Journal of Global Optimization (2020) 77:455–486
https://doi.org/10.1007/s10898-020-00881-4

Mathematical programming formulations for piecewise polynomial functions

Bjarne Grimstad[1,2] · Brage R. Knudsen[1,3]

Received: 1 May 2019 / Accepted: 18 January 2020 / Published online: 3 February 2020
© The Author(s) 2020

Abstract
This paper studies mathematical programming formulations for solving optimization problems with piecewise polynomial (PWP) constraints. We elaborate on suitable polynomial bases as a means of efficiently representing PWPs in mathematical programs, comparing and drawing connections between the monomial basis, the Bernstein basis, and B-splines. The theory is presented for both continuous and semi-continuous PWPs. Using a disjunctive formulation, we then exploit the characteristic of common polynomial basis functions to significantly reduce the number of nonlinearities, and to suggest a bound-tightening technique for PWP constraints. We derive several extensions using Bernstein cuts, an expanded Bernstein basis, and an expanded monomial basis, which upon a standard big-M reformulation yield a set of new MINLP models. The formulations are compared by globally solving six test sets of MINLPs and a realistic petroleum production optimization problem. The proposed framework shows promising numerical performance and facilitates the solution of PWP-constrained optimization problems using standard MINLP software.

Keywords Piecewise polynomials · Splines · Mixed integer programming · Nonlinear programming · Disjunctions

1 Introduction

Modeling of optimization problems frequently involves representing functions that are piecewise, discontinuous or nonsmooth. This includes inherently piecewise economical and physical characteristics [5,24,29], construction of surrogate models by sampling of simulators [15,32,58], and approximate or exact representation of nonconvex functions [2,10,37,39,47]. In this paper, we study the problem of efficiently representing and solving optimization prob-

✉ Bjarne Grimstad
bjarne.grimstad@gmail.com

Brage R. Knudsen
brage.knudsen@sintef.no

1 Department of Engineering Cybernetics, NTNU, 7034 Trondheim, Norway

2 Present Address: Solution Seeker, 0349 Oslo, Norway

3 Present Address: SINTEF Energy Research, 7491 Trondheim, Norway

lems containing piecewise polynomial (PWP) constraints. Piecewise polynomials are used in a wide range of disciplines, including efficiency curve modeling in electric-power unit commitment [40], rigid motion systems [11], image processing and data compression [44,51], probability density estimation [61], flow networks [5,15,25] and in optimal control [4,41].

We consider optimization problems where either or both of the objective function and a subset of the constraints are piecewise polynomial functions. Each polynomial may be nonconvex, and the piecewise polynomial function itself lower semi-continuous. There exist few targeted optimization methods for this class of optimization problems, while some approaches that exploit special structures of nonsmooth optimization problems are applicable, subject to certain modification methods: Womersley and Fletcher [62] developed a descent method for solving composite nonsmooth problems consisting of a finite number of smooth functions. Conn and Mongeau [8] constructed a method based on non-differentiable penalty functions for solving discontinuous piecewise linear optimization problems, sketching an extension to problems with PWP constraints. Scholtes [47] developed an active-set method for dealing with nonlinear programs (NLPs) with underlying combinatorial structure in the constraints. Li [30] used a conjugated gradient method for minimizing an unconstrained, strictly convex, quadratic spline. None of these methods are currently available in standard optimization software.

From a broader perspective, applicable solution approaches to PWP optimization problems include methods based on general nonsmooth optimization, smoothing techniques and mixed integer programming (MIP). Bundle-type and subgradient methods [21], originally developed for nonsmooth convex optimization, may be applied to optimization problems with general nonsmooth structures such as PWPs through Clark's generalized gradients [48]. These generalized methods for nonsmooth optimization are known to have poor convergence properties for nonconvex structures [47]. Smoothing techniques for nonsmooth functions encompass a variety of techniques, seeking to ensure sufficient smoothness for gradient-based methods [64]. Many of these methods are, however, designed for optimizing a nonsmooth function on a convex set, e.g [7,38]. Meanwhile, smoothing techniques for discontinuities by means of step-function approximations (e.g. [64]) are known to be prone to numerical instabilities, particularly for increasing accuracies of the discontinuity [8,60]. Exploitation of MIP for solving PWP optimization problems beyond complete approximative linearization [37] and direct solution as a nonconvex mixed integer nonlinear programming (MINLP) problem appears to be limited.

We adopt disjunctive representations of PWP constraints, drawing upon the extensive work on disjunctive programming (DP) formulations and representation of piecewise linear (PWL) functions [1,39,50,54]. Modeling piecewise functions as disjunctions enables application of MIP techniques, or specialized branch-and-bound or branch-and-cut schemes with a set condition for representing the piecewise constraints [2,28,39]. While adopting MIP techniques and formulations for solving PWP-constrained optimization problems facilitates exploitation of advancements in global optimization solvers [35,36,57], careful constraint formulations are required to overcome the inherent problem complexity. To this end, polynomial spline formulations [49] such as the B-spline is an attractive approach. Polynomial splines are constructed from overlapping (piecewise) polynomials with local support, and embodies a versatile set of techniques for modeling PWPs with favorable smoothness and numerical properties. For decades, polynomial splines, which we simply refer to as *splines* in this paper, have played an important role in function approximation and geometric modeling. In particular, they have been popular as nonlinear basis functions in regression problems [12,20], for example in kernel methods [22,63], and in finite element methods [23]. Yet, few references [5,15] apply splines within mathematical programming beyond the optimization of spline design parameters [45,59], trajectory optimization [18,34,43], and optimization of piecewise linear splines [33].

The availability of spline-compatible optimization algorithms and codes is limited. In a recent work, [16] develop a spatial branch-and-bound (sBB) algorithm for global optimization of spline-constrained problems. While the algorithm was shown to be highly efficient, it has only support for a limited set of algebraic functions, is only available as a specialized code and requires software for spline generation [17]. To address the comparably high modeling and implementation effort required for using the specialized sBB algorithm of [14,16] proposed an explicit constraint-formulation for continuous splines, yielding an ad-hoc mixed-integer quadratically constrained programming (MIQCP) model. In this paper, we build upon and significantly extend [14,16] to construct a general-purpose framework for mathematical programming of piecewise polynomial constraints, subsuming spline constraints. The framework is based on an epigraph formulation and we show how it accommodates lower semi-continuous PWPs given in the monomial, Bernstein or B-spline basis. The extension to lower semi-continuous PWPs has not been explicitly covered in previous works. However, the epigraph formulation in [14] can be applied to lower semi-continuous PWPs written as B-splines.

The main advancement of our work from previous works is our representation of PWPs as a disjunction of polynomial pieces. This allows us to exploit the fact that all the polynomial pieces can be written as a linear combination of a *single* multivariate polynomial basis. This leads to formulations that are minimal in the number of nonlinear (non-convex) constraints. Furthermore, we exploit properties of the polynomial bases and the grid structure for bound tightening and derivation of Bernstein cuts. Exact reformulations of the DP models yield MINLP formulations, which we benchmark and compare with existing solution methods.

The remainder of the paper is organized as follows. In Sects. 2 and 3, we present background theory of Bernstein polynomials, piecewise polynomials and polynomial splines. In Sect. 4, we present DP formulations of PWPs which we in Sect. 5 reformulate to MINLP models. In Sect. 6, we present computational results of the proposed formulations, comparing the results with existing methods for optimizing PWP functions. Concluding remarks in Sect. 7 ends the paper.

2 Background on polynomial bases

This section provides background material on polynomial functions to cover the theory needed for developing an optimization framework for piecewise polynomial functions. The theory is presented as a series of propositions that summarize some classical results for polynomials; cf. [13,31,42]. For brevity, most propositions are given without rigorous proofs; each proposition may, however, be proved by simple algebraic manipulations. To further simplify the disposition, we have put some computational details in "Appendix A".

We begin by introducing the monomial and Bernstein basis for polynomials in one variable. Several propositions are provided that ultimately enable computation of lower and upper bounds on any polynomial. These results are then extended to the multivariate case.

2.1 Univariate polynomials and the Bernstein basis

Let $\mathbb{P}_p$ denote the vector space of polynomial functions in one real variable with degree less than or equal to $p \in \mathbb{N}$, i.e.

$$\mathbb{P}_p = \operatorname{span}\{M_i\}_{i=0}^{p}, \quad M_i : \mathbb{R} \to \mathbb{R}, \quad M_i(x) = x^i, \quad 0 \le i \le p. \tag{1}$$

The set $\{M_i\}_{i=0}^{p}$ is commonly referred to as the *monomial basis* or *power basis* of $\mathbb{P}_p$. Any polynomial $f \in \mathbb{P}_p$ can be written as

$$f(x) = \sum_{i=0}^{p} a_i M_i(x) = \boldsymbol{a}^\top \boldsymbol{M}_p(x), \tag{2}$$

where the vector of coefficients $\boldsymbol{a} = [a_i]_{i=0}^{p} \in \mathbb{R}^{p+1}$, and the vector of monomial basis functions $\boldsymbol{M}_p(x) = [M_i(x)]_{i=0}^{p} \in \mathbb{R}^{p+1}$.

An alternative basis for $\mathbb{P}_p$ is the *Bernstein basis*

$$B_{i,p} = \binom{p}{i} x^i (1-x)^{p-i}, \quad 0 \le i \le p. \tag{3}$$

Since $\operatorname{span}\{B_{i,p}\}_{i=0}^{p} = \mathbb{P}_p$, any polynomial $f \in \mathbb{P}_p$ may be expressed in the Bernstein basis as

$$f(x) = \sum_{i=0}^{p} c_i B_{i,p}(x) = \boldsymbol{c}^\top \boldsymbol{B}_p(x), \tag{4}$$

where the vector of coefficients $\boldsymbol{c} = [c_i]_{i=0}^{p} \in \mathbb{R}^{p+1}$, and the vector of Bernstein basis functions $\boldsymbol{B}_p(x) = [B_{i,p}(x)]_{i=0}^{p} \in \mathbb{R}^{p+1}$.

The monomial and Bernstein basis are related via the linear mapping $\boldsymbol{M}_p = Q_p \boldsymbol{B}_p$, where $Q_p \in \mathbb{R}^{(p+1)\times(p+1)}$ is the *transformation matrix* given in "Appendix A.1". Consequently, $\boldsymbol{a}^\top \boldsymbol{M}_p(x) = \boldsymbol{c}^\top \boldsymbol{B}_p(x)$, given that $\boldsymbol{c} = Q_p^\top \boldsymbol{a}$.

The Bernstein polynomials possess several useful properties that facilitate the study of signs and bounds on polynomial functions. These properties, to be presented next, will later be utilized to devise bounds on PWPs.

Lemma 1 (Convex combination property of Bernstein polynomials) *The following holds true for a set of degree p Bernstein polynomials $\{B_{i,p}\}_{i=0}^{p}$:*

$$\begin{aligned} &B_{i,p}(x) \ge 0, \quad \forall x \in [0,1], \quad i = 0, \dots, p \\ &\sum_{i=0}^{p} B_{i,p}(x) = 1, \quad \forall x \in \mathbb{R} \end{aligned} \tag{5}$$

Proof The lemma is proved by applying Newton's binomial identity to (3). □

Proposition 1 (Bounds on Bernstein polynomials) *Let $f \in \mathbb{P}_p$ be a polynomial expressed in the Bernstein basis (4), and denote $c^L = \min\{c_i\}_{i=0}^{p}$ and $c^U = \max\{c_i\}_{i=0}^{p}$. Then, a valid bound on f is $c^L \le f(x) \le c^U$ $\forall x \in [0,1]$.*

Proof From Lemma 1 it follows that

$$c^L = \sum_{i=0}^{p} c^L B_{i,p}(x) \le \underbrace{\sum_{i=0}^{p} c_i B_{i,p}(x)}_{f(x)} \le \sum_{i=0}^{p} c^U B_{i,p}(x) = c^U.$$

□

Observe that Proposition 1 holds for $x \in [0, 1]$. To obtain a bounding box on a general domain $[x^L, x^U]$, we perform an affine change of variable.

Proposition 2 (Reparametrization of polynomial) *Let $f \in \mathbb{P}_p$ be a polynomial $f(x) = \boldsymbol{a}^T \boldsymbol{M}_p(x)$, for $x \in [x^L, x^U]$. Consider the affine change of variables $x = (x^U - x^L)u + x^L$, with $u \in [0, 1]$. The polynomial f can be reparametrized from x to u via the linear mapping $\boldsymbol{M}_p(x) = R_p \boldsymbol{M}_p(u)$, where $R_p \in \mathbb{R}^{(p+1)\times(p+1)}$ is the reparametrization matrix given in "Appendix A.2".*

Proof Cf. [42]. □

By combining Propositions 1 and 2, we obtain a bound for polynomials on a general domain $x \in [x^L, x^U]$.

Proposition 3 (Polynomial bounds on general intervals) *Let $\boldsymbol{a}$ be the coefficients of a polynomial $f \in \mathbb{P}_p$ in the monomial basis. Furthermore, let $\boldsymbol{c} = (R_p Q_p)^T \boldsymbol{a}$ be the coefficients obtained by first reparametrizing the monomial basis from $x \in [x^L, x^U]$ to $u \in [0, 1]$, and then transforming the monomial basis to the Bernstein basis in u. Then,*

$$c^L \le f(x) \le c^U \quad \forall x \in [x^L, x^U], \tag{6}$$

where $c^L = \min\{c_i\}_{i=0}^p$ and $c^U = \max\{c_i\}_{i=0}^p$.

Proof The result follows directly from Propositions 1 and 2. □

2.2 Multivariate polynomials

A vector space of multivariate polynomials $f : \mathbb{R}^d \to \mathbb{R}$ in the variables $\boldsymbol{x} = (x_1, \dots, x_d) \in \mathbb{R}^d$, can be constructed by taking the tensor product of univariate polynomial bases. Specifically, we construct a multivariate polynomial basis as

$$\boldsymbol{M}_p^d(\boldsymbol{x}) = \bigotimes_{j=1}^{d} \boldsymbol{M}_p(x_j), \tag{7}$$

where $\boldsymbol{M}_p(x_j) = [M_i(x_j)]_{i=0}^p$ is a vector of $p+1$ monomial basis functions in the variable x_j.

In (7), $\boldsymbol{M}_p^d$ is a vector of $n = (p+1)^d$ polynomials of degree less than or equal to dp in d variables: i.e. each multivariate basis function results from d products of univariate polynomial basis functions of degree less than or equal to p. The basis spans a (tensor product) vector space of multivariate polynomials, denoted $\mathbb{P}_p^d = \text{span}\{M_{i,p}^d\}_{i=0}^{n-1}$ with $\dim(\mathbb{P}_p^d) = (p+1)^d$. The exponential growth in the number of basis functions with the number of variables d, is a phenomenon often referred to as the curse of dimensionality [20]. This phenomenon limits most practical applications of tensor product bases to 5-6 variables.

For notational brevity, we assume in the above construction that the polynomial basis and degree are equal for all variables. These assumptions can be removed without loss of generality as the multivariate basis may be constructed from any combination of univariate polynomial bases of varying degrees. Subsequently, we consider also the multivariate Bernstein basis for $\mathbb{P}_p^d$, which we denote by $\boldsymbol{B}_p^d$.

Using the multivariate polynomial basis, we may express any $f \in \mathbb{P}_p^d$ as

$$f(\boldsymbol{x}) = \sum_{i=0}^{n-1} a_i M_{i,p}^d(\boldsymbol{x}) = \boldsymbol{a}^\top \boldsymbol{M}_p^d(\boldsymbol{x}). \tag{8}$$

The important property of the bounding box in Proposition (1) naturally extends to the multivariate case.

Proposition 4 (Multivariate polynomial bounds on the unit cube) *Let $f \in \mathbb{P}_p^d$ be a polynomial expressed in the multivariate Bernstein basis*

$$f(\boldsymbol{x}) = \boldsymbol{c}^T \boldsymbol{B}_p^d(\boldsymbol{x}) = \boldsymbol{c}^T \bigotimes_{j=1}^{d} \boldsymbol{B}_p(x_j). \tag{9}$$

Then, $c^L \leq f(\boldsymbol{x}) \leq c^U\ \forall \boldsymbol{x} \in [0,1]^d$, where $c^L = \min\{c_i\}_{i=0}^{n-1}$ and $c^U = \max\{c_i\}_{i=0}^{n-1}$.

Proof For any $1 \leq r \leq d$, let

$$\boldsymbol{B}_p^{(d,-r)}(\boldsymbol{x}) = \bigotimes_{j=1, j\neq r}^{d} \boldsymbol{B}_p(x_j). \tag{10}$$

Then, given $n = m^d = (p+1)^d$, it follows from Lemma 1 that

$$\begin{aligned} \sum_{i=0}^{m^d-1} B_{i,p}^d(\boldsymbol{x}) &= \left(\sum_{i=0}^{p} B_{i,p}(x_r)\right)\left(\sum_{i=0}^{m^{(d-1)}-1} B_{i,p}^{(d,-r)}(\boldsymbol{x})\right) \\ &= \sum_{i=0}^{m^{(d-1)}-1} B_{i,p}^{(d,-r)}(\boldsymbol{x}). \end{aligned} \tag{11}$$

The above relation implies that

$$\sum_{i=0}^{n-1} B_{i,p}^d(\boldsymbol{x}) = 1. \tag{12}$$

The identity in (12), combined with Lemma 1, ensures that for all $\boldsymbol{x} \in [0,1]^d$

$$c^L \leq c^L \sum_{i=0}^{n-1} B_{i,p}^d(\boldsymbol{x}) \leq \underbrace{\sum_{i=0}^{n-1} c_i B_{i,p}^d(\boldsymbol{x})}_{f(\boldsymbol{x})} \leq c^U \sum_{i=0}^{n-1} B_{i,p}^d(\boldsymbol{x}) \leq c^U, \tag{13}$$

which proves the proposition. □

Proposition 4 provides bounds on a polynomial expressed in the multivariate Bernstein basis for $\boldsymbol{x}$ constrained to the unit cube $[0,1]^d$. Analogous to the univariate case, we obtain bounds for general domains by reparametrizing the basis.

Proposition 5 (Multivariate polynomial bounds on general domains) *Let $f \in \mathbb{P}_p^d$ be a polynomial expressed in the multivariate monomial basis*

$$f(\boldsymbol{x}) = \boldsymbol{a}^T \bigotimes_{j=1}^{d} \boldsymbol{M}_p(x_j), \tag{14}$$

for $\boldsymbol{x} \in [x_1^L, x_1^U] \times \cdots \times [x_d^L, x_d^U] = [\boldsymbol{x}^L, \boldsymbol{x}^U]$. For each variable x_j, let $R_{p,j}$ be the reparametrization matrix that reparametrizes the monomial basis from $x_j \in [x_j^L, x_j^U]$ to $u_j \in [0,1]$, computed according to (33) in "Appendix A.2". Furthermore, let Q_p be the transformation matrix computed as in (32) in "Appendix A.1". Then, f may be mapped to the multivariate Bernstein basis as follows:

$$f(\boldsymbol{u}) = \boldsymbol{a}^T \bigotimes_{j=1}^{d} R_{p,j} Q_p \boldsymbol{B}_p(u_j) = \boldsymbol{c}^T \bigotimes_{j=1}^{d} \boldsymbol{B}_p(u_j) \tag{15}$$

where

$$\boldsymbol{c}^T = \boldsymbol{a}^T \bigotimes_{j=1}^{d} R_{p,j} Q_p. \tag{16}$$

Finally, we may apply Proposition 4 to obtain the bounds

$$c^L \leq f(\boldsymbol{x}) \leq c^U \quad \forall \boldsymbol{x} \in [\boldsymbol{x}^L, \boldsymbol{x}^U], \tag{17}$$

where $c^L = \min\{c_i\}_{i=0}^{n-1}$ and $c^U = \max\{c_i\}_{i=0}^{n-1}$.

Proof The result follows directly from Proposition 4. □

3 Piecewise polynomial functions

In this section, we describe the piecewise polynomial functions (PWPs) to which we first give a formal definition for the *continuous* case. We then depart from the continuity requirement in order to consider the more general case of *lower semi-continuous* PWPs. The definitions given below provide a framework for the development of the mathematical programming formulations in Sects. 4 and 5. The definitions are not novel, but help us highlight well-established connections between PWPs and splines; cf. [42,49].

Definition 1 (*Continuous piecewise polynomial function*) Let $D \in \mathbb{R}^d$ be a compact set. A function $f : D \subset \mathbb{R}^d \to \mathbb{R}$ is a *continuous piecewise polynomial* if and only if there exists a finite family of polytopes Π such that $D = \bigcup_{P \in \Pi} P$ and

$$f(\boldsymbol{x}) = \{f_P(\boldsymbol{x}),\ \ \boldsymbol{x} \in P,\ \ \forall P \in \Pi\ , \tag{18}$$

where $f_P : P \subset \mathbb{R}^d \to \mathbb{R}$ and $f_P \in \mathbb{P}_p^d$ for all $P \in \Pi$, and some degree $p \in \mathbb{N}_0$.

Note that the domain D does not need to be connected or convex. Continuity of f on D is ensured since $f_{P_1}(\boldsymbol{x}) = f_{P_2}(\boldsymbol{x})$ if $\boldsymbol{x} \in P_1 \cap P_2$ for two adjacent polytopes $P_1, P_2 \in \Pi$.

In the above definition, the polynomial pieces $\{f_P\}_{P \in \Pi}$ are constructed from some basis that spans the space $\mathbb{P}_p^d$. A special case occurs when $d = 1$ and $p = 1$, for which $\{f_P\}_{P \in \Pi}$ are linear functions in one variable, and f a continuous piecewise linear function. Furthermore, for $d > 1$ and $p = 1$, f is a continuous piecewise multilinear function due to the tensor product construction of $\mathbb{P}_p^d$. In general, the polynomial pieces are of degree less than or equal to $dp \in \mathbb{N}_0$.

3.1 Polytopes on a rectilinear grid

Definition 1 of continuous PWPs does not prescribe the polytopes; they may for instance be given as the convex hull of a finite number of points, or as a system of linear inequalities. Some formulations for piecewise linear functions require the polytopes to be simplices resulting from a triangulation of D [55]. For most practical applications of PWPs, however, the polytopes are assumed to be n-orthotopes (hyperrectangles/boxes) arranged on an axis-aligned rectilinear grid.[1] In the rest of this paper, we will assume that the domain D is partitioned on such a rectilinear grid, for which the polytopes in Π are characterized as follows.

For $i \in \{1, \ldots, d\}$, let $\pi^i = \{\pi_0^i, \ldots, \pi_{m_i}^i\} \in \mathbb{R}$ denote a strictly monotonically increasing sequence of m_i real numbers, e.g. $\pi_0^i < \pi_1^i < \cdots < \pi_{m_i}^i$. To index the polytopes we introduce the set $K := \{(k_1, \ldots, k_d) : k_i \in \{1, \ldots, m_i\}, \forall i = 1, \ldots, d\}$. This lets us define the rectilinear grid $G := \{P_k : k \in K\}$, consisting of $|G| = m_1 \cdots m_d$ boxes given by

$$P_k = \{\boldsymbol{x} \in \mathbb{R}^d : \pi_{k_i - 1}^i \leq x_i \leq \pi_{k_i}^i, \ \forall i \in \{1, \ldots, d\}\}. \tag{19}$$

The grid G is in compliance with Definition 1, since it is a family n-orthotopes P_k, which are bounded polytopes. Subsequently, we consider PWPs partitioned on a rectilinear grid and write $\Pi = G$. We will also denote with $\mathbb{P}_p^d(G)$ the space of piecewise degree p polynomials with a partition of the domain D given by the rectilinear grid G.

3.2 The epigraph of piecewise polynomials

A continuous PWP $f : D \subset \mathbb{R}^n \to \mathbb{R}$ may be modeled by its epigraph $\operatorname{epi}(f) := \{(\boldsymbol{x}, z) \in D \times \mathbb{R} : f(\boldsymbol{x}) \leq z\}$. We assume that D is a bounded domain and that f participates in a constraint of the form $f(\boldsymbol{x}) \leq 0$ or in an objective function to be minimized. That is, the constraint $f(\boldsymbol{x}) \leq 0$ can be modeled as $(\boldsymbol{x}, z) \in \operatorname{epi}(f)$, $z \leq 0$, and the objective f can be modeled as the minimization of z subject to $(\boldsymbol{x}, z) \in \operatorname{epi}(f)$.

The epigraph of f can be expressed as the union of epigraphs of its pieces f_P, i.e.

$$\operatorname{epi}(f) = \bigcup_{P \in \Pi} \operatorname{epi}(f_P), \tag{20}$$

where $\operatorname{epi}(f_P) := \{(\boldsymbol{x}, z) \in P \times \mathbb{R} : f_P(\boldsymbol{x}) \leq z\}$. As illustrated by Fig. 1, the epigraph of a piecewise function is in general a nonconvex set. Note that $\operatorname{epi}(f_P)$ is convex if and only if f_P is convex on P. Furthermore, f may be nonconvex, even if f_P (and $\operatorname{epi}(f_P)$) is convex for all $P \in \Pi$.

The theory developed in [26,27] shows that $\operatorname{epi}(f)$ can be modeled as a MILP if and only if f is piecewise linear and lower semi-continuous. Based on this theory, Vielma et al. [55,56] derived new MILP models and presented a unifying framework for piecewise linear functions. To follow on these works, we continue by extending Definition 1 to handle lower semi-continuous piecewise polynomials.

[1] To understand why a rectilinear grid is practical, consider a domain partitioned into a set of non-regular polytopes (resembling a shattered window). Patching together higher order polynomials on these polytopes in order to ensure continuity on all faces is non-trivial.

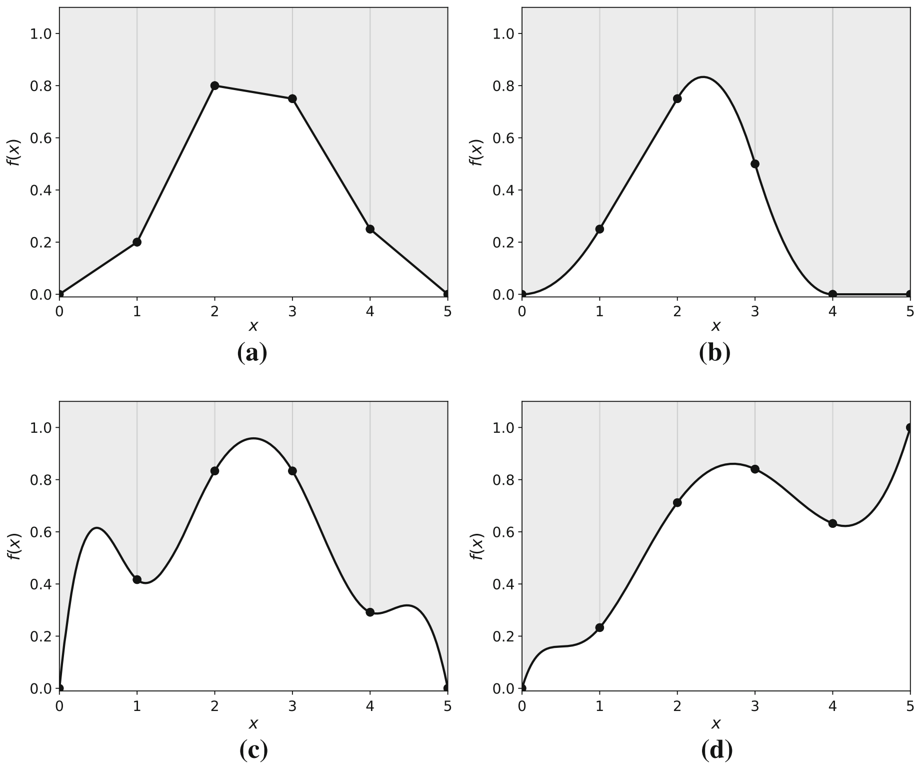

Fig. 1 Continuous piecewise polynomials and their epigraph (colored grey). The five pieces of the piecewise polynomial are linear in **a**, quadratic in **b**, cubic in **c**, and quartic in **d**

3.3 Extension to lower semi-continuous piecewise polynomials

Below we provide a definition of PWPs that are not necessarily continuous. This allows us to analyze and integrate in the framework the subset of discontinuous PWPs that are *lower semi-continuous*. The extended definition lets us tie our PWP framework to the B-spline modeling framework, which facilitates construction of PWPs with predefined smoothness.

Before extending Definition 1 of PWPs, we consider the property of lower semi-continuity with some simple examples. Formally, a function f is lower semi-continuous if for any $\boldsymbol{x}_0 \in D$

$$\liminf_{\boldsymbol{x} \to \boldsymbol{x}_0} f(\boldsymbol{x}) \geq f(\boldsymbol{x}_0). \tag{21}$$

The importance of lower semi-continuity comes from the fact that the epigraph of a function is closed if and only if it is lower semi-continuous. It is hence a requirement for the epigraph model in Sect. 3.2. Since a continuous function is lower semi-continuous, the requirement always holds for continuous PWPs. To illustrate this property, consider the two PWPs in Fig. 2. The PWP in Fig. 2a is lower semi-continuous at $x = 2$, but not at points $x = 1$ and $x = 3$. Thus, it is not a lower semi-continuous PWP and its epigraph is not closed. On the other hand, the PWP in Fig. 2b has one discontinuity ($x = 2$) at which it is lower semi-continuous, and its epigraph is closed.

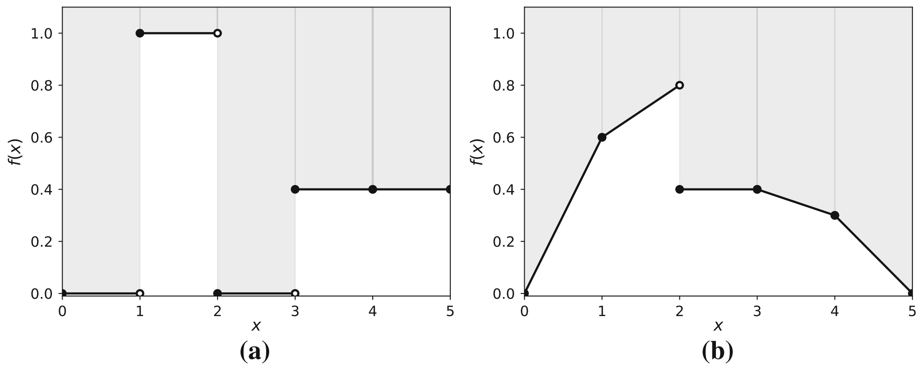

Fig. 2 Two discontinuous PWPs and their epigraph (colored grey). Both PWPs consist of five pieces defined on the intervals [0, 1), [1, 2), [2, 3), [3, 4), and [4, 5]; the open ends are marked with white-filled circles. The pieces are constant in **a** and linear in **b**

To allow discontinuities, [55] employed a characterization of the domain using *copolytopes* (sets defined by a finite set of strict and non-strict linear inequalities). Similarly, we use copolytopes to define not necessarily continuous PWPs as follows.

Definition 2 (*Piecewise polynomial function*) Let $D \in \mathbb{R}^d$ be a compact set. A (not necessarily continuous) function $f : D \subset \mathbb{R}^d \to \mathbb{R}$ is piecewise polynomial if and only if there exists a finite family of copolytopes Π such that $D = \bigcup_{P \in \Pi} P$ and

$$f(\boldsymbol{x}) = \{f_P(\boldsymbol{x}), \;\; \boldsymbol{x} \in P, \;\; \forall P \in \Pi \;, \tag{22}$$

where $f_P : P \subset \mathbb{R}^d \to \mathbb{R}$ and $f_P \in \mathbb{P}_p^d$ for all $P \in \Pi$, and some degree $p \in \mathbb{N}_0$.

With a minor adjustment to the continuous case, we may express the epigraph of a function f defined according to Definition 2, as the union of epigraphs of its pieces f_P. That is, we model

$$\text{epi}(f) = \bigcup_{P \in \Pi} \text{epi}(f_P), \tag{23}$$

where we now use $\text{epi}(f_P) := \left\{(\boldsymbol{x}, z) \in \bar{P} \times \mathbb{R} : f_P(\boldsymbol{x}) \leq z\right\}$ in which $\bar{P}$ is the closure of P (note that since $f_P \in \mathbb{P}_p^d$ we may evaluate it at the right boundary of $\bar{P}$). Recall that epi(f) is closed if and only if f is lower semi-continuous.

Similar to continuous PWPs, we consider a partition of the domain on a rectilinear grid G. However, we now compose the grid of *left half-closed* boxes:

$$P_k = \{\boldsymbol{x} \in \mathbb{R}^d : \pi^i_{k_i - 1} \leq x_i < \pi^i_{k_i}, \forall i \in \{1, \dots, d\}\}. \tag{24}$$

Note that P_k, being a left half-closed box, is a special type of copolytope.

A technicality arises with this partitioning in that the rightmost boundaries of D are open, and hence D is open, which breaks compatibility with Definition 2. To close these boundaries we must require that the rightmost boundaries of the rightmost boxes are closed; i.e. in (24) we must replace $\pi^i_{k_i - 1} \leq x_i < \pi^i_{k_i}$ with $\pi^i_{k_i - 1} \leq x_i \leq \pi^i_{k_i}$ if $k_i = m_i$. The addition of this requirement on the partitioning ensures that the domain is closed and hence compatible with Definition 2.

3.4 Relation to splines

With piecewise linear functions one is often concerned with $\mathcal{C}^0$ continuity at intersections $\boldsymbol{x} \in P_1 \cap P_2$, where $P_1, P_2 \in \Pi$. For PWPs in general, $\mathcal{C}^1$, $\mathcal{C}^2$, or higher-order continuity at the intersections is an obtainable and often desired property. Definition 1 only guarantees $\mathcal{C}^0$ continuity, but does not exclude PWPs with higher order continuity.

Splines are PWPs constructed with continuity constraints. The most widely used framework for polynomial splines is the B-spline, in which PWPs with a desired degree of smoothness can be constructed by allowing basis functions to overlap. Consider the multivariate B-spline

$$f(\boldsymbol{x}) = \boldsymbol{c}^\mathsf{T} \bigotimes_{j=1}^{d} \boldsymbol{N}_p(x_j; \boldsymbol{t}_j) = \boldsymbol{c}^\mathsf{T} \boldsymbol{N}_p^d(\boldsymbol{x}; T), \tag{25}$$

where $\boldsymbol{N}_p(x_j; \boldsymbol{t}_j) = [N_{i,p}(x_j; \boldsymbol{t}_j)]_{i=0}^{n_j-1}$ is a vector of n_j degree p B-spline basis functions in the variable x_j, $\boldsymbol{c}$ are coefficients, and $\boldsymbol{t}_j$ is a non-decreasing sequence of reals called a *knot sequence*. The B-spline basis functions in a variable x_j, $\boldsymbol{N}_p(x_j; \boldsymbol{t}_j)$, are polynomials with local support that are spliced together at the points specified by the knot sequence $\boldsymbol{t}_j$.

The multivariate B-spline basis is denoted $\boldsymbol{N}_p^d(\boldsymbol{x}; T)$, where $T = \{\boldsymbol{t}_j\}_{j=1}^d$ is the set of knot sequences that parametrize the partition of the domain D. The vector space of B-splines spanned by the above basis is denoted $\mathbb{S}_p^d(T) = \mathbb{S}_p(\boldsymbol{t}_1) \times \cdots \times \mathbb{S}_p(\boldsymbol{t}_d)$, where $\mathbb{S}_p(\boldsymbol{t}_j) = \operatorname{span}\{N_{i,p}(x_j; \boldsymbol{t}_j)\}_{i=0}^{n_j-1}$.

The B-spline basis can viewed as an extension of the Bernstein basis, generalizing the description of a single polynomial on a continuous interval to piecewise polynomials over a partitioned domain, specified by the knot sequence [13]. It shares the non-negativity and partition-of-unity properties of the Bernstein basis, and is invariant to an affine change of variables. The close relationship is made clear by the Proposition 6 in "Appendix B", which shows that Bernstein and B-spline bases are equivalent for a certain knot sequence.

Multivariate B-splines are defined on a rectilinear grid of left half-closed boxes aligned to the variable axes, due to its construction by the Kronecker product. The partitioning is thus the same as for the piecewise polynomials in Definition 2. The well-known equality $\mathbb{S}_p^d(T) = \mathbb{P}_p^d(G)$, where G is a rectilinear grid, implies that any spline can be accommodated by the framework in Definition 2, given appropriate knot sequences T. This equality was first stated for the univariate case in the Curry-Schoenberg theorem [9]. In Lemma 2, "Appendix B", we restate this relationship for the multivariate case using our notational framework. Since most PWPs are built as splines, e.g. using cubic spline interpolation or smoothing splines, this relationship holds a practical value and we will later use it to generate PWPs for the numerical study.

4 Disjunctive formulations for piecewise polynomial functions

In this section, we use disjunctions as a means of representing PWP constraints. Consider a piecewise polynomial $f : D \to \mathbb{R}$, defined as in Definition 1 with a rectangular domain $D = \{\boldsymbol{x} : \boldsymbol{x}^L \leq \boldsymbol{x} \leq \boldsymbol{x}^U\} = \bigcup_{P \in \Pi} P \subset \mathbb{R}^d$. The epigraph of f, $\operatorname{epi}(f) = \{(\boldsymbol{x}, z) \in D \times \mathbb{R} : f(\boldsymbol{x}) \leq z\}$, can be represented by the disjunction [27]:

$$\bigvee_{P\in\Pi}\begin{bmatrix} Y_P \\ \boldsymbol{x}\in P \\ f_P(\boldsymbol{x})\le z \end{bmatrix}, \quad \veebar_{P\in\Pi} Y_P, \tag{DP-1}$$

where we have associated a boolean variable $Y_P \in \{\text{True}, \text{False}\}$ with each polytope $P \in \Pi$. Each disjunctive term is related to a polytope P, so that $\boldsymbol{x} \in P$ and z is restricted to the epigraph epi(f_P) of a piece f_P of f. The disjunction thus models epi(f) as the union of epigraphs in (20). The exclusive OR operator on the boolean variables ensures that exactly one polytope $P \in \Pi$ is selected.

DP-1 is also a valid model for lower semi-continuous PWPs according to Definition 2, if we replace in each term the domain constraint with $\boldsymbol{x} \in \bar{P}$, where $\bar{P} = \mathbf{cl}(P)$. In both cases, DP-1 is a proper disjunction [1,53] in the sense that no single polytope covers the entire feasible region. Observe that Definition 1 ensures continuity in the overlap between the polytopes constituting the disjunction. In the derivations that follow we assume that f is a continuous PWP, but remark that the resulting formulations are also valid for lower semi-continuous PWPs subject to the mentioned domain-substitution.

The disjunction DP-1 contains a nonlinear, possibly nonconvex inequality for each term, thereby severely impeding the scalability of the formulation and hence its practical application. As a partial remedy, we may utilize that each polynomial f_P can be expressed as a linear combination of the basis functions spanning $\mathbb{P}_p^d$, that is, $f_P \in \mathbb{P}_p^d$, for all $P \in \Pi$. This salient characteristic allows us to exploit that the basis functions are independent of P, and hence can be extracted outside the disjunction as a common set of nonlinearities. The reformulation simplifies DP-1 to

$$\bigvee_{P\in\Pi}\begin{bmatrix} Y_P \\ \boldsymbol{x}\in P \\ \boldsymbol{a}_P^{\mathsf{T}}\boldsymbol{\beta}\le z \end{bmatrix}, \quad \veebar_{P\in\Pi} Y_P, \quad \boldsymbol{\beta} = \boldsymbol{M}_p^d(\boldsymbol{x}), \tag{DP-2}$$

where the polynomial piece f_P is expressed as a linear combination of the $n = \dim(\mathbb{P}_p^d)$ multivariate monomial basis functions $\boldsymbol{\beta} = \boldsymbol{M}_p^d$. We stress that even though it is always possible to substitute nonlinearities with new variables to obtain linear disjunctions as in DP-2, the benefit comes solely from having common basis functions and hence reducing the number of nonlinear constraints.

Generally, DP-2 requires $n = \dim(\mathbb{P}_p^d) = (p+1)^d$ polynomial constraints to model a PWP, invariant to the discretization of the domain. Consider a PWP defined on a rectilinear grid of $|\Pi| = m^d$ boxes, resulting from a discretization with $m \ge 1$ intervals in each of the d variables. DP-1 requires m^d polynomial constraints to model this PWP. Thus, when $m > p+1$, then $m^d > (p+1)^d$, and the formulation in DP-2 is likely preferable to DP-1. To summarize the above argument: modeling the polynomial function space $\mathbb{P}_p^d$ via its n basis functions, as opposed to modeling each of the $|\Pi|$ polynomial pieces separately, will generally result in fewer nonlinear constraints. Still, the exponential increase with d in the number of nonlinear constraints puts a practical limit on formulations derived from either DP-1 or DP-2.

Remark 1 Piecewise McCormick envelopes and other linear relaxations (e.g. [19]) for bilinear terms $f_P(x) = x_1x_2$, with $(x_1, x_2) \in P$, can be derived from DP-1. Such relaxations of DP-1 render linear approximations, whereas DP-2 is an exact formulation.

4.1 Bounds on polynomial constraints

The efficiency of numerical methods for solving DP-2 relies strongly on the ability to derive strong upper bounds on the constraints $f_P = \boldsymbol{a}_P^\top \boldsymbol{\beta} \le z$, for $P \in \Pi$. Suppose that $\boldsymbol{a}_P^\top \boldsymbol{\beta} \in [m_P^L, m_P^U]$ and that $z^L \le z$ for any $\boldsymbol{x} \in D$. We may then define $M_P^U := m_P^U - z^L$ so that, for any $\boldsymbol{x} \in D$,

$$\boldsymbol{a}_P^\top \boldsymbol{\beta} - z \le M_P^U. \tag{26}$$

To obtain a valid upper bound M_P^U, we must determine the values of m_P^U and z^L. We observe that any feasible solution must satisfy $\boldsymbol{a}_P^\top \boldsymbol{\beta} \le z$ for some $P \in \Pi$. Thus, a valid lower bound on z is $z \ge z^L = \min\{m_P^L\}_{P\in\Pi}$, and we may rewrite the upper bound on the polynomial constraint to

$$M_P^U = m_P^U - \min\{m_P^L\}_{P\in\Pi}. \tag{27}$$

It is then obvious that computing M_P^U for each $P \in \Pi$ requires a lower and upper bound on all polynomials $\{f_P\}_{P\in\Pi}$.

Returning to DP-2, there is a subtlety due to the substitution of the nonlinearities that impedes the derivation of tight bounds on the polynomials. The issue is that the bounds on $f_P(\boldsymbol{x}) = \boldsymbol{a}_P^\top \boldsymbol{\beta} \in [m_P^L, m_P^U]$ must be valid for all $\boldsymbol{x} \in [\boldsymbol{x}^L, \boldsymbol{x}^U]$, not only for $\boldsymbol{x} \in P$. This poses numerical problems since the piecewise polynomials may become prohibitively large on $[\boldsymbol{x}^L, \boldsymbol{x}^U]$, resulting in undesirably large bounds.

To solve this issue and obtain tighter bounds on the polynomials $\{f_P\}_{P\in\Pi}$, we utilize the procedure in Proposition 5 to perform a reparametrization before computing the polynomial bounds. Let $\boldsymbol{u} \in [\boldsymbol{0}, \boldsymbol{1}] \in \mathbb{R}^d$ and $f_P = \boldsymbol{a}_P^\top \boldsymbol{M}_p^d(\boldsymbol{x}) = \boldsymbol{c}_P^\top \boldsymbol{B}_p^d(\boldsymbol{u})$, where the coefficients of the reparametrized polynomial in Bernstein form are given as

$$\boldsymbol{c}_P^\top = \boldsymbol{a}_P^\top \bigotimes_{j=1}^{d} R_{p,j} Q_p.$$

Using the multivariate Bernstein basis $\boldsymbol{B}_p^d(\boldsymbol{u})$ for $\boldsymbol{0} \le \boldsymbol{u} \le \boldsymbol{1}$ enables reformulation of DP-2 to the disjunction

$$\begin{aligned}
&\bigvee_{P_k\in\Pi} \begin{bmatrix} Y_P \\ x_i = (\pi_{k_i}^i - \pi_{k_i-1}^i)u_i + \pi_{k_i-1}^i, \ \forall i \in \{1,\dots,d\} \\ \boldsymbol{c}_P^\top \boldsymbol{\beta} \le z \end{bmatrix} \\
&\underline{\vee}_{P\in\Pi} \ Y_P \\
&\boldsymbol{\beta} = \boldsymbol{B}_p^d(\boldsymbol{u}) \\
&\boldsymbol{0} \le \boldsymbol{\beta} \le \boldsymbol{1} \\
&\boldsymbol{0} \le \boldsymbol{u} \le \boldsymbol{1}
\end{aligned} \tag{DP-3}$$

In DP-3, the prescription of the polytopes in (19) is enforced using $\boldsymbol{u}$. We have also bounded the Bernstein basis functions to be in [0, 1], in accordance with Lemma 1.

Within each term $P \in \Pi$ in DP-3, the variables $\boldsymbol{x}$ and $\boldsymbol{u}$ are linearly dependent. The reformulation DP-3 enables computation of bounds on the reparametrized polynomials. In particular, by invoking Proposition 4 we obtain $c_P^L \le \boldsymbol{c}_P^\top \boldsymbol{\beta} \le c_P^U$, where $c_P^L = \min\{c_{i,P}\}_{i=0}^{n-1}$ and $c_P^U = \max\{c_{i,P}\}_{i=0}^{n-1}$. This allows us to set

$$M_P^U = c_P^U - \min\{c_P^L\}_{P\in\Pi}, \tag{28}$$

and thereby obtain a valid upper bound $\boldsymbol{c}_P^\top \boldsymbol{\beta} - z \leq M_P^U$ for all $\boldsymbol{u} \in [\mathbf{0}, \mathbf{1}]$ and $P \in \Pi$.

While (28) provides an upper bound on $\boldsymbol{c}_P^\top \boldsymbol{\beta} - z$ for $P \in \Pi$, it is global in the sense that it involves all the polynomial pieces via the lower bound $\min\{c_P^L\}_{P \in \Pi}$ on z. "Local" bounds that concern only a single polynomial, can be obtained by introducing auxiliary variables $\{z_P\}_{P \in \Pi}$ and reformulating to

$$
\begin{aligned}
& \begin{bmatrix} Y_P \\ x_i = (\pi^i_{k_i} - \pi^i_{k_i-1}) u_i + \pi^i_{k_i-1}, \ \forall i \in \{1, \ldots, d\} \\ \boldsymbol{c}_P^\top \boldsymbol{\beta} \leq z_P \end{bmatrix} \vee \begin{bmatrix} \neg Y_P \\ z_P = 0 \end{bmatrix}, \quad \forall P_k \in \Pi \\
& \underset{P \in \Pi}{\veebar} \ Y_P \\
& \sum_{P \in \Pi} z_P \leq z \\
& \boldsymbol{\beta} = \boldsymbol{B}_p^d(\boldsymbol{u}) \\
& \mathbf{0} \leq \boldsymbol{\beta} \leq \mathbf{1} \\
& \mathbf{0} \leq \boldsymbol{u} \leq \mathbf{1}
\end{aligned}
\tag{DP-BASIC}
$$

With DP-BASIC, it is sufficient to derive an upper bound M_P^U so that $\boldsymbol{c}_P^\top \boldsymbol{\beta} - z_P \leq M_P^U$. A bound is readily obtained as

$$
M_P^U = c_P^U - \min\{0, c_P^L\}. \tag{29}
$$

Remark 2 The special case of an equality constraint $\boldsymbol{c}_P^\top \boldsymbol{\beta} = z_P$ can be handled by writing $0 \leq \boldsymbol{c}_P^\top \boldsymbol{\beta} - z_P \leq 0$. Using the same arguments as above we obtain the bounds $M_P^L \leq \boldsymbol{c}_P^\top \boldsymbol{\beta} - z_P \leq M_P^U$, where $M_P^L = c_P^L - \max\{0, c_P^U\}$.

4.2 Exploiting the rectilinear grid structure

In the preceding formulations, a disjunction of $|\Pi| = \prod_{i=1}^d m_i$ terms is used to enforce the box constraints $\boldsymbol{x} \in P_k$, where P_k is given as in (19). This seems unnecessary since the grid structure can be modeled by d disjunctions of m_i terms (for $i = 1, \ldots, d$), as shown next.

We introduce a boolean variable Y_j^i for each interval $\pi_{j-1}^i \leq x_i \leq \pi_j^i$, for $j = 1, \ldots, m_i$, and $i = 1, \ldots, d$. Then, by adding the conditions

$$
\underset{j=1,\ldots,m_i}{\veebar} Y_j^i, \quad \forall i \in \{1, \ldots, d\}, \tag{30}
$$

we ensure that each variable x_i is constrained to a single interval. With this modeling strategy, we modify DP-BASIC to arrive at the following formulation.

$$\begin{aligned}
&\bigvee_{j=1,\ldots,m_i} \begin{bmatrix} Y^i_j \\ x_i = (\pi^i_j - \pi^i_{j-1})u_i + \pi^i_{j-1} \end{bmatrix}, && \forall i \in \{1,\ldots,d\} \\
&\begin{bmatrix} \bigwedge_{i=1,\ldots,d} Y^i_{k_i} \\ \boldsymbol{c}_P^\mathsf{T}\boldsymbol{\beta} \le z_P \end{bmatrix} \vee \begin{bmatrix} \bigvee_{i=1,\ldots,d} \neg Y^i_{k_i} \\ z_P = 0 \end{bmatrix}, && \forall P_k \in \Pi \\
&\underset{j=1,\ldots,m_i}{\veebar} Y^i_j, && \forall i \in \{1,\ldots,d\} \\
&\sum_{P\in\Pi} z_P \le z \\
&\boldsymbol{\beta} = \boldsymbol{B}^d_p(\boldsymbol{u}) \\
&\boldsymbol{0} \le \boldsymbol{\beta} \le \boldsymbol{1} \\
&\boldsymbol{0} \le \boldsymbol{u} \le \boldsymbol{1}
\end{aligned} \tag{DP-GRID}$$

Above, the $|\Pi|$ disjunctions for the polynomial pieces are conditioned on the expression $\bigwedge_{i=1,\ldots,d} Y^i_{k_i}$. This condition will be *True* only for polytope P_k, ensuring that the polynomial pieces are exclusively selected. Otherwise, z_P is set to zero.

4.3 Partition-of-unity cut

The DP-GRID formulation can be augmented with a *partition-of-unity* cut that may strengthen its relaxation. The cut is given by the identity for Bernstein polynomials in (12), i.e. $\boldsymbol{1}^\mathsf{T}\boldsymbol{\beta} = 1$, where $\boldsymbol{1}$ is vector of n ones. Adding this cut to DP-GRID yields a formulation which we name DP-CUT.

The partition-of-unity cut was introduced for B-splines in [14]. The same cut is applicable here, as the Bernstein polynomials are a special case of B-splines (see Sect. 3.4).

4.4 Expanded Bernstein basis

The preceding formulations do not utilize the tensor product structure of the multivariate basis $\boldsymbol{B}^d_p$; cf. (9). In these formulations, the multivariate basis is formed by the tensor product of all univariate bases, resulting in constraints with polynomials of degree dp. In the following formulation, the multivariate basis is expanded to exploit the inherent structure due to the tensor product. The expansion lets us add additional polyhedral cuts (as was done for the multivariate basis in Sect. 4.3) and reduce the maximum degree of any polynomial in the set of constraints to $\max\{2, p\}$ (for $p \ge 1$). It was demonstrated in [14] that an expansion of the tensor product can yield better mathematical programming formulations for multivariate splines.

Specifically, each univariate Bernstein basis is assigned to the $p+1$ auxiliary variables $\boldsymbol{\xi}^i = \boldsymbol{B}_p(u_i)$ for $i \in \{1,\ldots,d\}$. To express the multivariate Bernstein basis, we introduce additional auxiliary variables $\boldsymbol{\beta}^i$ for $i \in \{1,\ldots,d\}$, and the constraints $\boldsymbol{\beta}^1 = \boldsymbol{\xi}^1$ and $\boldsymbol{\beta}^{i+1} = \boldsymbol{\xi}^{i+1} \otimes \boldsymbol{\beta}^i$, $\forall i \in \{1,\ldots,d-1\}$. The multivariate basis is then represented by $\boldsymbol{\beta}^d$.

The expansion permits nonnegativity bounds and partition-of-unity cuts on the univariate basis functions $\boldsymbol{\xi}^i$ and the intermediate basis functions $\boldsymbol{\beta}^i$. For $i \in \{1,\ldots,d\}$, we add the constraints $\boldsymbol{\xi}^i \ge \boldsymbol{0}$, $\boldsymbol{1}^\mathsf{T}\boldsymbol{\xi}^i = 1$, $\boldsymbol{\beta}^i \ge \boldsymbol{0}$, and $\boldsymbol{1}^\mathsf{T}\boldsymbol{\beta}^i = 1$, where $\boldsymbol{0}$ and $\boldsymbol{1}$ are vectors of zeros and ones of appropriate sizes.

In addition, we include the upper bounds on the Bernstein polynomials given in "Appendix A.3". Denote with $\bar{\boldsymbol{B}}_p$ the upper bounds on the Bernstein polynomials $\boldsymbol{B}_p$. We then include the bounds $\boldsymbol{\xi}^i \leq \bar{\boldsymbol{B}}_p$ for $i \in \{1, \dots, d\}$.

The resulting formulation, with the expanded Bernstein basis and polyhedral cuts, is given below.

$$
\begin{aligned}
&\bigvee_{j=1,\dots,m_i} \begin{bmatrix} Y_j^i \\ x_i = (\pi_j^i - \pi_{j-1}^i)u_i + \pi_{j-1}^i \end{bmatrix}, && \forall i \in \{1, \dots, d\} \\
&\begin{bmatrix} \bigwedge_{i=1,\dots,d} Y_{k_i}^i \\ \boldsymbol{c}_P^\top \boldsymbol{\beta}^d \leq z_P \end{bmatrix} \vee \begin{bmatrix} \bigvee_{i=1,\dots,d} \neg Y_{k_i}^i \\ z_P = 0 \end{bmatrix}, && \forall P_k \in \Pi \\
&\underline{\bigvee}_{j=1,\dots,m_i} Y_j^i, && \forall i \in \{1, \dots, d\} \\
&\sum_{P \in \Pi} z_P \leq z \\
&\boldsymbol{\xi}^i = \boldsymbol{B}_p(u_i), && \forall i \in \{1, \dots, d\} \\
&\boldsymbol{0} \leq \boldsymbol{\xi}^i \leq \bar{\boldsymbol{B}}_p, \ \boldsymbol{1}^\top \boldsymbol{\xi}^i = 1, && \forall i \in \{1, \dots, d\} \\
&\boldsymbol{\beta}^1 = \boldsymbol{\xi}^1 \\
&\boldsymbol{\beta}^{i+1} = \boldsymbol{\xi}^{i+1} \otimes \boldsymbol{\beta}^i, && \forall i \in \{1, \dots, d-1\} \\
&\boldsymbol{0} \leq \boldsymbol{\beta}^i, \ \boldsymbol{1}^\top \boldsymbol{\beta}^i = 1, && \forall i \in \{1, \dots, d\} \\
&\boldsymbol{0} \leq \boldsymbol{u} \leq \boldsymbol{1}
\end{aligned}
\tag{DP-EXP}
$$

DP-EXP uses $d(p+1) + \sum_{i=1}^d (p+1)^i$ auxiliary variables to represent the multivariate polynomial basis. The univariate basis functions are expressed by $d(p+1)$ polynomial constraints of degree p, while $\sum_{i=1}^{d-1} (p+1)^{i+1}$ bilinear constraints are used to form the multivariate basis (not counting the $p+1$ constraints $\boldsymbol{\beta}^1 = \boldsymbol{\xi}^1$, which are linear). In addition, $2d$ linear partition-of-unity cuts are included.

4.5 Expanded monomial basis

With DP-EXP the maximum degree of the polynomial constraints were reduced to $\max\{2, p\}$ (for $p \geq 1$). For general polynomial constraints, it is possible to achieve a maximum degree of 2 by further expansion of the basis. We achieve this by utilizing the monomial basis $\boldsymbol{\xi}^i$ in variable u_i, setting $\xi_0^i = 1$ and $\xi_j^i = \xi_{j-1}^i u_i$ for $j \in \{1, \dots, p\}$. We compute the multivariate basis $\boldsymbol{\beta}^d$ as in DP-EXP, and form the polynomial pieces as $f_P = \boldsymbol{\alpha}_P^\top \boldsymbol{\beta}^d = \boldsymbol{\alpha}_P^\top \boldsymbol{M}_p^d(\boldsymbol{u})$, where the coefficients are given as

$$
\boldsymbol{\alpha}_P^\top = \boldsymbol{a}_P^\top \bigotimes_{j=1}^{d} R_{p,j}.
$$

Bounds $\boldsymbol{0} \leq \boldsymbol{\xi}^i \leq \boldsymbol{1}$ and $\boldsymbol{0} \leq \boldsymbol{\beta}^i \leq \boldsymbol{1}$ for all $i \in \{1, \dots, d\}$, follow from the fact that $\boldsymbol{0} \leq \boldsymbol{u} \leq \boldsymbol{1}$.

With the expanded monomial basis we obtain the following formulation.

$$
\begin{aligned}
&\bigvee_{j=1,\ldots,m_i} \begin{bmatrix} Y_j^i \\ x_i = (\pi_j^i - \pi_{j-1}^i) u_i + \pi_{j-1}^i \end{bmatrix}, && \forall i \in \{1, \ldots, d\} \\
&\begin{bmatrix} \bigwedge_{i=1,\ldots,d} Y_{k_i}^i \\ \boldsymbol{\alpha}_P^\mathsf{T} \boldsymbol{\beta}^d \leq z_P \end{bmatrix} \vee \begin{bmatrix} \bigvee_{i=1,\ldots,d} \neg Y_{k_i}^i \\ z_P = 0 \end{bmatrix}, && \forall P_k \in \Pi \\
&\underline{\bigvee}_{j=1,\ldots,m_i} Y_j^i, && \forall i \in \{1, \ldots, d\} \\
&\sum_{P \in \Pi} z_P \leq z \\
&\xi_0^i = 1, && \forall i \in \{1, \ldots, d\} \\
&\xi_j^i = \xi_{j-1}^i u_i, && \forall j \in \{1, \ldots, p\}, i \in \{1, \ldots, d\} \\
&\mathbf{0} \leq \boldsymbol{\xi}^i \leq \mathbf{1}, && \forall i \in \{1, \ldots, d\} \\
&\boldsymbol{\beta}^1 = \boldsymbol{\xi}^1 \\
&\boldsymbol{\beta}^{i+1} = \boldsymbol{\xi}^{i+1} \otimes \boldsymbol{\beta}^i, && \forall i \in \{1, \ldots, d-1\} \\
&\mathbf{0} \leq \boldsymbol{\beta}^i \leq \mathbf{1}, && \forall i \in \{1, \ldots, d\} \\
&\mathbf{0} \leq \boldsymbol{u} \leq \mathbf{1}
\end{aligned}
\tag{DP-MON}
$$

Remark 3 The partition-of-unity cuts that were added for the Bernstein basis in Sect. 4.3, can be applied to the monomial basis via a transformation. Let $T_p = Q_p^{-1}$ so that $\boldsymbol{B}_p = T_p \boldsymbol{M}_p$, where Q_p is the transformation matrix in "Appendix A.1". Inserting this mapping into the cut constraints we obtain for the monovariable basis $\mathbf{1}^\mathsf{T} T_p \boldsymbol{\xi}^i = 1$ for $i \in \{1, \ldots, d\}$. It can be shown that these cuts are redundant since $\mathbf{1}^\mathsf{T} T_p \boldsymbol{\xi}^i = \xi_0^i = 1$. Furthermore, it can be shown that the transformed upper bounds are redundant as well; that is, $\boldsymbol{\xi}^i \leq \mathbf{1} \leq Q_p \bar{\boldsymbol{B}}_p$. We have thus omitted these in DP-MON.

5 MINLP formulations for piecewise polynomial functions

Algorithmic approaches for mathematical programming problems with disjunctions either reformulate the disjunction to enable mixed-integer programming, or seeks to exploit the disjunctive constraints explicitly in a branch-and-bound or cutting-plane algorithm, possibly through combinations thereof [3,52]. Linear [1] and convex nonlinear disjunctions [6] may be reformulated either by its convex hull representations or by big-M reformulations. Pertaining to the linear disjunction in DP-3, the convex hull formulation [1] is at least as tight as big-M reformulations, though requiring more variables and constraints. Big-M formulations, on the other hand, are known to be prone to the choice of the big-M parameters, and often yield weaker relaxations. For large nonlinear, possibly nonconvex DP problems, it is important to keep the size of reformulation small, in which big-M reformulations may be advantageous [53]. Consequently, we pursue big-M based MINLP reformulations of the DP models of PWP constraints in Sect. 4. Utilizing the bounds derived in Sect. 4.1 upon an elementary big-M reformulation of DP-BASIC yields the MINLP formulation

$$
\begin{aligned}
&\left.\begin{aligned}
&x_i - (\pi^i_{k_i} - \pi^i_{k_i-1})u_i - \pi^i_{k_i-1} \le (\pi^i_{m_i} - \pi^i_{k_i-1})(1-y_P)\\
&x_i - (\pi^i_{k_i} - \pi^i_{k_i-1})u_i - \pi^i_{k_i-1} \ge (\pi^i_{0} - \pi^i_{k_i})(1-y_P)
\end{aligned}\right\} && \forall P_k \in \Pi,\ i \in \{1,\dots,d\}\\
&\boldsymbol{c}_P^{\mathsf{T}}\boldsymbol{\beta} - z_P \le M_P^U(1-y_P), && \forall P \in \Pi\\
&z_P^L y_P \le z_P \le z_P^U y_P, && \forall P \in \Pi\\
&\sum_{P\in\Pi} y_P = 1\\
&\sum_{P\in\Pi} z_P \le z\\
&\boldsymbol{\beta} = \boldsymbol{B}_p^d(\boldsymbol{u})\\
&\boldsymbol{0} \le \boldsymbol{\beta} \le \boldsymbol{1}\\
&\boldsymbol{0} \le \boldsymbol{u} \le \boldsymbol{1}\\
&y_P \in \{0,1\}, && \forall P \in \Pi
\end{aligned}
\tag{MINLP-BASIC}
$$

where $M_P^U = c_P^U - z_P^L$, $z_P^L = \min\{0, c_P^L\}$, $z_P^U = \max\{0, c_P^U\}$, and a binary variable y_P replaces Y_P, for each $P \in \Pi$. MINLP-BASIC has $n = \dim(\mathbb{P}_p^d)$ nonlinear constraints; all other constraints are linear. The formulation has $|\Pi| + d + n$ continuous auxiliary variables $\{z_P\}_{P\in\Pi}$, $\boldsymbol{u}$ and $\boldsymbol{\beta}$, and $|\Pi|$ binary variables $\{y_P\}_{P\in\Pi}$.

5.1 Reformulation of DP-GRID, DP-EXP, and DP-MON

Analogous to the derivation of MINLP-BASIC, a MINLP reformulation of DP-GRID is obtained by introducing a binary variable y_j^i for each boolean variable Y_j^i, and applying a big-M reformulation. The last disjunction requires special treatment, since each term contains a conjunction of clauses connected by the AND operator. This means that the term for P_k in the disjunction should be True if and only if $Y^1_{k_1} = \cdots = Y^d_{k_d} =$ True. A big-M reformulation for a given P_k can then be formed as $\boldsymbol{c}_P^{\mathsf{T}}\boldsymbol{\beta} - z_P \le M_P^U(d - \sum_{i=1}^d y^i_{k_i})$, leading to the following MINLP formulation.

$$
\begin{aligned}
&\left.\begin{aligned}
&x_i - (\pi^i_j - \pi^i_{j-1})u_i - \pi^i_{j-1} \le (\pi^i_{m_i} - \pi^i_{j-1})(1 - y^i_j)\\
&x_i - (\pi^i_j - \pi^i_{j-1})u_i - \pi^i_{j-1} \ge (\pi^i_0 - \pi^i_j)(1 - y^i_j)
\end{aligned}\right\} \quad \begin{aligned}&\forall i \in \{1,\dots,d\},\\ &j \in \{1,\dots,m_i\}\end{aligned}\\
&\boldsymbol{c}_P^\top \boldsymbol{\beta} - z_P \le M_P^U\Big(d - \sum_{i=1}^{d} y^i_{k_i}\Big), \quad \forall P_k \in \Pi\\
&z_P^L y^i_{k_i} \le z_P \le z_P^U y^i_{k_i}, \quad \forall i \in \{1,\dots,d\}, P_k \in \Pi\\
&\sum_{j=1}^{m_i} y^i_j = 1, \quad \forall i \in \{1,\dots,d\}\\
&\sum_{P\in\Pi} z_P \le z\\
&\boldsymbol{\beta} = \boldsymbol{B}^d_p(\boldsymbol{u})\\
&\boldsymbol{0} \le \boldsymbol{\beta} \le \boldsymbol{1}\\
&\boldsymbol{0} \le \boldsymbol{u} \le \boldsymbol{1}\\
&y^i_j \in \{0,1\}, \quad \forall i \in \{1,\dots,d\}, j \in \{1,\dots,m_i\}
\end{aligned}
\tag{MINLP-GRID}
$$

We note that the MINLP-GRID has $\sum_{i=1}^d m_i$ binary variables for polytope selection, in contrast to $|\Pi| = \prod_{i=1}^d m_i$ as for the MINLP-BASIC formulation.

In an analogous fashion, we derive MINLP reformulations of DP-CUT, DP-EXP, and DP-MON, and denote them MINLP-CUT, MINLP-EXP, MINLP-MON, respectively. We note here that MINLP-MON consists of linear and bilinear constraints only, and is hence a MIQCP formulation.

5.2 Summary of formulations

The size of the formulations in terms of number of variables and constraints are summarized in Table 1. To ease comparisons, we assume that $m_1 = \cdots = m_d = M$, so that $|\Pi| = M^d$, and utilize the geometric series

$$g_d(k) := \sum_{i=1}^{d} k^i = (k^d - 1)k/(k-1), \qquad \text{for } k \ne 1.$$

The three formulations MINLP-BASIC, MINLP-GRID, and MINLP-CUT have $n = \dim(\mathbb{P}^d_p) = (p+1)^d$ nonlinear equality constraints. These constraints model the Bernstein basis functions that span $\mathbb{P}^d_p$, and are nonconvex since the Bernstein basis functions are polynomials of degree dp. MINLP-EXP has $d(p+1)$ polynomial constraints of degree p, and $g_d(p+1) - (p+1)$ bilinear constraints. Finally, MINLP-MON has $2d$ less nonlinear constraints than MINLP-EXP, all being bilinear constraints. The reduction in nonlinear constraints is due to some basis functions being constant (equal to one).

Comparing number of binary variables, MINLP-BASIC seems to be at a disadvantage since $M^d \ge dM$ for $M > 1$ and $d \ge 1$. The one-dimensional case ($d = 1$) is an exception since MINLP-BASIC then becomes equivalent to MINLP-GRID.

Table 1 Sizes of formulations

Formulation	# Continuous variables*	# Binary variables
MINLP-BASIC	$n + d + M^d$	M^d
MINLP-GRID	$n + d + M^d$	dM
MINLP-CUT	$n + d + M^d$	dM
MINLP-EXP	$d(p+1) + g_d(p+1) + n + d + M^d$	dM
MINLP-MON	$d(p+1) + g_d(p+1) + n + d + M^d$	dM
MIQCP-CUT [14]	$\frac{dp}{2}(3p-1) + pdM + g_d(M+p)$	$2dp + dM$
	# Linear constraints**	# Nonlinear constraints
MINLP-BASIC	$2 + (2d+3)M^d$	n
MINLP-GRID	$2dM + d + 1 + 3M^d$	n
MINLP-CUT	$2dM + d + 2 + 3M^d$	n
MINLP-EXP	$2dM + 3d + p + 1 + 3M^d$	$(d-1)(p+1) + g_d(p+1)$
MINLP-MON	$2dM + 3d + p + 1 + 3M^d$	$-2d + (d-1)(p+1) + g_d(p+1)$
MIQCP-CUT [14]	$2 + d(1 + 3p + 3p^2) + 2(p+1)dM + 2g_d(M+p)$	$\frac{dp}{2}(3p-1) + pdM + g_d(M+p)$

*Not counting $\boldsymbol{x}$ and z
**Not counting variable bounds

Finally, we note that the MIQCP-CUT formulation from [14] scales poorly in the number of nonlinear constraints due to the terms pdM and $g_d(M+p)$. It also has $2dp$ more binary variables than the formulation based on MINLP-GRID.

6 Numerical study

To benchmark the performance of the proposed MINLP formulations, we conducted a three-part numerical study. In the first part we minimized randomly generated cubic splines of varying input dimensions. In the second part, we solved a set of optimization problems involving five randomly generated PWP constraints of varying degrees. Finally, we solved a realistic production optimization case with ten PWP constraints.

The five formulations MINLP-BASIC, MINLP-GRID, MINLP-CUT, MINLP-EXP, and MINLP-MON, were solved using the global optimization solver BARON [46]. We compare the computational performance of the proposed MINLP formulations with the MIQCP-CUT formulation proposed in [14]. This formulation was also solved using BARON. We further include the results from solving the test problems using the special-purpose spline solver CENSO, which solves spline constrained problems as NLPs using a spatial branch-and-bound algorithm [16].

All solvers were run with an absolute ϵ-convergence termination criteria of $\epsilon = 1 \cdot 10^{-6}$. Remaining settings were left at default values. The problems were solved on a computer equipped with an Intel Core i7-8700K processor and 32 GB of RAM.

6.1 Minimization of randomly generated PWPs

Three sets of test problems were created by randomly generating cubic splines. The three sets contain problems with a piecewise polynomial objective function in one, two, and three variables, respectively. For the monovariable problems (*random1d*), the variable is discretized into ten intervals, leading to ten polynomial pieces. The set of problems in two variables (*random2d*) have an objective function defined on a 10×10 grid with 100 polynomial pieces. Finally, the problems with three variables (*random3d*) have an objective function defined on a $6 \times 6 \times 6$ grid, for a total of 216 n-orthotopes. Thus, the hardest problems (*random3d*) contain 216 polynomial pieces of degree $dp = 3 \times 3 = 9$. Properties of the sets of test problems are summarized in Table 2.

The problems are in the epigraph form $\min_{\boldsymbol{x},z} \left\{z : f(\boldsymbol{x}) \leq z,\ \boldsymbol{x} \in D \subset \mathbb{R}^d\right\}$, where $f : D \to \mathbb{R}$ is a piecewise polynomial function. Continuous cubic B-splines with equidistant knots are constructed by randomly drawing coefficients from a Gaussian distribution with zero mean and unity standard deviation, that is $c_i \sim \mathcal{N}(0,1),\ \forall i = 0,\dots,n-1$. Figure 3 shows one of the generated univariate cubic splines. Using the procedure described in "Appendix B", the B-spline is transformed into a piecewise polynomial f compatible with Definition 2.

Table 2 Test sets of piecewise polynomial problems

Problem	d	p	pd	$\lvert\Pi\rvert$	# instances
random1d	1	3	3	10	100
random2d	2	3	6	100	100
random3d	3	3	9	216	100

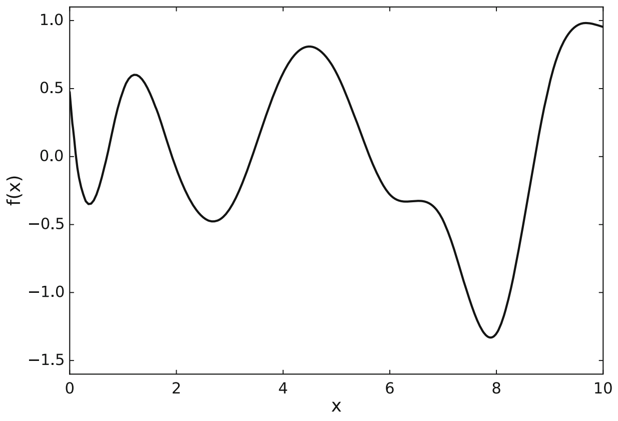

Fig. 3 A cubic spline with randomly generated coefficients. The function has several local minima. The global minimum for $x \in [0, 10]$ lies close to $x = 8$

Table 3 Mean solve times in seconds for randomly generated cubic splines

Formulation/problem	random1d	random2d	random3d
MINLP-BASIC	0.042	2.45	(*) 95.6
MINLP-GRID	0.042	3.75	104.9
MINLP-CUT	0.027	1.42	25.8
MINLP-EXP	**0.021**	**0.98**	**11.2**
MINLP-MON	0.028	2.57	(*) 2769.0
MIQCP-CUT	0.107	2.32	25.7
NLP	0.012	0.06	0.4

The lowest mean times among the MIP formulations are marked in bold
*Solver did not converge before the time limit of 3600 s on some problems

Table 3 gives a summary of the full results, which are reported in “Appendix C”, Table 6. Comparing the mean solve times, we see that MINLP-EXP outperforms the other MIP formulations, including the MIQCP-CUT formulation, on all three test sets. CENSO [16] had the lowest solve time on all problems.

For the one-dimensional PWPs in *random1d*, the MINLP-BASIC and MINLP-GRID formulations are equivalent and achieve similar results. For *random2d* and *random3d*, MINLP-BASIC seems to perform slightly better than MINLP-GRID when comparing mean solve times. However, for *random3d* the MINLP-BASIC formulation has a higher variation in the solve times as seen from the standard deviation in Table 6, with some problems not being solved within the time limit. From these results it is difficult to decide which is the better of the two formulations. However, the improvements to MINLP-GRID made in MINLP-CUT and MINLP-EXP, have a large effect on solve times, especially for the three-dimensional PWPs in *random3d*.

We note that the monomial formulation MINLP-MON performs quite poorly on the *random3d* test set. It thus seems that the Bernstein basis is beneficial for these problems.

6.2 Optimization with randomly generated PWP constraints

We solved test problems of the following form:

$$\begin{aligned} &\min z_0 \\ &\text{s.t. } f_0(\boldsymbol{x}) = z_0 \\ &\quad f_i(\boldsymbol{x}) \leq z_0, \ \forall i \in \{1, \ldots, 4\} \\ &\quad \boldsymbol{x} \in D = [0, 10]^2 \end{aligned} \tag{31}$$

where $\{f_0, f_1, \ldots, f_4\}$ are PWPs in two variables ($\boldsymbol{x}$). Three sets of test problems were generated, each with 20 problems consisting of either bi-linear, bi-quadratic, or bi-cubic PWPs. The PWPs were randomly generated as described in Sect. 6.1, and constructed on a 10×10 grid, partitioning D.

A summary of the results is given in Table 4. A complete table of results is given in "Appendix C", Table 7. A boxplot of the solve times for bi-cubic PWPs is shown in Fig. 4.

Comparing the MIP formulations, MINLP-EXP has the overall best performance on the bi-quadratic and bi-cubic problems in terms of mean solve time. On the most challenging

Table 4 Solve times in seconds for problems constrained by randomly generated PWPs

Formulation/problem	Bi-linear	Bi-quadratic	Bi-cubic
MINLP-BASIC	112.0	195.6	357.5
MINLP-GRID	37.2	56.1	108.7
MINLP-CUT	35.6	52.5	94.3
MINLP-EXP	34.8	**38.4**	**68.7**
MINLP-MON	31.7	69.2	296.1
MIQCP-CUT	**21.4**	58.8	361.0
NLP	1.0	1.7	3.2

The lowest mean times among the MIP formulations are marked in bold

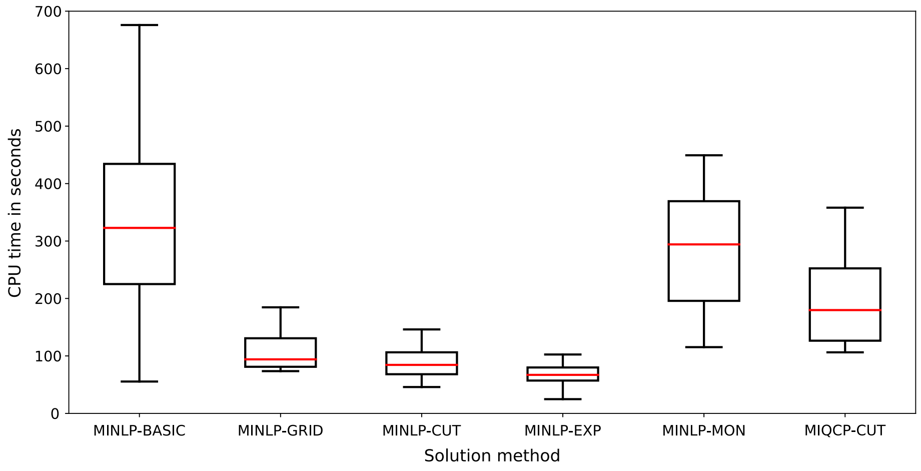

Fig. 4 A boxplot showing the results for problems with bi-cubic PWP constraints. Each box extends from the first to third quartile, and shows the median in red. The whiskers extending from each box represent the lower and upper value still within the lower and upper 1.5 interquartile range, respectively. Outlying values are not shown

set of bi-cubic problems, the MIQCP-CUT formulation performs worst on average, followed by MINLP-BASIC and MINLP-MON. On the bi-linear problems, however, the MIQCP formulations MIQCP-CUT and MINLP-MON performs best. The results demonstrate that these formulations scale poorly with the degree p.

It is clear from the results that MINLP-BASIC is inferior to MINLP-GRID and the formulations derived from MINLP-GRID. This is to be expected based on the formulation sizes in Table 1, which shows that the number of binary variables in MINLP-BASIC scales exponentially with d. The developments made in MINLP-GRID, MINLP-CUT, and MINLP-EXP improve solve times and reduce variability. For example, the addition of cuts in MINLP-CUT and MINLP-EXP is clearly advantageous. We finally note that CENSO has the lowest solve times on all problems.

6.3 Production optimization case

To test the formulations on a real large-scale problem we solved a production optimization case adopted from [15]. The case involves eight subsea petroleum wells, each producing a mix of oil, gas, and water. The fluid streams are routed to a topside processing facility via two pipelines (often called *risers*). The objective is to maximize oil production by routing and choking the well flows, satisfying constraints on momentum balances and variable bounds.

The production system was modeled by a directed acyclic graph $G = (N, E)$, with nodes $N = N^{\mathrm{w}} \cup N^{\mathrm{m}} \cup N^{\mathrm{s}}$ and edges $E = E^{\mathrm{d}} \cup E^{\mathrm{r}}$. An illustration of the graph is given in Fig. 5. Notice that each well node $i \in N^{\mathrm{w}}$ has two leaving *discrete edges* $(i, j) \in E^{\mathrm{d}}$ and $(i, k) \in E^{\mathrm{d}}$ to the manifold nodes $j, k \in N^{\mathrm{m}}$. By allowing zero or one of these edges to be active, we model routing of fluid flows and shut-in of wells. The total flow entering a manifold node $i \in N^{\mathrm{m}}$ is then led to a respective separator node $j \in N^{\mathrm{s}}$ via a riser edge $(i, j) \in E^{\mathrm{r}}$.

The variables in the problem are the pressure p_i at each node $i \in N$, the pressure drop Δp_e on each edge $e \in E$, the flow rate $q_{e,c}$ of phase $c \in C$ on edge $e \in E$, and the binary variable y_e on the discrete edge $e \in E^{\mathrm{d}}$. The variables y_e are used to model on/off switching of the discrete edges ($y_e = 1$ means that the edge is on/open, while $y_e = 0$ means that it is off/closed). In addition, auxiliary variables $q_{e,\mathrm{liq}}$ are introduced to model the liquid flow (oil + water) on each riser edge $e \in E^{\mathrm{r}}$. For further details about the modeling approach we refer the reader to [15].

The problem includes ten nonlinearities which were modeled using B-splines. The eight wells were modeled by well performance curves $f_i(p_i)$ for $i \in N^{\mathrm{w}}$, represented by polynomial basis functions on nine boxes. The pressure drop over the two risers were modeled by B-splines $g_e(q_{e,\mathrm{liq}}, q_{e,\mathrm{gas}}, p_i)$ for $e \in E^{\mathrm{r}}$, composed of polynomial basis functions on 64

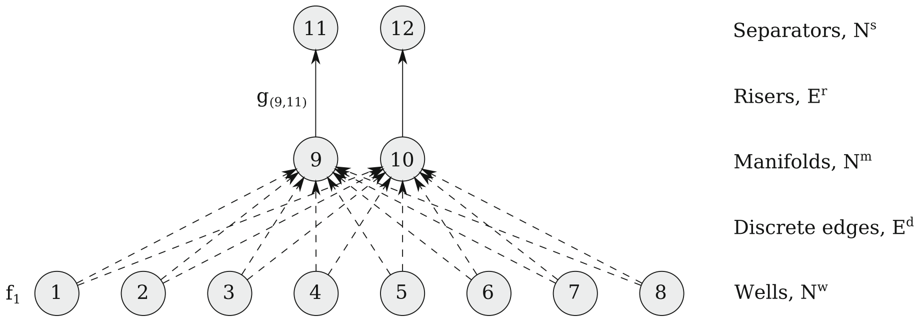

Fig. 5 Production system flow graph, with nodes represented by grey circles and edges by arrows. Discrete (on/off) edges are dashed. The nonlinearities f_1 and $g_{(9,11)}$, related to Node 1 and Edge (9, 11), are shown

boxes. The splines were fitted to simulated data from a multiphase flow simulator calibrated to real well test data. To give an example on problem size, for MINLP-BASIC the complete problem has 216 binary variables, 200 related to the splines, and 16 related to flow routing.

The results for various PWP degrees ($p = 1, 2, 3$) are given in Table 5. Among the MIP formulations, MINLP-EXP performed best. With this formulation, BARON solved the problems with $p = 1$ and $p = 2$ within the practical time limit of 3600 s. For $p = 3$, MINLP-EXP gave the best solution, although it did not close the optimality gap. This confirms the merits of MINLP-EXP observed in Tables 6 and 7 for increasing problem sizes.

We again observe that MIQCP-CUT and MINLP-MON performs well for $p = 1$, but poorly for higher degrees. The NLP solution method employed by CENSO does not seem to be affected much by the degree. In fact, it performs well for $p = 2$ and $p = 3$, which can be attributed to the availability of continuous derivatives in NLP searches,[2] and the weak dependence of the linear relaxations on p.

7 Concluding remarks

This paper has presented a MIP modeling framework for solving mathematical programs with continuous and semi-continuous PWP constraints. The numerical study indicates that among the derived mixed-integer formulations, MINLP-EXP has the best overall performance. Currently, it is the best-performing MIP formulation for PWP-constraints (solved by BARON). Although the MIQCP-CUT formulation is outperformed by several of the MINLP formulations, it has an advantage in that it is rather straightforward to implement when the PWP is given as a B-spline. In this case, the MINLP formulations require an extra preprocessing step to bring the spline to the PWP form in (18). With strong requirements on solve speed the special-purpose spline-based solver CENSO is still the best option, but it comes at the cost of using experimental software.

CENSO holds a great advantage in that it treats each PWP constraint as a *single* black-box constraint when searching for feasible solutions. Although the MINLP formulations herein allow the application of general-purpose global optimization solvers, the solution efficiency is still hampered by these solvers' lack of support for treating PWPs as black-box functions. These solvers must satisfy all the constraints in the applied MINLP formulation when searching for feasible solutions (see formulation sizes in Table 1).

There is a growing use of splines for modeling purposes, causing a demand for PWP support in optimization solvers. To this end, the MINLP formulations presented in this paper provide a basis for solving PWP constrained engineering and economic optimization problems. Our strategy of modeling a PWP constraint via its epigraph provides a general framework compatible with semi-continuous piecewise polynomials and B-splines, encompassing a broad set of spline modeling techniques. Furthermore, the formulations herein may for many applications reduce the need for special-purpose solvers like CENSO.

Our study confirms that the main difficulties in solving PWP constrained problems lie in the modeling of the polynomial basis $\mathbb{P}_p^d$ (requiring non-convex constraints) and the grid partitioning of the domain (requiring integer variables), which both contribute to the hardness of these problems. Besides optimization software support for PWP constraints, elements from related fields such as sum-of-squares programming, semidefinite relaxations and geometric programming, may be explored in combination with the proposed framework to improve the

[2] For PWPs in $\mathcal{C}^2$, CENSO may evaluate continuous first- and second-order derivatives to accelerate the search.

modeling of $\mathbb{P}_p^d(G)$. We also emphasize that the formulations presented are exact; approximations of these formulations may hence aid efforts toward reducing the computational demand.

Acknowledgements Open Access funding provided by NTNU Norwegian University of Science and Technology (incl St. Olavs Hospital - Trondheim University Hospital).

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

A Polynomials

This appendix provides some important transformations for manipulating polynomials. These transformations can be found in most textbooks treating polynomials and are reproduced here without proofs; cf. [42].

A.1 Transformation between the monomial and Bernstein basis

There exist a linear mapping $Q_p \in \mathbb{R}^{(p+1)\times(p+1)}$ that transforms a p-th degree Bernstein basis $\boldsymbol{B}_p = \{B_{i,p}\}_{i=0}^p$ to a p-th degree monomial basis $\boldsymbol{M}_p = \{M_i\}_{i=0}^p$; that is, $\boldsymbol{M}_p = Q_p\boldsymbol{B}_p$. The transformation matrix Q_p is an upper triangular matrix given as:

$$Q_p(i,j) = \begin{cases} 0, & i > j \\ \binom{j}{i}/\binom{p}{i}, & i \le j \end{cases} \tag{32}$$

Conversely, the inverse mapping $\boldsymbol{B}_p = Q_p^{-1}\boldsymbol{M}_p$ transforms a monomial basis to a Bernstein basis.

A.2 Reparametrization of the monomial basis

There exist a linear mapping $R_p \in \mathbb{R}^{(p+1)\times(p+1)}$ that reparametrizes a monomial basis $\boldsymbol{M}_p(u)$ on $u \in [0,1]$, to a monomial basis $\boldsymbol{M}_p(x)$ on $x \in [a,b]$; that is, $\boldsymbol{M}_p(x) = R_p\boldsymbol{M}_p(u)$. R_p is a lower triangular matrix given as:

$$R_p(i,j) = \begin{cases} 0, & i < j \\ \binom{i}{j}(b-a)^j a^{i-j}, & i \ge j \end{cases} \tag{33}$$

Conversely, the inverse mapping $\boldsymbol{M}_p(u) = R_p^{-1}\boldsymbol{M}_p(x)$ reparametrizes a monomial basis from x to u.

A.3 Maximum of Bernstein polynomials

The Bernstein polynomial $B_{i,p}$ of degree p has an upper bound

$$B_{i,p} \le \bar{B}_{i,p} := \begin{cases} 1, & i = 0 \\ i^i p^{-p}(p-i)^{p-i}\binom{p}{i}, & i = 1,\dots,p. \end{cases} \tag{34}$$

B B-splines

The following proposition shows that the Bernstein basis is equivalent to the B-spline basis for a special knot sequence.

Proposition 6 (Equivalence of B-spline and Bernstein basis) *The B-spline basis functions* $\{N_{i,p}(x; \boldsymbol{t})\}_{i=0}^{p}$ *in the variable* $x \in [a, b)$ *defined by the knot sequence*

$$\boldsymbol{t} = \{\underbrace{a, \dots, a}_{p+1}, \underbrace{b, \dots, b}_{p+1}\}, \tag{35}$$

are equivalent to the Bernstein polynomials, i.e. for $y \in [0, 1)$ *we have that* $N_{i,p}(x; \boldsymbol{t}) = N_{i,p}((b-a)y + a; \boldsymbol{t}) = B_{i,p}(y)$. *For* $a = 0$ *and* $b = 1$, $N_{i,p}(x; \boldsymbol{t}) = B_{i,p}(x)$.

Proof Cf. [42,49]. □

In the following lemma, we utilize Proposition 6 to show the equivalence of B-splines and PWPs defined as in Definition 2.

Lemma 2 (Equivalence of B-splines and PWPs) *Given the space* $\mathbb{P}_p^d(G)$ *of piecewise degree p polynomials on a bounded domain D partitioned on a rectilinear grid G of left half-closed boxes specified in* (24). *Then* $\mathbb{S}_p^d(T) = \mathbb{P}_p^d(G)$, *if the knot sequences* $T = \{\boldsymbol{t}_i\}_{i=1}^{d}$ *are given as*

$$\boldsymbol{t}_i = \{\underbrace{\pi_0^i, \dots, \pi_0^i}_{p+1}, \dots, \underbrace{\pi_k^i, \dots, \pi_k^i}_{p+1}, \dots, \underbrace{\pi_{m_i}^i, \dots, \pi_{m_i}^i}_{p+1}\}, \quad \forall i \in \{1, \dots, d\}. \tag{36}$$

That is, the knot sequence $\boldsymbol{t}_i$ *for the B-spline basis in variable* x_i *contains all partition points with multiplicity* $p + 1$.

Proof Consider the one-dimensional case first ($d = 1$). The specified knot sequence parametrizes the B-spline to have $p + 1$ supported B-spline basis functions in each half-open knot interval $P_k = [\pi_k^1, \pi_{k+1}^1)$; the partition is thus equivalent to the partition of D by the grid G. According to Proposition 6, these basis functions are equivalent to the Bernstein basis, and hence spans the space $\mathbb{P}_p$ on P_k. For the general case of $d > 1$, we utilize the fact the multivariate B-spline basis functions are constructed using the Kronecker product. D is thus partitioned into left half-open boxes $P_k = [\pi_{k_1}^1, \pi_{k_1+1}^1) \times \cdots \times [\pi_{k_d}^d, \pi_{k_d+1}^d)$, equivalent to the partition given by G. Furthermore, the multivariate B-spline basis is equivalent to the multivariate Bernstein basis since they both are constructed by the Kronecker product. It follows that the multivariate B-spline basis spans the space $\mathbb{P}_p^d$ on P_k, and by equivalence of the bases $\mathbb{S}_p^d(T) = \mathbb{P}_p^d(G)$ on the domain D. □

There are a couple of technicalities with Lemma 2 that deserve a comment. First, the knot sequences in (36) are special since all knots have multiplicity $p + 1$. Any B-spline can be transformed to an equivalent B-spline with such knot sequences using a *knot insertion* method. The effect of raising the multiplicity of all knots to $p + 1$ is that the B-spline is decomposed into a set of disjoint (non-overlapping) polynomial pieces, as illustrated by Fig. 6. A virtue of knot insertion methods is that they do not geometrically alter the spline.

Second, in Lemma 2, the domain D is not required to be closed since the support of the B-spline is restricted to a half-open domain. However, a compact domain may be considered if the definition of the B-spline is extended to include support on the right boundary, and the

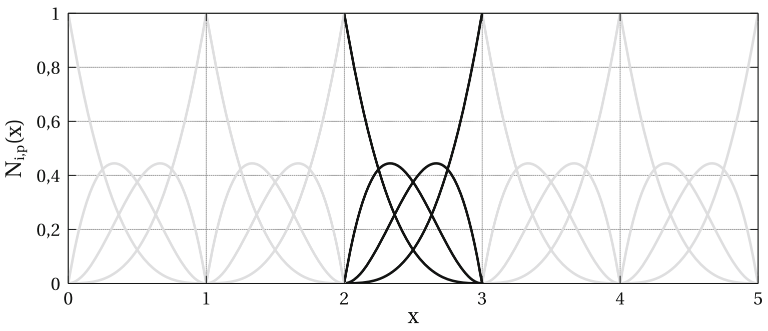

Fig. 6 B-spline basis functions for $p = 3$, $n = 8$, and knot sequence $\boldsymbol{t} = [0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5]$. Each basis function spans only a single knot interval. In each knot interval, the cubic B-spline is a cubic polynomial expressed by four basis functions that are equivalent to the Bernstein polynomials (on the unit interval)

rightmost boxes in G are closed as in (37). To close the domain of a (univariate) B-spline f we simply define

$$f(t_{n+p}) := \lim_{\substack{x \to t_{n+p} \\ x \leq t_{n+p}}} f(x), \tag{37}$$

and refer to the domain of f as $D = \text{cl}(D_o) = [t_0, t_{n+p}]$.

To complete the comparison of B-splines with our definition of PWPs, we note that a B-spline is lower semi-continuous if no internal knot is repeated more than p times. B-splines, for which the multiplicity of one or more internal knots are $p + 1$, may be continuous, lower semi-continuous, or discontinuous.

C Numerical results

The results from the numerical study in Sect. 6 are listed in Tables 5, 6 and 7.

Table 5 Results for production optimization problem (maximization problem)

Degree (p)	Formulation	Solver	Solve time*	$z^\star$	Gap (%)
1	MINLP-CUT	BARON	3600	126.20	3.36
	MINLP-GRID	BARON	3600	126.20	3.56
	MINLP-CUT	BARON	3600	126.20	5.09
	MINLP-EXP	BARON	373	126.20	0
	MINLP-MON	BARON	272	126.20	0
	MIQCP-CUT	BARON	143	125.24	0
	NLP	CENSO	184	126.20	0
2	MINLP-CUT	BARON	3600	102.51	32.849
	MINLP-GRID	BARON	3600	116.93	23.460
	MINLP-CUT	BARON	3600	–	–
	MINLP-EXP	BARON	1645	126.35	0

Table 5 continued

Degree (p)	Formulation	Solver	Solve time*	$z^\star$	Gap (%)
	MINLP-MON	BARON	3600	–	–
	MIQCP-CUT	BARON	3600	124.80	8.16
	NLP	CENSO	604	126.35	0
3	MINLP-CUT	BARON	3600	–	–
	MINLP-GRID	BARON	3600	–	–
	MINLP-CUT	BARON	3600	96.68	38.34
	MINLP-EXP	BARON	3600	105.25	24.45
	MINLP-MON	BARON	3600	–	–
	MIQCP-CUT	BARON	3600	–	–
	NLP	CENSO	343	126.36	0

*Solve time in seconds. Time limit was set to 3600 s

Table 6 Solve times in seconds for randomly generated cubic splines

Problem	Formulation	Solver	T_{med}	T_{mean}	T_{std}	T_{min}	T_{max}
random1d	MINLP-BASIC	BARON	0.050	0.042	0.027	0.001	0.100
	MINLP-GRID	BARON	0.050	0.042	0.028	0.001	0.110
	MINLP-CUT	BARON	0.030	0.027	0.020	0.001	0.080
	MINLP-EXP	BARON	0.020	0.021	0.014	0.001	0.060
	MINLP-MON	BARON	0.030	0.028	0.019	0.001	0.060
	MIQCP-CUT	BARON	0.110	0.107	0.059	0.001	0.250
	NLP	CENSO	0.013	0.012	0.007	0.001	0.031
random2d	MINLP-BASIC	BARON	1.225	2.453	2.723	0.060	12.49
	MINLP-GRID	BARON	1.585	3.745	4.355	0.430	17.08
	MINLP-CUT	BARON	0.900	1.415	1.253	0.260	6.810
	MINLP-EXP	BARON	0.700	0.980	0.729	0.320	4.170
	MINLP-MON	BARON	1.700	2.571	2.260	0.480	10.61
	MIQCP-CUT	BARON	1.620	2.317	2.078	0.200	11.30
	NLP	CENSO	0.058	0.063	0.036	0.001	0.195
random3d	MINLP-BASIC	BARON	32.4	95.6	359.3	0.9	*3600
	MINLP-GRID	BARON	50.0	104.9	219.3	5.4	2089
	MINLP-CUT	BARON	13.4	25.8	39.3	1.9	304.9
	MINLP-EXP	BARON	8.7	11.2	8.0	2.4	65.3
	MINLP-MON	BARON	3600	2769	1306	12.0	*3600
	MIQCP-CUT	BARON	15.9	25.7	24.8	2.7	140.9
	NLP	CENSO	0.35	0.36	0.19	0.04	0.92

*Solver did not converge before the time limit of 3600 s on some problems

Table 7 Solve times in seconds for problems constrained by randomly generated PWPs

Problem	Formulation	Solver	T_{med}	T_{mean}	T_{std}	T_{min}	T_{max}
Bi-linear	MINLP-BASIC	BARON	80.7	112.0	105.9	6.0	359.9
	MINLP-GRID	BARON	33.9	37.2	20.1	4.3	73.3
	MINLP-CUT	BARON	37.0	35.6	16.0	4.7	60.0
	MINLP-EXP	BARON	34.7	34.8	16.2	4.0	57.5
	MINLP-MON	BARON	32.7	31.7	15.1	4.3	58.9
	MIQCP-CUT	BARON	20.5	21.4	9.2	1.6	36.9
	NLP	CENSO	1.0	1.0	0.3	0.6	1.7
Bi-quad	MINLP-BASIC	BARON	159.1	195.6	104.6	64.0	491.4
	MINLP-GRID	BARON	54.1	56.1	11.8	32.6	74.2
	MINLP-CUT	BARON	52.7	52.5	14.4	24.4	87.8
	MINLP-EXP	BARON	35.8	38.4	14.2	18.0	82.0
	MINLP-MON	BARON	65.2	69.2	17.6	39.6	95.6
	MIQCP-CUT	BARON	56.2	58.8	10.6	42.0	85.2
	NLP	CENSO	1.7	1.7	0.4	0.8	2.3
Bi-cubic	MINLP-BASIC	BARON	322.8	357.5	188.2	55.5	886.5
	MINLP-GRID	BARON	94.1	108.7	31.6	73.6	184.6
	MINLP-CUT	BARON	84.4	94.3	45.9	45.9	259.1
	MINLP-EXP	BARON	67.1	68.7	23.3	24.8	118.2
	MINLP-MON	BARON	294.1	296.1	126.0	115.3	652.0
	MIQCP-CUT	BARON	179.8	361.0	660.9	106.3	3193.4
	NLP	CENSO	3.2	3.2	1.1	1.3	6.0

References

1. Balas, E.: Disjunctive programming and a hierarchy of relaxations for discret optimization problems. SIAM J. Algebraic Discrete Methods **6**(3), 466–486 (1985)
2. Beale, E.M.L., Tomlin, J.A.: Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables. OR **69**(447–454), 99 (1970)
3. Beaumont, N.: An algorithm for disjunctive programs. Eur. J. Oper. Res. **48**(3), 362–371 (1990)
4. Biegler, L.T.: Simultaneous methods for dynamic optimization. In: Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes, Chap. 10, pp. 287–324. SIAM, New York (2010)
5. Bragalli, C., D'Ambrosio, C., Lee, J., Lodi, A., Toth, P.: On the optimal design of water distribution networks: a practical MINLP approach. Optim. Eng. **13**(2), 219–246 (2012)
6. Ceria, S., Soares, J.: Convex programming for disjunctive convex optimization. Math. Program. **86**(3), 595–614 (1999)
7. Chen, X.: Smoothing methods for nonsmooth, nonconvex minimization. Math. Program. **134**(1), 71–99 (2012)
8. Conn, A.R., Mongeau, M.: Discontinuous piecewise linear optimization. Math. Program. **80**(3), 315–380 (1998)
9. Curry, H.B., Schoenberg, I.J.: On Pólya frequency functions IV: the fundamental spline functions and their limits. J. d'Anal. Math. **17**(1), 71–107 (1966)
10. Dantzig, G.B.: On the significance of solving linear programming problems with some integer variables. Econometrica **28**(1), 30–44 (1960)
11. Demeulenaere, B., Pipeleers, G., De Caigny, J., Swevers, J., De Schutter, J., Vandenberghe, L.: Optimal splines for rigid motion systems: a convex programming framework. ASME J. Mech. Des. **131**(10), 101004–101004-11 (2009)
12. Eilers, P.H., Marx, B.D.: Flexible smoothing with B-splines and penalties. Stat. Sci. **11**(2), 89–102 (1996)

13. Farouki, R.T.: The Bernstein polynomial basis: a centennial retrospective. Comput. Aided Geom. Des. **29**(6), 379–419 (2012)
14. Grimstad, B.: A MIQCP formulation for B-spline constraints. Optim. Lett. **12**(4), 713–725 (2018)
15. Grimstad, B., Foss, B., Heddle, R., Woodman, M.: Global optimization of multiphase flow networks using spline surrogate models. Comput. Chem. Eng. **84**, 237–254 (2016)
16. Grimstad, B., Sandnes, A.: Global optimization with spline constraints: a new branch-and-bound method based on B-splines. J. Glob. Optim. **65**(3), 401–439 (2016)
17. Grimstad, B., et al.: SPLINTER: a library for multivariate function approximation with splines (2015). http://github.com/bgrimstad/splinter. Accessed 16 May 2015
18. Hargraves, C., Paris, S.W.: Direct trajectory optimization using nonlinear programming and collocation. J. Guid. Control Dyn. **10**(4), 338–342 (1987)
19. Hasan, M.M.F., Karimi, I.: Piecewise linear relaxation of bilinear programs using bivariate partitioning. AIChE J. **56**(7), 1880–1893 (2010)
20. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning, Springer Series in Statistics, vol. 1, 2nd edn. Springer, New York (2009)
21. Hiriart-Urruty, J.B., Lemarechal, C.: Convex Analysis and Minimization Algorithms II—Advanced Theory and Bundle Methods. Springer, Berlin (1993)
22. Hofmann, T., Schölkopf, B., Smola, A.J.: Kernel methods in machine learning. Ann. Stat. **36**(3), 1171–1220 (2008)
23. Höllig, K.: Finite Element Methods with B-Splines. Society for Industrial and Applied Mathematics, Philadelphia (2003)
24. Holmberg, K.: Solving the staircase cost facility location problem with decomposition and piecewise linearization. Eur. J. Oper. Res. **75**(1), 41–61 (1994)
25. Jahanshahi, E., Grimstad, B., Foss, B.: Spline fluid models for optimization. In: Proceedings of the IFAC Symposium on DYCOPS, pp. 400–405, Trondheim (2016)
26. Jeroslow, R.G.: Representability in mixed integer programming, I: characterization results. Discrete Appl. Math. **17**, 223–243 (1987)
27. Jeroslow, R.G.: Representability of functions. Discrete Appl. Math. **23**(2), 125–137 (1989)
28. Keha, A.B., de Farias Jr, I.R., Nemhauser, G.L.: A branch-and-cut algorithm without binary variables for nonconvex piecewise linear optimization. Oper. Res. **54**(5), 847–858 (2006)
29. Knudsen, B.R., Foss, B.: Shut-in based production optimization of shale-gas systems. Comput. Chem. Eng. **58**, 54–67 (2013)
30. Li, W.: A conjugate gradient method for the unconstrained minimization of strictly convex quadratic splines. Math. Program. **72**(1), 17–32 (1996)
31. Lorentz, G.G.: Bernstein Polynomials. American Mathematical Soc., New York (2013)
32. Luo, Y.: Simulation-based optimization over discrete sets with noisy constraints. Ph.D. Thesis, University of Miami (2011)
33. Martinez, N., Anahideh, H., Rosenberger, J.M., Martinez, D., Chen, V.C., Wang, B.P.: Global optimization of non-convex piecewise linear regression splines. J. Glob. Optim. **68**(3), 563–586 (2017)
34. Mercy, T., Jacquod, N., Herzog, R., Pipeleers, G.: Spline-based trajectory generation for CNC machines. IEEE Trans. Ind. Electron. **66**(8), 6098–6107 (2019)
35. Misener, R., Floudas, C.A.: GloMIQO: global mixed-integer quadratic optimizer. J. Glob. Optim. **57**(1), 3–50 (2013)
36. Misener, R., Floudas, C.A.: ANTIGONE: algorithms for continuous/integer global optimization of nonlinear equations. J. Glob. Optim. **59**(2), 503–526 (2014)
37. Natali, J.M., Pinto, J.M.: Piecewise polynomial interpolations and approximations of one-dimensional functions through mixed integer linear programming. Optim. Methods Softw. **24**(4–5), 783–803 (2009)
38. Nesterov, Y.: Smooth minimization of non-smooth functions. Math. Program. **152**, 127–152 (2005)
39. Padberg, M.: Approximating separable nonlinear functions via mixed zero-one programs. Oper. Res. Lett. **27**(1), 1–5 (2000)
40. Park, J., Kim, Y., Eom, I., Lee, K.: Economic load dispatch for piecewise quadratic cost function using Hopfield neural network. IEEE Trans. Power Syst. **8**(3), 1030–1038 (1993)
41. Patrinos, P., Sarimveis, H.: Convex parametric piecewise quadratic optimization: theory, algorithms and control applications. Automatica **47**(8), 1770–1777 (2011)
42. Piegl, L.A., Tiller, W.: The NURBS Book. Springer, Berlin (1997)
43. Posa, M., Kuindersma, S., Tedrake, R.: Optimization and stabilization of trajectories for constrained dynamical systems. In: 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 1366–1373 (2016)
44. Prandoni, P., Vetterli, M.: Approximation and compression of piecewise smooth functions. Philos. Trans. Math. Phys. Eng. Sci. **357**(1760), 2573–2591 (1999)

45. Royset, J.O.: Approximations and solution estimates in optimization. Math. Program. **170**, 479–506 (2018)
46. Sahinidis, N.V.: BARON: a general purpose global optimization software package. J. Glob. Optim. **8**(2), 201–205 (1996)
47. Scholtes, S.: Nonconvex structures in nonlinear programming. Oper. Res. **52**(3), 368–383 (2004)
48. Schramm, H., Zowe, J.: A version of the bundle idea for minimizing a nonsmooth function: conceptual idea, convergence analysis, numerical results. SIAM J. Optim. **2**(1), 121–152 (1992)
49. Schumaker, L.L.: Spline Functions: Basic Theory, 3rd edn. Cambridge University Press, Cambridge (2007)
50. Sherali, H.D.: On mixed-integer zero-one representations for separable lower-semicontinuous piecewise-linear functions. Oper. Res. Lett. **28**(4), 155–160 (2001)
51. Shukla, R., Dragotti, P.L., Do, M.N., Vetterli, M.: Rate-distortion optimized tree structured compression algorithms for piecewise smooth images. IEEE Trans. Image Process. **14**(3), 343–359 (2005)
52. Stubbs, R.A., Mehrotra, S.: A branch-and-cut method for 0–1 mixed convex programming. Math. Program. **86**, 515–532 (1999)
53. Vecchietti, A., Lee, S., Grossmann, I.E.: Modeling of discrete/continuous optimization problems: characterization and formulation of disjunctions and their relaxations. Comput. Chem. Eng. **27**(3), 433–448 (2003)
54. Vielma, J.P.: Mixed integer linear programming formulation techniques. SIAM Rev. **57**(1), 3–57 (2015)
55. Vielma, J.P., Ahmed, S., Nemhauser, G.: Mixed-integer models for nonseparable piecewise-linear optimization: unifying framework and extensions. Oper. Res. **58**(2), 303–315 (2010)
56. Vielma, J.P., Nemhauser, G.L.: Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. Math. Program. **128**(1–2), 49–72 (2011)
57. Vigerske, S., Gleixner, A.: SCIP: global optimization of mixed-integer nonlinear programs in a branch-and-cut framework. Optim. Methods Softw. **33**, 1–31 (2017)
58. Vu, K.K., D'Ambrosio, C., Hamadi, Y., Liberti, L.: Surrogate-based methods for black-box optimization. Int. Trans. Oper. Res. **24**(3), 393–424 (2017)
59. Wang, W., Pottmann, H., Liu, Y.: Fitting B-spline curves to point clouds by curvature-based squared distance minimization. ACM Trans. Graph. **25**(2), 214–238 (2006)
60. Wechsung, A., Barton, P.I.: Global optimization of bounded factorable functions with discontinuities. J. Glob. Optim. **58**(1), 1–30 (2014)
61. Wegman, E.J., Wright, I.W.: Splines in statistics. J. Am. Stat. Assoc. **78**(382), 351–365 (1983)
62. Womersley, R.S., Fletcher, R.: An algorithm for composite nonsmooth optimization problems. J. Optim. Theory Appl. **48**(3), 493–523 (1986)
63. Yuan, Y., Fan, W., Pu, D.: Spline function smooth support vector machine for classification. J. Ind. Manag. Optim. **3**(3), 529–542 (2007)
64. Zang, I.: Discontinuous optimization by smoothing. Math. Oper. Res. **6**(1), 140–152 (1981)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.