back to Step 1.

In the initialization step, a feasible initial point and the error tolerance are specified. In Step 1, the algorithm solves a convex upper-bounding problem (18). The problem is constructed according to the DC structure of $\phi_m$ and $g_m$; specifically, some subgradients of the negative convex components of $\phi_m$ and $g_m$ (i.e., $\alpha_\phi$, $\alpha_g$ in (19)) are used to bound the negative convex components from above. Step 1 is repeated until the desired solution precision is achieved.

Problem (18) can be written as a second-order cone program (SOCP) (Lobo et al., 1998), which can be solved reliably by specialized solvers. The DCA approach has several significant advantages over the mixed-integer approach. First and foremost, it is much faster. Second, it does need to specify the big-M value, which could be hard to determine for complex models. Third, it reduces the requirements for the solver, only needing a SOCP solver, which is attractive for RTO implementation.

## 5. CASE STUDY

The performance of the RTO algorithm using CPWL-Q surrogate is illustrated by a benchmark William-Otto reactor (Forbes and Marlin, 1996). The CPWL approximation problems and problem (15) were solved by CPLEX Optimization Studio 22.1 (IBM, 2023). The other nonlinear programs or equations were solved by IPOPT 3.14.3 (Wächter and Biegler, 2006). The simulation was performed on a Windows 10 system with a 2.80 GHz 11th Gen Intel CPU and 24 GB RAM.

The process consists of a stirred tank reactor. Three reactions occur that involve six components: A, B, C, E, G, and P. A and B are fed into the reactor with flowrates $F_A$ and $F_B$, respectively. The output stream is denoted by R, and its flowrate is $F_R$. E and P are the two valuable products in the mixture.

$$\begin{aligned} A + B &\xrightarrow{k_1} C \qquad k_1 = 1.660\times10^6\exp(-666.4/T_R) \\ B + C &\xrightarrow{k_2} P + E \quad k_2 = 7.212\times10^8\exp(-8333.3/T_R) \\ P + C &\xrightarrow{k_3} G \qquad k_3 = 2.675\times10^{12}\exp(-11111/T_R)\end{aligned}$$

The process model neglects intermediate product C and has only two reactions:

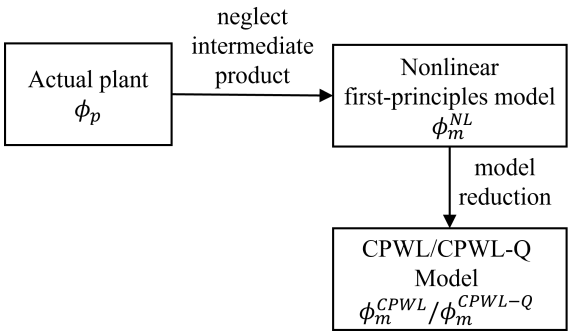


Fig. 2. Relations between the plant, the first-principles model, and the surrogate model

$$\begin{aligned} A + 2B &\xrightarrow{k_1'} P + E \quad k_1' = 2.19\times10^8\exp(-8.08\times10^3/T_R) \\ A + B + P &\xrightarrow{k_2'} G \quad k_2' = 4.31\times10^{10}\exp(-1.24\times10^4/T_R)\end{aligned}$$

The optimization problem is Eq. (20), which minimizes the plant cost by determining flow rate $F_B$ and temperature $T_R$.

$$\begin{aligned}\min_{F_B,T_R} \quad & -1143.38x_P F_R - 25.92x_E F_R + 76.23F_A + 114.34F_B \\ \text{s.t. } & (F_R, x_p, x_E) = h(F_B, T_R), \\ & F_B \in [3,6], T_R \in [70,100].\end{aligned} \tag{20}$$

Flow rate $F_A$ is fixed. The input-output mapping $h$ relates the two decision variables and the outlet flow rate $F_R$ and mass fractions $x_p$, $x_E$. Due to the omission of the intermediate product and associated reaction, the actual plant and the nonlinear first-principles model have different $h$.

For convenience of RTO implementation, Problem (20) is written into the following forms so that only the decision variables are involved explicitly in the formulation:

$$\min_{(F_B,T_R)\in[3,6]\times[70,100]} \phi_p(F_B,T_R) \tag{21}$$

$$\min_{(F_B,T_R)\in[3,6]\times[70,100]} \phi_m^{NL}(F_B,T_R) \tag{22}$$

Problem (21) includes the actual plant with three reactions. Problem (22) includes the mismatched nonlinear first-principles model that assumes two reactions. The calculation of $\phi_m^{NL}$ includes intermediate variables that come from the first-principles. We can eliminate the need for intermediate variables by generating a surrogate for $\phi_m^{NL}$. By 10000 random points generated in the domain of the decision variables using Latin hypercube sampling (McKay et al., 2000), we approximate $\phi_m^{NL}$ by a CPWL surrogate $\phi_m^{CPWL}$ using the method in Kazda and Li (2023) and a CPWL-Q surrogate $\phi_m^{CPWL-Q}$ using the method shown in Section 2.2. The error tolerance $\epsilon = 1$.

The relations between the plant, the first-principles model, and the surrogate model are shown in Fig. 2. The first-principles model and the two surrogate models have structural mismatches. Each surrogate model involves two sources of mismatch: inaccurate reaction modeling and model reduction. The RTO algorithms are expected to converge to the plant optimum using any of the three mismatched models.

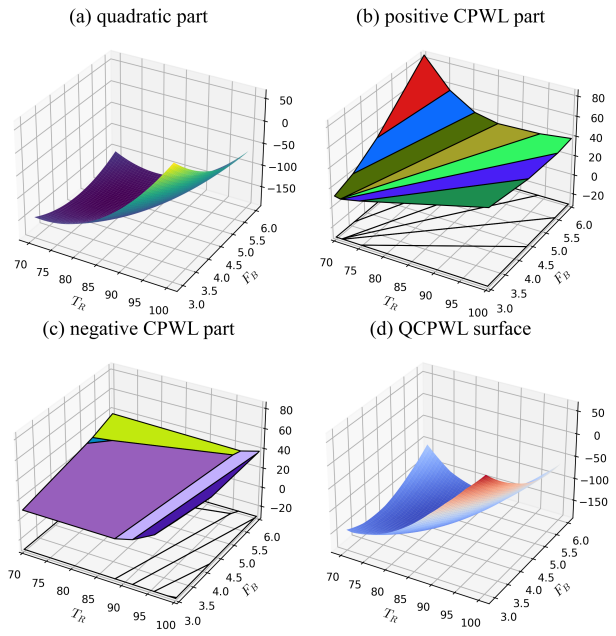


Fig. 3. Illustration of $\phi_m^{CPWL-Q}$ for the reactor example

First, we compare the model fitting results of the two surrogate models in Table 1. The generated CPWL functions are in the DC form, so the column "number of pieces" shows the number of pieces in the positive convex CPWL component plus the number of pieces in the negative convex CPWL component. We can see that both surrogate models reach the target precision, while the CPWL-Q model has fewer pieces and requires a much shorter generation time. The reason is that the quadratic part of CPWL-Q has already accounted for some nonlinearity, so the CPWL part of the CPWL-Q surrogate is less complex than the CPWL surrogate.

Table 1. Model fitting results of the William-Otto reactor example

| model | maximum error | number of pieces | CPU time/s |
|---|---|---|---|
| CPWL | 0.994 | 47+1 | 995.05 |
| CPWL-Q | 0.993 | 7+7 | 51.80 |

Fig. 3 illustrates the CPWL-Q surrogate. Subplot (a) is the convex quadratic part. Subplots (b) and (c) are the positive and negative components of the CPWL part. We can see that the shapes of CPWL subregions are not limited to rectangles or simplices, which would otherwise lead to more pieces for the given tolerance. Combining the three parts yields subplot (d), which shows that the fitted surrogate model has good continuity.

Next, the performances of RTO using the first-principles model and two surrogate models are accessed. To highlight the impact of using different models, we assume that the measurements of the plant outputs and gradients are noise-free. RTO iteration begins from $u_0 = [4, 75]$. $F_A$ is fixed at 1.8275 (kg/s). In each RTO iteration, Algorithm 1 is used to adapt the first-principles model, and Algorithm 2 is used to adapt the two nonsmooth surrogate models.

Fig. 4 shows the profiles of input variables and the plant objective function. The plant optimum is the gray dashed line. The blue dash-dotted line presents the RTO profile

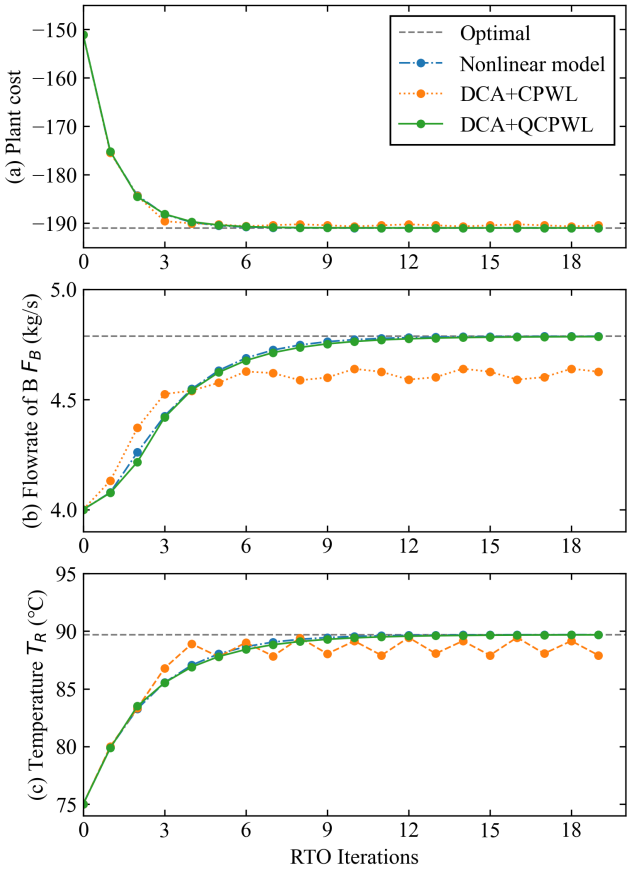


Fig. 4. RTO Iteration profiles of the reactor example

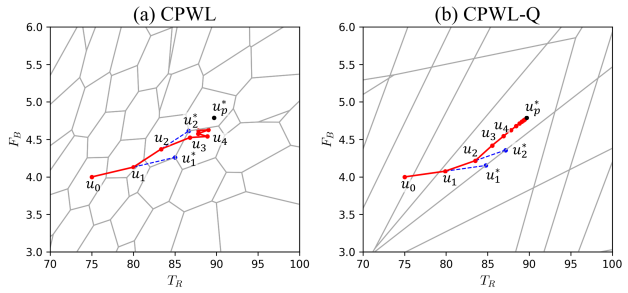


Fig. 5. Input decisions over the RTO iterations based on the CPWL surrogate and the CPWL-Q surrogate. The grey lines are boundaries of the CPWL polytopic partitions. The plant optimum is indicated by $u_p^*$.

using the first-principles model. The profile using the CPWL surrogate model is the orange dotted line, and the green solid line shows the CPWL-Q surrogate model. The iteration profile of the CPWL-Q model overlaps that of the nonlinear first-principles model, and in both cases, the solutions converge to the plant optimum. This indicates that, although the CPWL-Q surrogate differs slightly from the first-principles model, it did not deteriorate the RTO performance. However, RTO using the CPWL surrogate yielded an oscillation. This is because the CPWL model restricts the model optimum to the vertices of its polytopic subregions, which do not coincide with the plant optimum.

Fig. 5 demonstrates the reason why CPWL and CPWL-Q surrogate models performed differently. The blue points

$u_1^*$, $u_2^*$ are the optimal inputs calculated by solving problem (15). They are modified by input filtering (i.e., Step 3 of Algorithm 2) and implemented in the plant. The red points mark the implemented input values. Here, the filter coefficient $K = 0.5$, so $u_k$ is the midpoint of the line segment connecting $u_{k-1}$ and $u_k^*$. It is observed that the blue points in Fig. 5(a) always lie at the vertices of the polytopic subregions, so it always misses the plant optimum $u_p^*$ that is in the interior of a polytope. Because of the convex quadratic part, the CPWL-Q surrogate avoided this deficiency. In addition, it also has fewer polytopic subregions than the CPWL surrogate.

Table 2 shows the RTO computational results of the first-principles model and the CPWL-Q surrogate. The CPU time for each model includes the time for solving problem (9) and the time for calculating the model gradient. We can see that the RTO with the CPWL-Q surrogate is more than ten times faster than that with the first-principles model. The short computational time is attributed to the lower dimension of the surrogate. Moreover, solving the CPWL-Q model using DCA only requires a SOCP solver, which has a more solid convergence guarantee and is more robust than the general nonlinear solver needed for the first-principles model.

Table 2. RTO computational results of the William-Otto reactor example

| model type | number of variables | optimization algorithm | CPU time/s |
|---|---|---|---|
| First-principles | 10 | IPOPT | 2.88 |
| CPWL-Q | 2 | DCA | 0.18 |

## 6. CONCLUSION

This paper proposes to use the CPWL-Q surrogate model with the DC structure in RTO. Compared to the CPWL surrogate, the CPWL-Q surrogate has lower complexity because the quadratic part already accounts for certain nonlinearity. The CPWL-Q surrogate also has better convergence properties because its extreme values are not restricted to the vertices of polyhedral subregions. By using the CPWL-Q surrogate instead of the first-principles model, the RTO solution time can be significantly reduced because many intermediate variables used in the first-principles model are eliminated. Plant-model mismatches are handled by the extended modifier adaptation scheme, where the Clarke generalized gradient is used to address the nonsmoothness. By using the DCA algorithm, solving the CPWL-Q model-based RTO problem only requires a SOCP solver rather than a general nonlinear optimization solver that is more complex. In future work, we plan to consider more complicated process systems for which we believe the advantages of the CPWL-Q surrogate would be more significant.

## REFERENCES

Biegler, L.T., Lang, Y.d., and Lin, W. (2014). Multi-scale optimization for process systems engineering. *Computers & Chemical Engineering*, 60, 17–30.

Chen, T., Hadinoto, K., Yan, W., and Ma, Y. (2011). Efficient meta-modelling of complex process simulations with time–space-dependent outputs. *Computers & chemical engineering*, 35(3), 502–509.

Clarke, F.H. (1981). Generalized gradients of lipschitz functionals. *Advances in Mathematics*, 40(1), 52–67.

Forbes, J.F. and Marlin, T.E. (1996). Design cost: A systematic approach to technology selection for model-based real-time optimization systems. *Computers & Chemical Engineering*, 20(6-7), 717–734.

Gomes, M.V., Bogle, I.D.L., Biscaia Jr, E.C., and Odloak, D. (2008). Using kriging models for real-time process optimisation. In *Computer Aided Chemical Engineering*, volume 25, 361–366. Elsevier.

Gorokhovik, V.V., Zorko, O.I., and Birkhoff, G. (1994). Piecewise affine functions and polyhedral sets. *Optimization*, 31(3), 209–221.

Gunnerud, V. and Foss, B. (2010). Oil production optimization—a piecewise linear model, solved with two decomposition strategies. *Computers & Chemical Engineering*, 34(11), 1803–1812.

IBM (2023). Ibm ilog cplex optimization studio 22.1.0. `https://www.ibm.com/docs/es/icos/22.1.0`. Accessed: 2023-07-09.

Kazda, K. and Li, X. (2023). A linear programming approach to difference-of-convex piecewise linear approximation. *European Journal of Operational Research*, 312(2), 493–511.

Le Thi, H.A. and Pham Dinh, T. (2018). Dc programming and dca: thirty years of developments. *Mathematical Programming*, 169(1), 5–68.

Li, H., Wang, W., Wang, Y., Li, C., Wang, Y., Zhu, Z., Cui, P., Li, X., and Li, Y. (2023). Dynamic real-time energy saving control of pressure-swing distillation based on artificial neural networks. *Chemical Engineering Science*, 282, 119271.

Lobo, M.S., Vandenberghe, L., Boyd, S., and Lebret, H. (1998). Applications of second-order cone programming. *Linear algebra and its applications*, 284(1-3), 193–228.

Marchetti, A., Chachuat, B., and Bonvin, D. (2009). Modifier-adaptation methodology for real-time optimization. *Industrial & Engineering Chemistry Research*, 48(13), 6022–6033.

McKay, M.D., Beckman, R.J., and Conover, W.J. (2000). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1), 55–61.

Rosen, J.B. and Marcia, R.F. (2004). Convex quadratic approximation. *Computational Optimization and Applications*, 28, 173–184.

Srinivasan, B. and Bonvin, D. (2019). 110th anniversary: a feature-based analysis of static real-time optimization schemes. *Industrial & Engineering Chemistry Research*, 58(31), 14227–14238.

Wächter, A. and Biegler, L.T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106, 25–57.

Yang, Y. and Kelly, J. (2019). Efficient real time optimization using a data-driven piecewise affine model. *Computers & Chemical Engineering*, 125, 545–557.