Computers and Chemical Engineering 196 (2025) 109042

Contents lists available at ScienceDirect

Computers and Chemical Engineering

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

Piecewise linear approximation using J1 compatible triangulations for efficient MILP representation

Felix Birkelbach

Institute for Energy Systems and Thermodynamics, Getreidemarkt 9/E302, Vienna, 1060, Austria

ARTICLE INFO

Dataset link: https://doi.org/10.5281/zenodo.14669069

Keywords:
Piecewise-linear approximation
J1 triangulation
Triangulation on grid
Logarithmic formulation

ABSTRACT

For including piecewise linear (PWL) functions in MILP problems, the logarithmic convex combination (Log) formulation has been shown to yield very fast solving times. However, identifying approximations that can be used with Log is a big challenge since the approximation has to be compatible with a J1 triangulation. In this article, an algorithm is proposed that identifies approximations using J1 compatible triangulations. It seeks to satisfy the specified error tolerance with the minimum number of linear pieces, so that the MILP formulation is small. To evaluate the performance of the J1 approach it is applied to two sets of benchmark functions from literature and results are compared to state-of-the-art approaches.

Overall the J1 approach is shown to efficiently approximate functions in up to 3 dimensions. Especially for tight error tolerances, these J1 approximations require fewer auxiliary variables in MILP compared to alternative approaches.

1. Introduction

Despite the tremendous progress in the area of mixed integer non-linear programming solvers, mixed-integer linear programming (MILP) in combination with linearization of non-linear terms is still the method of choice for many applications. Since fast and reliable MILP solvers are readily available, this shifts much of the complexity from solving the optimization problem to finding viable piece-wise linear (PWL) approximations of the non-linear terms. The challenge is to find approximations that meet a specified error tolerance without compromising MILP solving times.

In recent years, interest in PWL approximation algorithms specifically for applications in MILP has risen. For one-dimensional functions, algorithms exist that are proven to identify minimum-breakpoint approximations (Rebennack and Krasko, 2020; Kong and Maravelias, 2020; Rebennack and Kallrath, 2015b). For two and higher dimensional functions no such algorithm exists, to the best of our knowledge. Though, algorithms have been developed to find efficient piecewise-linear approximations of higher dimensional functions. Toriello and Vielma (2012) proposed an algorithm that starts with a triangulation on an axis-parallel grid to approximate the non-linear function. They observed that the approximation error depends on the orientation of the simplices in each segment of the grid and proposed a MILP approach to select the optimal orientation. Rebennack and Kallrath (2015a) developed an algorithm to approximate a two-dimensional function by triangulating its domain. Kazda and Li (2021) introduced the "difference of convex" (DC) approach to find approximations of n-dimensional functions with polytopes instead of simplices. Since polytopes have more degrees of freedom than simplices, these approximations generally require considerably fewer linear segments than approximations that are based on triangulations. Ruela et al. (2024) used a J1 triangulation on an axis-parallel grid for the approximation and proposed a heuristic for optimizing the break points of the grid. They found that, while this approach works well for crude error tolerances, it requires a very large number of linear pieces to meet tight error tolerances.

An interesting research direction is the application of machine learning approaches for PWL approximation, specifically the use of networks with rectifier linear units (ReLU) (Grimstad and Andersson, 2019). Somewhat disconnected from the main body of literature are the approaches proposed by Obermeier et al. (2021) and Kämper et al. (2021). While the former uses Delaunay triangulations, the latter revolves around hinging-hyperplane-trees.

The impact of PWL approximations on MILP solving time is largely determined by the number of auxiliary variables that are required to implement the PWL function into the MILP problem. To yield fast MILP solving times, PWL approximations should introduce as few auxiliary variables as possible. In general, the number of auxiliary variables is proportional to the number of linear pieces. Consequently the goal of PWL approximation algorithms is to meet the error tolerance with the smallest possible number of linear pieces. The other major factor that affects MILP solving times is the MILP formulation used for integrating the PWL function into the MILP problem.

E-mail address: felix.birkelbach@tuwien.ac.at.

https://doi.org/10.1016/j.compchemeng.2025.109042
Received 14 November 2024; Received in revised form 17 January 2025; Accepted 7 February 2025
Available online 15 February 2025

Nomenclature

Acronyms

DC	Difference of convex (method)
Log	Logarithmic binary branching (formulation)
MC	Multiple choice (formulation)
MILP	Mixed integer linear programming
PWL	Piece-wise linear
RK	Rebennack and Kalrath (method)

Roman symbols

D	Domain of function
∂D	Boundary of the domain
$\hat{f}$	Piecewise linear function
f	Function to be approximated
$f_{\mathrm{D},d}$	Function value of data point d
f_v	Value of function at vertex v
$\mathcal{G}$	Grid
$\partial\mathcal{G}$	Boundary of the grid
J_1^α	J1 triangulation with scheme α
n	Number of dimensions
N_{data}	Number of data points
$\boldsymbol{x}$	Vector of coordinates
s	Simplex
s_i	Number of segments in dimension i
$S(\mathcal{T})$	Simplices of triangulation $\mathcal{T}$
$\mathcal{T}$	Triangulation
v	Vertex
vol_d	d-dimensional volume
$V(\mathcal{T})$	Vertices of triangulation $\mathcal{T}$
$\boldsymbol{x}_{\mathrm{D},d}$	Coordinate of data point d
$\boldsymbol{x}_v$	Coordinate of vertex v

Greek symbols

α, β	J1 scheme
λ	Weight factor

Subscripts

i	Dimension index
d	Data point index

A PWL function in one dimension is a linear spline, which can be translated to MILP with a SOS2 formulation. For two and higher dimensional functions, the formulations presented by Vielma et al. (2010) can be used. For the present study, three of these formulations are relevant: Multiple Choice (MC), disaggregated convex combination with logarithmic coding (DLog) and the logarithmic branching convex combination (Log). While the number of auxiliary binary variables grows proportional to the number of linear pieces for MC, it only grows proportional to the base-2 logarithm of the number of linear pieces for Log and DLog. For the DLog method, this advantage is offset by a comparably large number of continuous variables. Recently, other logarithmic branching schemes (binary and general integer zig-zag) have been proposed that put less strict requirements on the structure of the PWL function (Huchette and Vielma, 2019, 2023). However, finding these special branching schemes is difficult in itself and they may require slightly more auxiliary binary variables than the Log formulation.

Just by considering the number of auxiliary variables, the Log formulation can be expected to yield the best MILP performance. This has also been confirmed with computational experiments, which showed that the Log method resulted in between 3 and 4 times shorter solving times compared to the DLog method in a multi-commodity transportation problem; Log also outperformed MC (the fastest non-logarithmic formulation) by a factor of 2 (32 linear segments) and even 300 (128 linear segments) (Vielma et al., 2010). The performance of different MILP formulations has also been investigated in oil production optimization (Silva and Camponogara, 2014) and in unit commitment of hydropower units (Brito et al., 2020). Also in these studies, the Log formulation outperformed all other formulations by a large margin (the zig-zag formulations would probably also have performed well, but they were not yet available).

Despite these impressive results, the Log formulation is not routinely used to implement PWL functions in MILP problems. One can only speculate about the reason for this, but probably a reason are the fairly strict requirements that the Log formulation puts on the structure of the PWL function. Specifically that the triangulation of the function domain must be compatible with the J1 triangulation (Vielma and Nemhauser, 2011), which is a big challenge for PWL approximation and the lack of readily available tools may deter practitioners (Huchette and Vielma, 2023). Of the PWL approximation approaches in literature, only the approach by Ruela et al. (2024) produces J1 compatible approximations (Vielma et al., 2010; Silva and Camponogara, 2014) and Brito et al. (2020) used equally spaced grids without optimization). However, this approach is limited in that it is restricted to an axis-parallel grid, which results in a disproportionate number of linear pieces to meet tight error tolerances.

The obvious solution is to relax the restriction to an axis parallel grid. This, however, makes it very challenging to maintain compatibility with the J1 triangulation throughout the approximation process. In this paper, an algorithm to compute PWL approximation with deformed (i.e. non axis parallel) J1 triangulations is proposed. It is capable of satisfying tight error tolerances with a small number of linear pieces while leveraging the superior MILP performance of the Log formulation.

The remainder of the paper is organized as follows: In the next section, some background on J1 triangulations and MILP formulations for PWL functions is reviewed. Then, the approximation algorithm is introduced. In the Evaluation section, the algorithm is applied to two sets of benchmark functions and the results are compared to results reported in literature. The proposed algorithm is shown to be capable of finding efficient approximations of a wide range of functions. Results also show, that by leveraging the Log formulation these approximations introduce considerably fewer auxiliary variables to MILP problems than other approaches from literature in some cases.

2. Background

In their seminal papers (Vielma et al., 2010; Vielma and Nemhauser, 2011) Vielma et al. introduced the independent logarithmic branching scheme for the convex combination method (the Log formulation). They showed that this formulation is both small (the number of auxiliary binary variable increases logarithmically with the number of linear pieces) and ideal (the LP relaxation is tight) (Vielma, 2015). Consequently it generally yields very fast solving times, especially for PWL functions with many linear pieces. This has been confirmed in independent studies (Silva and Camponogara, 2014; Brito et al., 2020; Birkelbach et al., 2023).

The Log formulation requires the PWL function to be defined on a highly structured gridded triangulation. Specifically, the triangulation has to be compatible with the J1 triangulation (Todd, 1977), which is also referred to as the “union jack” triangulation, because of the similarity with the British flag in two dimensions.

The triangulation of the $[0,1]^n$ hypercube in n dimension is given by the simplices that can be constructed by going from the $(0,\dots,0)$ corner to the $(1,\dots,1)$ corner of the hypercube along its edges. There are exactly $n!$ direct paths. Each path visits $n+1$ corners, thus giving

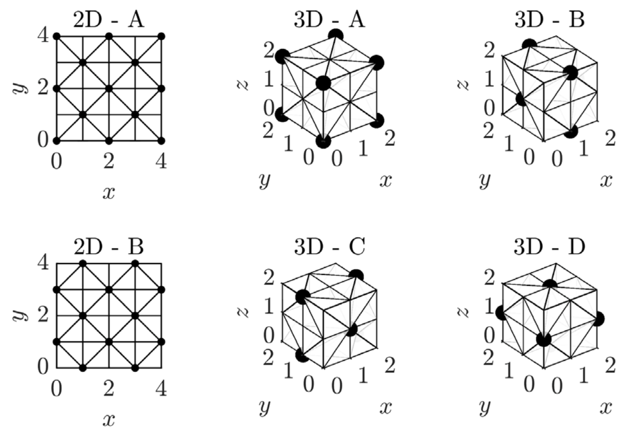

Fig. 1. Permutations of the J1 triangulation (J1 schemes) in two and three dimensions. The dots mark the center nodes, where all simplices of the adjacent grid segments meet.

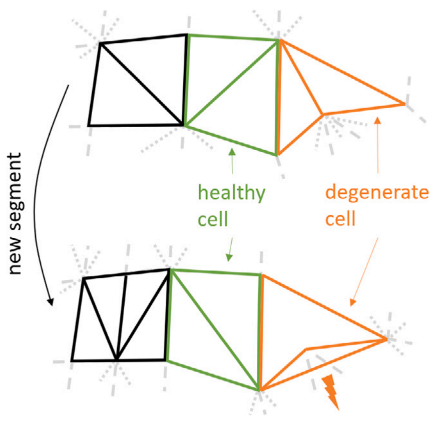

Fig. 2. Illustration of how adding a grid segment may result in an invalid triangulation if the grid is not in a healthy state.

the vertices of a simplex. The J1 triangulation of a grid is constructed by triangulating the first cell of the grid and then mirroring it over and over in each dimension (see Fig. 1 2D-A and 3D-A). This gives the characteristic point-wise symmetry with respect to each vertex of the grid and it makes the triangulation repeat periodically every two segments.

Alternatively, any other pair of vertices on opposite corners can be chosen to construct a J1 triangulation. Thus, there are 2^{n-1} possible orientations of the triangulation of the unit cube, each inducing a J1 triangulation of a grid. These permutations will be referred to as J1 schemes and they will play a key role in the approximation algorithm. For the two and tree dimensional case, the permutations are illustrated in Fig. 1, where the central nodes are marked with a dot to help discern the J1 schemes.

On coarse grids, the choice of the J1 scheme may have an impact on the accuracy of the approximation (Toriello and Vielma, 2012; Kazda and Li, 2023). Though, on fine grained grids, the approximation algorithm should be able to offset the effect of a sub-optimal choice of the J1 scheme by choosing the vertex positions accordingly.

For the Log formulation, the J1 triangulation does not necessarily have to be on an axis-parallel grid. The grid may be deformed, as long as the ordering of the simplices stays the same. Leveraging this flexibility is one of the key innovations of the algorithm that will be introduced in the next section. However, this also makes maintaining a valid J1 triangulation throughout the approximation process a challenge.

For the PWL function to be uniquely defined on the whole domain of the function that is to be approximated, the J1 triangulation has to cover the whole domain and simplices must not overlap. This constrains the deformation of the grid. A specific challenge arises, when grid segments are added or removed, because it "flips" the triangulations of all grid cells on one side of the added/removed segment. Consider the illustration in Fig. 2. On the top, the initial state of the grid where the triangulation is valid is shown. Splitting the segment on the left (while maintaining the J1 structure) causes the triangulation to the right to flip. While this is not an issue for the cell in the middle, it makes the triangulation of the cell on the right invalid. Since manipulating the grid is an essential part of the approximation algorithm, situations like this need to be avoided. When the grid is deformed, it needs to be kept in a healthy state, where the triangulation is valid for each orientation of the J1 triangulation (i.e. for each J1 scheme) at all times.

3. The approximation algorithm

Finding a PWL approximation of a multi-variate function is a complex mixed-integer non-linear problem, for which direct solution approaches are currently intractable. For this reason, the problem is divided into a series of sub-problems: fitting, refinement and sampling. These sub-problems need to fulfill specific requirements to ensure that the J1 structure is maintained throughout the approximation process.

In the following subsection the notation used in the remainder of the paper is introduced. Then the problem formulation and the division into the sub-problems are presented. The requirements on the sub-problems are discussed and the solution approach for each sub-problem is outlined.

3.1. Notation and preliminaries

The objective is to approximate the continuous function $f(\boldsymbol{x}) : D \to \mathbb{R}$, $\boldsymbol{x} \in D \subset \mathbb{R}^n$ with a continuous PWL function $\hat{f}(\boldsymbol{x}) : D \to \mathbb{R}$. For a PWL function on a triangulation $\mathcal{T}$, each simplex $s \in S(\mathcal{T})$ is associated with a linear function $\hat{f}_s$. Using an indicator function $\mathbb{1}_s(\boldsymbol{x})$, which is 1 if $\boldsymbol{x} \in s$ and 0 otherwise, the PWL function can be written as

$$\hat{f}(\boldsymbol{x}) = \sum_{s\in S(\mathcal{T})} \hat{f}_s(\boldsymbol{x})\mathbb{1}_s(\boldsymbol{x}) \tag{1a}$$

where

$$\hat{f}_s(\boldsymbol{x}) = \boldsymbol{f}_s^\top M_s^{-1} \begin{bmatrix} \boldsymbol{x} \\ 1 \end{bmatrix} \tag{1b}$$

$$\boldsymbol{f}_s = [f_{v_1}, \dots, f_{v_{n+1}}]^\top \tag{1c}$$

$$M_s = \begin{bmatrix} \boldsymbol{x}_{v_1} & \cdots & \boldsymbol{x}_{v_{n+1}} \\ 1 & \cdots & 1 \end{bmatrix} \tag{1d}$$

$$\{v_1, \dots, v_{n+1}\} = V(s). \tag{1e}$$

$f_v \in \mathbb{R} : v \in V(\mathcal{T})$ are the function values at the vertices, $\boldsymbol{x}_v \in \mathbb{R}^n$ are the vertex coordinates and $V(\cdot)$ gives the vertices of a simplex/triangulation/grid.

For the piecewise linear function to be uniquely defined on the domain D, the triangulation $\mathcal{T}$ has to cover the domain and simplices $S(\mathcal{T})$ must not overlap:

$$\bigcup_{s\in S(\mathcal{T})} s \supseteq D \tag{2}$$

$$\text{vol}_d(s \cap s') = 0 \quad \forall s, s' \in S(\mathcal{T}), s \neq s'. \tag{3}$$

vol_d is the d-dimensional volume of the intersection of two simplices. I.e. in three dimensions simplices may share a face, an edge or a vertex but no volume; in two dimensions they may share an edge or a vertex but no area. If these conditions are met, the triangulation is referred to as valid.

Since this paper focuses on J1 compatible triangulations, triangulations on a grid $\mathcal{G}(s_1 \times \cdots \times s_n)$ with s_i segments in each dimension are considered. Such a grid has a total of $\prod_i (s_i + 1)$ vertices, whose coordinates are denoted $\boldsymbol{x}_v \in \mathbb{R}^n, v \in V(\mathcal{G})$ and the corresponding function values $f_v \in \mathbb{R}$. A J1 triangulation of vertices on a grid is written as $\mathcal{T} = J_1^\alpha(\mathcal{G})$ where $\alpha \in \{1,\dots,2^{n-1}\}$ specifies the J1 scheme, i.e. the orientation of the triangulation. Finally, the number of linear pieces is $|\mathcal{T}| = n! \prod_i s_i$.

A data set of sample points and the corresponding function values is denoted $\{\mathbf{x}_{\mathrm{D},d}, f_{\mathrm{D},d}\}$ with $d \in \{1,\dots,N_{\mathrm{D}}\}$.

3.2. Problem formulation

The full approximation problem can be stated as: Find the PWL approximation with the fewest number of linear pieces that meets the error tolerance; the triangulation must be a valid J1 triangulation. I.e.

$$\min\ |\mathcal{T}| = n! \prod_{i=1}^{n} s_i \tag{4a}$$

$$\text{s.t.}\ \|\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x})\| \le \text{tol} \tag{4b}$$

$$\mathcal{T} = J_1^\alpha(\mathcal{G}),\ \mathcal{G} = \mathcal{G}(s_1 \times \cdots \times s_n) \tag{4c}$$

$$\bigcup_{s \in S(\mathcal{T})} s \supseteq D \tag{4d}$$

$$\mathrm{vol}_d(s \cap s') = 0 \quad \forall s, s' \in S(\mathcal{T}), s \neq s' \tag{4e}$$

$$f_v \in \mathbb{R}, \boldsymbol{x}_v \in \mathbb{R}^n \quad \forall v \in V(\mathcal{G}) \tag{4f}$$

$$\alpha \in \{1,\dots,2^{n-1}\},\ s_1,\dots,s_n \in \mathbb{N} \tag{4g}$$

The constraint Eq. (4b) specifies the error tolerance. Typically, the error is defined as the maximum absolute point-wise deviation. Eq. (4c) specifies that the triangulation must be a J1 triangulation and the following two constraints are the requirements that the PWL function is uniquely defined on D.

To find the optimal PWL approximation of the function, the algorithm has to determine

1. the number of grid segments in each dimension $s_1,\dots,s_n$,
2. the J1 scheme α,
3. the position of the vertices $\boldsymbol{x}_v$ and
4. the function values at the vertices f_v.

Solving this problem directly is currently intractable, so it is divided into three sub-problems (fitting, refinement and sampling), which are solved iteratively.

For the fitting sub-problem, the grid $\mathcal{G}(s_1 \times \cdots \times s_n)$ is fixed. Consequently, the number of linear pieces is fixed and the objective of the fitting problem becomes to minimize the error of the approximation:

$$\min\ \|\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x})\| \tag{5a}$$

$$\text{s.t.}\ \mathcal{T} = J_1^\alpha(\mathcal{G}) \tag{5b}$$

$$\bigcup_{s \in S(\mathcal{T})} s \supseteq D \tag{5c}$$

$$\mathrm{vol}_d(s \cap s') = 0 \quad \forall s, s' \in S(\mathcal{T}), s \neq s' \tag{5d}$$

$$\mathrm{vol}_d(s \cap s') = 0 \quad \forall s, s' \in S(J_1^\beta(\mathcal{G})), s \neq s', \forall \beta \in \{1,\dots,2^{n-1}\} \setminus \alpha \tag{5e}$$

$$f_v \in \mathbb{R}, \boldsymbol{x}_v \in \mathbb{R}^n \quad \forall v \in V(\mathcal{G}) \tag{5f}$$

$$\alpha \in \{1,\dots,2^{n-1}\} \tag{5g}$$

The new constraint in Eq. (5e) is required to ensure that the J1 structure of the triangulation is maintained in the refinement sub-problem. It prevents situations like the one illustrated in Fig. 2, where the triangulation could become invalid when adding a grid segment. The constraint specifies that the grid must not only be valid for the J1 scheme of the approximation α, but also for all other possible J1 schemes β. Thus, regardless of how the grid is modified in the refinement sub-problem, the result is a valid J1 triangulation.

The grid needs to be refined, if the solution of the fitting sub-problem does not satisfy the error tolerance, i.e. if it is not possible to meet the error tolerance with the given grid $\mathcal{G}$. In general, additional segments will be added to the grid so that the function can be approximated more accurately. The decision of how and where to add the new segments is referred to as the refinement sub-problem.

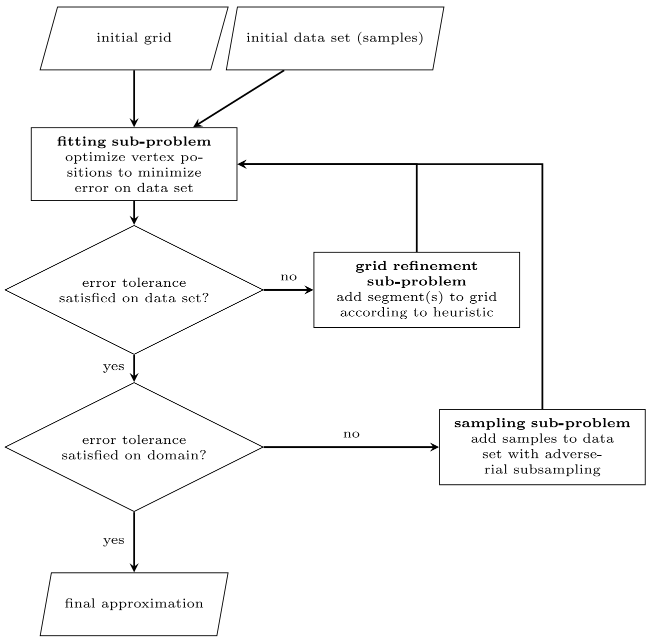
initial grid
initial data set (samples)
fitting sub-problem
optimize vertex positions to minimize error on data set
error tolerance satisfied on data set?
no
grid refinement sub-problem
add segment(s) to grid according to heuristic
yes
error tolerance satisfied on domain?
no
sampling sub-problem
add samples to data set with adversarial subsampling
yes
final approximation

Fig. 3. Flow chart of the piecewise linear approximation algorithm.

The final issue that needs to be addressed is that computing the error measure, the maximum absolute point-wise deviation between the function and its approximation, is a challenging optimization problem in itself (Kazda and Li, 2021). To arrive at a tractable problem, the error measure $\|\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x})\|$ for continuous $\boldsymbol{x} \in D$ is replaced by the error on a set of sample points $\|\hat{f}(\boldsymbol{x}_{\mathrm{D},d}) - f_{\mathrm{D},d}\|$, where $\boldsymbol{x}_{\mathrm{D},d} \in D$, $f_{\mathrm{D},d} = f(\boldsymbol{x}_{\mathrm{D},d}), d \in \{1,\dots,N_{\mathrm{Data}}\}$. This allows to compute the error efficiently, but it requires a sampling algorithm that selects the points in the data set in a way that the specified error tolerance is satisfied on the whole domain D. This is referred to as the sampling sub-problem.

The algorithm that is proposed to solve the full problem in Eq. (4) is illustrated as a flow chart in Fig. 3. The starting point for the algorithm is an initial grid (usually one with only one single segment) and an initial data set containing some samples (e.g. at the position of the vertices of the initial grid). Then the sub-problems are solved iteratively until the specified error tolerance is satisfied. The additional constraints in the fitting subproblem ensure that the J1 structure is maintained throughout the approximation process.

For the fitting sub-problem, a solution approach with a gradient-based optimization is introduced in Section 3.3. For the refinement sub-problem, a heuristic is proposed in Section 3.4. For the sampling sub-problem, the adversarial sub-sampling algorithm by Kazda and Li (2021) is used.

3.3. Fitting sub-problem

The main factor that makes the fitting sub-problem in Eq. (5) challenging is that the number of variables (vertex positions and function values at the vertices) increases steeply with the grid size. Though, besides the J1 scheme α, all variables are continuous. Consequently, for a given J1 scheme α (which will be assumed in the remainder of this section), the fitting sub-problem is a continuous non-linear optimization problem, which lends itself to gradient-based optimization.

A challenge with gradient based optimization is that by itself it is a local optimization method. As a consequence, the burden of converging towards a globally optimal solution of the full problem is put on the grid refinement algorithm. This will be addressed in Section 3.4. The main advantage of gradient-base optimization (and the reason, why it was chosen) is that it is very efficient in terms of solving time even for large problems. This addresses the challenge with the potentially large number of variables, but it requires the problem in Eq. (5) to be adapted. Specifically

1. the objective function needs to be modified to be differentiable,
2. the constraints in Eqs. (5c)–(5e) need to be reformulated and
3. the problem needs to be regularized.

The adapted fitting problem is shown in Eq. (8). In the remainder of this section, the adaptations are discussed in detail.

For gradient based optimization to be effective, the objective function needs to be differentiable. The norm in the objective function in Eq. (5) typically is the max-norm (the maximum absolute error), which is not differentiable. Thus, an approximation, the smooth-max is proposed: The p-norm $\|\cdot\|_p^q$ approaches the max-norm as $p \to \inf$ and $q = 1$. However, at the same time, the objective function also becomes less smooth, increasing the risk that the algorithm gets stuck in a local minimum. Initial testing showed, that $p = 10$ provides a good trade-off between a sufficiently smooth objective and effectively minimizing the maximum absolute error. q is a power exponent, which was included for generality. (For $p = q = 2$ Eq. (8) would be a least squares problem.)

To ensure that the J1 triangulation $J_1^\alpha(\mathcal{G})$ is a valid for the domain D, the problem needs to be constrained so that the triangulation always covers the domain (Eq. (5c)) and that no two simplices overlap (5d). The first set of constraints is realized by fixing the boundary vertices of the grid $V(\partial\mathcal{G})$ on the boundary of the domain ∂D. The second set of constraints is realized by requiring the signed volume of each simplex to be positive (assuming that the volume of simplices on an equally spaced grid is positive by convention). These constraints effectively require $J_1^\alpha(\mathcal{G})$ to be a tiling of D.

To account for Eq. (5e), also the simplex volumes of the complementary J1 triangulations are required to be positive. The constraints on the simplex volumes can be written as one set of constraints. Thus, the constraints Eqs. (5c)–(5e) are replaced by Eqs. (8c) and (8d).

The simplex volumes also offer a convenient way to regularize the problem to improve the solution quality. Initial testing revealed that the algorithm tends to get stuck in local minima, if it is allowed to make simplices arbitrarily small. To alleviate this issue, a regularization term is introduced, which penalizes simplices with small volume:

$$\mathcal{L}_\mathrm{V} = \frac{1}{|\mathcal{T}|}\sum_{s_\alpha}\left(\frac{\bar{V}}{\mathrm{vol}_d(s_\alpha)}\right)^2 + \frac{\lambda_\beta}{|\mathcal{T}|(2^{n-1}-1)}\sum_{s_\beta}\left(\frac{\bar{V}}{\mathrm{vol}_d(s_\beta)}\right)^2 \tag{6a}$$

where

$$\mathcal{T} = J_1^\alpha(\mathcal{G}), s_\alpha \in S(\mathcal{T}) \tag{6b}$$

$$s_\beta \in S(J_1^\beta(\mathcal{G})), \forall\beta \in \{1,\dots,2^{n-1}\}\setminus\alpha \tag{6c}$$

$$\bar{V} = \frac{1}{|\mathcal{T}|}\sum_{s\in S(\mathcal{T})}\mathrm{vol}_d(s) \tag{6d}$$

The first term penalizes small simplices in the triangulation used for the approximation. The second term does the same for the simplices of all complementary triangulations. Even though, the complementary triangulations do not affect the error of the approximation, it is essential to incorporate them, because they determine if the fitting sub-problem can be solved effectively after the next iteration of the refinement sub-problem.

Both terms are normalized with the average simplex volume $\bar{V}$, which is the same regardless of the J1 scheme, and the number of simplices. With this normalization both terms are equal to 1 on an equally spaced grid, where all simplices have equal size. The more distorted the grid is, the larger these terms become. If any simplex was approaching zero volume, the regularization term would tend to infinity.

The parameter λ_β controls the trade-off between the J1 scheme of the current triangulation α and the complementary ones β. Preliminary testing showed that the specific value of λ_β has only a small impact on the solution. Intuitively, the complementary triangulations should have a smaller importance than the current one, so λ_β is set to $\frac{1}{4}$, which works well.

The second proposed regularization term is a penalty on the gradient of each of the linear pieces:

$$\mathcal{L}_\nabla = \sum_{s\in S(\mathcal{T})}\sum_{j=1}^{n}\left(\frac{\Delta x_j}{\Delta f}\frac{\partial f_s}{\partial x_j}\right)^2. \tag{7}$$

The elements of the gradient are normalized with Δf and Δx_j, which are the typical scales of the function value and each of the coordinates respectively. This regularization term can be used to address the vanishing gradient problem, which can occur on small or unevenly distributed data sets.

The regularization terms are weighted with the factors λ_V and λ_∇ respectively. To make weighting the regularization terms against the error term more consistent, the error term is normalized with λ_E, which is set to the reciprocal value of the error term during the initialization of the fitting sub-problem.

This leads to the final formulation of the fitting sub-problem adapted for gradient based algorithms.

$$\min\ \lambda_E\|\hat{f}(\boldsymbol{x}_{\mathrm{D},d}) - f_{\mathrm{D},d}\|_p^q + \lambda_V\mathcal{L}_\mathrm{V} + \lambda_\nabla\mathcal{L}_\nabla \tag{8a}$$

$$\text{s.t. } \mathcal{T} = J_1^\alpha(\mathcal{G}) \tag{8b}$$

$$\boldsymbol{x}_{v'} \in \partial D \quad \forall v' \in V(\partial\mathcal{G}) \tag{8c}$$

$$\mathrm{vol}_d(s) \geq 0 \quad \forall s \in S(J_1^\beta(\boldsymbol{x}_v)), \forall\beta \in \{1,\dots,2^{n-1}\} \tag{8d}$$

$$f_v \in \mathbb{R}, \boldsymbol{x}_v \in \mathbb{R}^n \quad \forall v \in V(\mathcal{G}) \tag{8e}$$

The inputs are a data set of sample points $\{\boldsymbol{x}_{\mathrm{D},d}, f_{\mathrm{D},d}\}$, a grid $\mathcal{G}(s_1,\dots,s_n)$ and a J1 scheme for the approximation α. The outputs of the fitting sub-problem are the vertex coordinates $\boldsymbol{x}_v$ and the corresponding function values f_v that define the PWL approximation.

To solve the fitting sub-problem efficiently, the gradients of the objective function need to be computed analytically. For the sake of compactness, these formulae have been moved to the appendix.

Provided that the algorithm converges, the error of solution is the smallest error that can be achieved with the given grid.

3.4. Refinement sub-problem

If the error tolerance cannot be satisfied with a given grid, more linear pieces have to be added to the approximation. Since the J1 structure needs to be maintained, it is not possible to add a single additional piece. Rather, the smallest unit of change is one additional grid segment in one of the dimensions. Depending on the number of segments in the other dimensions, this adds a number of linear pieces. For fine grids, this number can be quite large. Consequently, choosing the right dimension to add a new segment is critical. Further, the position, where the new segment is added is also critical, since the proposed algorithm for the fitting sub-problem only performs local search. If the position of the new segment is far off the optimal position, the fitting sub-problem may not converge to the optimal solution.

The challenge is to add a grid segment at the position and in the dimension, which will ultimately result in the approximation with the fewest possible linear pieces. In lieu of a clear strategy for achieving this, a greedy heuristic is proposed:

1. Identify the grid cell that contains the point with the maximum absolute error.
2. Split that cell in half, by adding a segment in the dimension that yields the largest reduction in error.

This heuristic adds new linear pieces where they lead to the largest decrease in error, thus, putting the focus on meeting the specified error tolerance in the fewest possible number of iterations.

The major downside of this heuristic in terms of performance is that, to take a decision, it needs to solve the fitting sub-problem once for each dimension to determine the reduction of the error. Estimating the error reduction, without solving the fitting problem, is unreliable since splitting one segment in half not only affects the approximation in that segment, but also in all segments to one side due to change of the orientation of the triangulation (see Fig. 2). However, splitting will never result in an invalid triangulation, since this is accounted for in the fitting sub-problem. While solving the fitting problem repeatedly is not an issue for approximating functions with few independent variables, it may become limiting for higher dimensional approximation tasks.

3.5. Further insights and limitations

For the proposed solution approach to the fitting sub-problem, the J1 scheme α is predefined. Though, for approximations with only a few segments per dimension, it may be worth to also consider α as optimization variable. This could be done in the refinement sub-problem in the second step by considering each possible J1 scheme in addition to the dimensions for the split.

The constraints on the complementary triangulations and the corresponding regularization term are required for the algorithm to work. However, they may also increase the approximation error. In practice, once the algorithm has terminated, one could run the fitting sub-problem once more without the constraint Eq. (5e) and $\lambda_\beta = 0$ to reduce the approximation error further.

Initial testing showed, that the refinement heuristic may make sub-optimal decisions on very sparse data sets. In itself this may not be a big issue but, since only local optimization is done in the fitting sub-problem, the algorithm may converge to a local minimum and not recover from that. To avoid this, the algorithm should be initialized with a sufficiently large number of sample points distributed over the domain of the target function.

An interesting option for getting better results in the refinement sub-problem is to add two new segments at a time instead of just one. Since a property of the J1 triangulation is that it repeats periodically every two segments, adding two segments does not affect the orientation of the triangulation of the other parts of the grid. This may make the convergence of the full problem more stable, but it will generally not result in the approximation with the smallest possible number of simplices. Though, this may not be a big issue: Huchette and Vielma (2023) noted that the efficiency of the logarithmic branching formulation suffers, when the number of segments in each dimension is not a power of two. This suggests, that instead of adding a segment at each refinement step, the number of segments should actually be doubled. If the grid has at least two segments, doubling the number of segments requires adding a multiple of two.

Even though optimizing the vertex coordinates allows the grid to adapt to the underlying function quite well, it is still confined to this basic grid structure in the direction of each axis. This poses a challenge when approximating functions, whose main features are not aligned with any of the main axis. A solution could be to rotate the main axes so that they are aligned either as a pre-processing step or as a part of the approximation procedure.

The fitting sub-problem can be generalized to other triangulations besides J1, where the structure of the triangulation needs to be maintained while optimizing the vertices. For example, the algorithm could be adapted to find approximations of a specific structure for use with a zig-zag formulation (Huchette and Vielma, 2023). This would also allow to relax the constraints on the complementary triangulation and thus could lead to approximations with fewer linear pieces.

4. Evaluation

To evaluate the performance of the proposed algorithm, it is applied to the same two sets of benchmark functions that Kazda and Li (2021) used to evaluate their Difference of Convex (DC) approach. The first set are the nine functions that were proposed by Rebennack and Kallrath (2015a) in their paper on their bi-variate approximation method (which will be referred to as RK). The second set is the product of 3, 4 and 5 variables. While the first set of benchmark functions serves to evaluate the ability of the approximation methods to approximate a wide range of different topologies, the second set serves to assess the ability to approximate higher dimensional functions.

Table 1
2D Benchmark functions by Rebennack and Kallrath (2015a) and the regularization coefficients used for the J1 approach.

#	Function	Domain	λ_V	λ_∇
1	$y = x_1^2 - x_2^2$	$[0.5, 7.5] \times [0.5, 3.5]$	10^{-2}	0
2	$y = x_1^2 + x_2^2$	$[0.5, 7.5] \times [0.5, 3.5]$	10^{-3}	0
3	$y = x_1 \cdot x_2$	$[2, 8] \times [2, 4]$	10^{-1}	0
4	$y = x_1 \cdot \exp(-x_1^2 - x_2^2)$	$[0.5, 2] \times [0.5, 2]$	10^{-2}	0
5	$y = x_1 \cdot \sin(x_2)$	$[1, 4] \times [0.05, 3.1]$	10^{-1}	0
6	$y = x_2^2 \cdot \sin(x_1)/x_1$	$[1, 3] \times [1, 2]$	10^{-1}	0
7	$y = x_1 \cdot \sin(x_1) \cdot \sin(x_2)$	$[0.05, 3.1] \times [0.05, 3.1]$	10^{-2}	0
8	$y = (x_1^2 - x_2^2)^2$	$[1, 2] \times [1, 2]$	10^{-2}	0
9	$y = \exp(-10(x_1^2 - x_2^2)^2)$	$[1, 2] \times [1, 2]$	10^{-3}	0

The computations were performed using Matlab R2023b on a 128-core system (AMD EPYC 7702P) with 256 GB RAM. For each approximation the maximum absolute error δ, the number of grid segments in each dimension, the number of linear pieces n_S and the total time t are reported. To put these results into context, the number of linear pieces and computation times of the other approximation methods are reproduced: RK and DC for the first set of benchmark functions and only DC for the second set (since the RK approach is limited to bivariate functions). Note that the reported times for each approach have been realized on different computer systems and consequently the absolute values are not directly comparable. Finally, to gauge the performance that can be expected when including each of the approximations in a MILP problem, we report the number of binary ($\mathbb{Z}_2 = \{0, 1\}$) and continuous ($\mathbb{R}$) auxiliary variables that are introduced by the J1 and the DC approximation. For the J1 approximations we assume that the logarithmic branching convex combination (Log) formulation (Vielma and Nemhauser, 2011) is used. For the DC approximation we assume that the Multiple Choice (MC) formulation is used, which has been shown to yield the best performance for this approximation type (Birkelbach et al., 2024).

The first set of benchmark functions are the nine functions proposed by Rebennack and Kallrath (2015a). Information on the functions is listed in Table 1. The last two columns give the value of the regularization coefficients that were used with the J1 approximation algorithm. The algorithm was initialized with a 1×1 grid and a data set of 50×50 sample points on an equally spaced grid.

Each of the nine functions is approximated with five error tolerances (maximum absolute pointwise deviation). Fig. 4 shows the approximations with the J1 approach for the tightest error tolerance for each benchmark function. It illustrates how the J1 triangulation is deformed to fit the topology of the benchmark functions. Additionally, a 3D rendering of these results is provided in the appendix. The quantitative results are reported in Table 2.

Both the J1 and the RK approach use triangles as the basic geometric shape. The results show that on the tightest error tolerance the J1 approach outperforms the RK approach on all benchmark functions except for #8. This function has strong curvature on two opposite corners of the domain, which seems to be an obstacle for approximation on a grid, even if it is not axis parallel. Also at looser error tolerances, the J1 approach mostly finds approximations with fewer linear segments than the RK approach. Where the RK approach finds approximations with fewer segments, the difference is quite small (except for function #8).

The DC approach consistently produced approximations with fewer linear pieces than both the RK and the J1 approach. This is not surprising, considering that it uses polygons (polytopes) instead of triangles

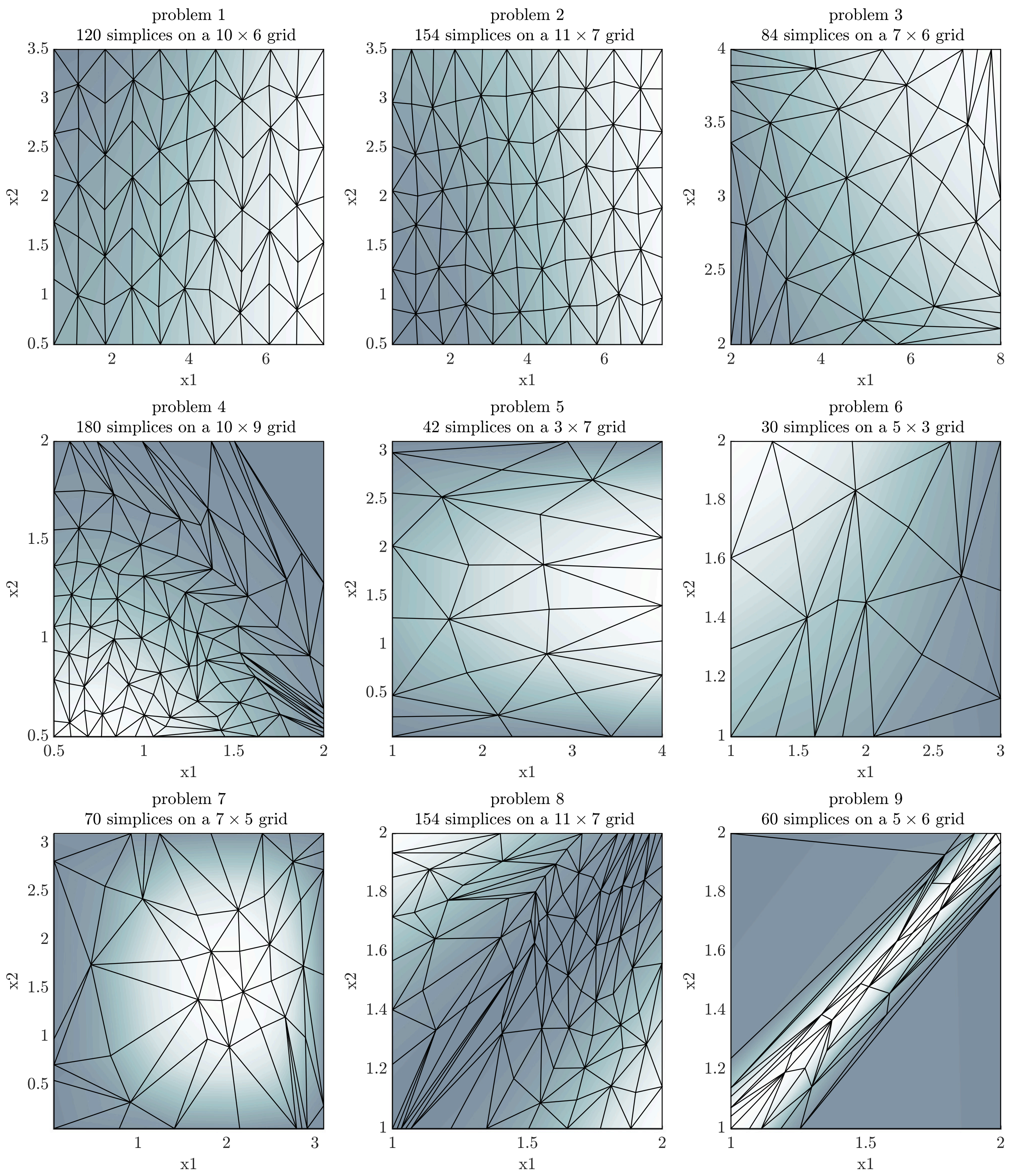

Fig. 4. Approximations of the benchmark functions at the lowest error tolerance with the J1 approach. The color shows the function value: lowest (dark) to highest (light).

(simplices) as the basic geometric shape. However, the J1 approach found an approximation for function #2 at a maximum error of 0.1, where the DC approach could not find any solution in the specified time limit (3600 s). For this function, the RK approach found a solution, but it requires 351 linear pieces compared to 154 with the J1 approach.

Predicting the MILP performance that can be expected with a given approximation is difficult as it depends on a variety of factors. A good indication is the number of auxiliary variables that are introduced in the MILP problem (Vielma et al., 2010; Kazda and Li, 2021). The smaller the formulation, the better the MILP performance. The values in last four columns in Table 2 show the number of binary ($\mathbb{Z}_2 = \{0, 1\}$) and continuous ($\mathbb{R}$) auxiliary variables for the J1 and the DC approach. For loose error tolerances, J1 and DC introduce a similar number of auxiliary variables. With tighter error tolerances, J1 approximations introduce considerably fewer auxiliary variables even if the number of linear pieces is higher. Even for function #8, where J1 required considerably more linear pieces, the MILP formulation is smaller for tight error tolerances. These results indicate that J1 approximations will

Table 2
Results on the first set of benchmark functions. Maximum absolute error δ, grid size, number of simplices n_S and time t. Values from reference publications RK (Rebennack and Kallrath, 2015a) and DC (Kazda and Li, 2023). Number of auxiliary variables in MILP problem.

#	$\delta_{\max}$	J1 approximation				RK		DC		MILP J1		MILP DC	
		δ	Grid	n_S	t	n_S	t	n_S	t	$\mathbb{Z}_2$	$\mathbb{R}$	$\mathbb{Z}_2$	$\mathbb{R}$
	1.500	1.070	3 × 2	12	1.1	16	30.8	6	0.3	4	12	6	12
	1.000	0.998	3 × 2	12	7.0	20	84.4	8	0.8	4	12	8	16
1	0.500	0.498	5 × 3	30	6.2	48	150.4	16	3.0	6	24	16	32
	0.250	0.246	7 × 4	56	17.2	80	272.6	32	23.0	6	40	32	64
	0.100	0.100	10 × 6	120	80.2	224	380.6	83	156.0	8	77	83	166
	1.500	1.258	3 × 2	12	1.3	24	26.8	6	30.0	4	12	6	12
	1.000	0.960	4 × 2	16	2.3	28	7.4	7	1.7	4	15	7	14
2	0.500	0.497	5 × 3	30	8.1	84	38.0	14	45.0	6	24	14	28
	0.250	0.248	7 × 4	56	20.6	121	35.8	25	361.0	6	40	25	50
	0.100	0.099	11 × 7	154	107.7	351	171.7	–	–	8	96	–	–
	1.000	0.523	2 × 2	8	0.7	4	0.8	4	0.1	3	9	4	8
	0.500	0.358	2 × 3	12	1.7	12	72.4	10	1.6	4	12	10	20
3	0.250	0.216	3 × 3	18	3.1	20	4.7	14	6.0	5	16	14	28
	0.100	0.100	4 × 5	40	12.4	59	59.3	30	46.0	6	30	30	60
	0.050	0.050	7 × 6	84	28.4	94	45.3	57	88.0	7	56	57	114
	0.100	0.066	1 × 1	2	0.4	2	0.3	1	0.0	1	4	1	2
	0.050	0.046	2 × 1	4	0.3	6	18.7	4	0.1	2	6	4	8
4	0.030	0.020	2 × 2	8	0.6	10	12.7	4	0.4	3	9	4	8
	0.010	0.007	4 × 3	24	3.8	31	54.6	18	6.2	5	20	18	36
	0.001	0.001	10 × 9	180	75.2	350	652.6	181	8418.0	9	110	181	362
	1.000	0.410	1 × 2	4	0.2	5	1.0	3	0.1	2	6	3	6
	0.500	0.410	1 × 2	4	0.3	8	13.1	3	0.1	2	6	3	6
5	0.250	0.232	2 × 3	12	1.1	16	30.0	8	0.3	4	12	8	16
	0.100	0.099	2 × 5	20	3.9	44	74.6	19	8.0	5	18	19	38
	0.050	0.050	3 × 7	42	11.3	85	141.9	27	37.0	6	32	27	54
	0.500	0.312	1 × 1	2	0.1	2	1.8	2	0.0	1	4	2	4
	0.250	0.207	1 × 2	4	0.3	4	1.0	4	0.1	2	6	4	8
6	0.100	0.071	3 × 2	12	1.4	9	25.8	6	0.7	4	12	6	12
	0.050	0.037	4 × 3	24	3.6	23	14.4	14	3.4	5	20	14	28
	0.030	0.029	5 × 3	30	5.6	40	161.4	26	12.0	6	24	26	52
	1.000	0.908	1 × 1	2	0.2	6	7.7	1	0.0	1	4	1	2
	0.500	0.256	2 × 2	8	0.7	6	1.3	4	0.5	3	9	4	8
7	0.250	0.250	2 × 2	8	0.9	21	30.8	6	1.1	3	9	6	12
	0.100	0.100	4 × 4	32	11.8	96	73.0	21	44.0	5	25	21	42
	0.050	0.049	7 × 5	70	17.0	274	305.5	44	130.0	7	48	44	88
	1.000	0.984	2 × 2	8	0.7	6	22.8	3	0.3	3	9	3	6
	0.500	0.492	2 × 3	12	2.0	9	15.6	4	0.4	4	12	4	8
8	0.250	0.247	4 × 3	24	3.7	12	22.9	10	11.0	5	20	10	20
	0.100	0.093	8 × 6	96	27.2	40	202.8	16	17.0	7	63	16	32
	0.050	0.049	11 × 7	154	72.7	87	174.1	38	80.0	8	96	38	76
	1.000	0.620	1 × 1	2	0.1	2	0.8	1	0.0	1	4	1	2
	0.500	0.410	1 × 1	2	0.2	4	66.6	2	0.1	1	4	2	4
9	0.250	0.080	3 × 3	18	2.2	6	4.4	12	2.9	5	16	12	24
	0.100	0.080	3 × 3	18	2.3	84	231.5	8	1.8	5	16	8	16
	0.050	0.046	5 × 6	60	22.4	86	57.8	14	8.0	7	42	14	28

likely yield faster MILP solving times than DC for tight error tolerances. For crude error tolerances, DC will likely yield better performance.

The second set of benchmark functions are the products of $n = 3$, 4 and 5 variables on the domain $[0, 1]^n$. The J1 algorithm, was again initialized with a grid with one segment in each dimension and a set of sample points from an equally spaced grid with 5 points in each dimension. The regularization parameters were set to $\lambda_V = 10^{-2}$ and $\lambda_\nabla = 0$.

The results are reported in Table 3 and an illustration of the results in 3 dimensions is shown in Fig. 5. The number of linear pieces that are required with the J1 approach increases steeply with the number of dimensions. The reason is that the J1 approach uses a grid to partition the domain into cells, which in turn are partitioned into simplices. Given an equal number of segments in each dimension, the number of cells grows exponentially with the number of dimensions n. In addition, the number of simplices in each hypercube grows as $n!$. I.e. partitioning a 3D cell requires 6 simplices, while partitioning a 5D cell requires 120. Due to this steep increase, which also affects the number of optimization variables, no solution was found within the time limit of 10 000 s for the tighter error tolerances in 4 and 5 dimensions. Even for the approximations that did satisfy the error tolerance, the number of linear pieces with the J1 approach was much larger than with the DC approach.

This curse of dimensionality, however, is less pronounced for the number of auxiliary variables in MILP since the J1 structure allows for a very efficient encoding of the simplices. In 3D, the number of auxiliary variables with the J1 approximations is about the same as for DC. At the tightest error tolerance the number of binary variables is even considerably lower. In 4D and 5D, J1 requires about the same number of binary variables, but it requires more continuous variables.

Unfortunately, the inherent scaling of triangulations limits the usefulness of the J1 approach for higher dimensional problems. Though, for 3 dimensional problems and tight error tolerances it still results in comparatively small MILP formulations.

5. Conclusion

In this paper, an algorithm to compute a piecewise-linear approximation of multi-variate functions for MILP was introduced. It uses a J1 triangulation on a deformed grid for the approximation. The

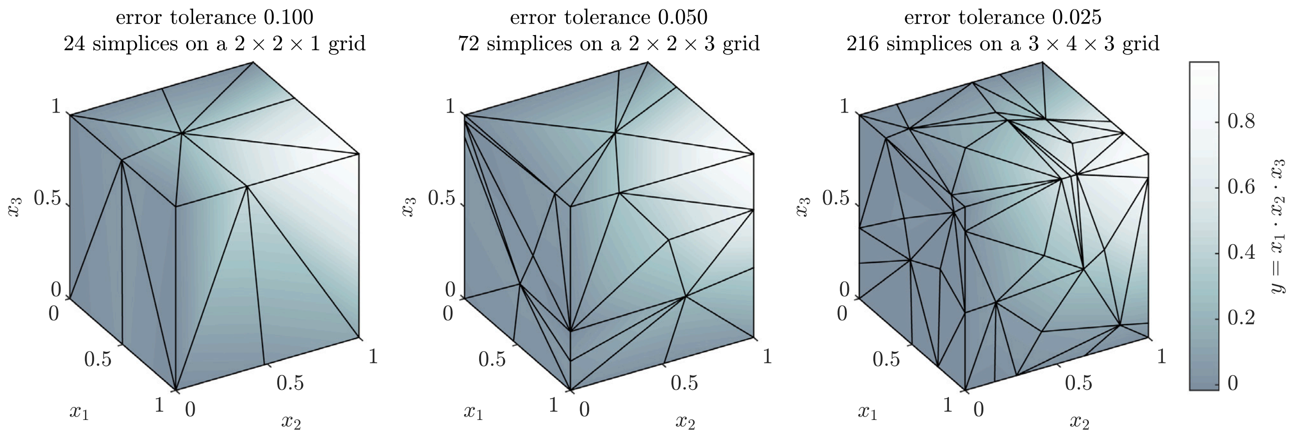

Fig. 5. Approximations of the product of three variables at three error levels.

Table 3
Results on the second set of benchmark functions. Maximum absolute error δ, grid size, number of simplices n and time t. Values from the reference publication DC (Kazda and Li, 2023). Number of auxiliary variables in MILP problem.

	$\delta_{\max}$	J1 approximation				DC		MILP J1		MILP DC	
		δ	Grid	n_S	t	n_S	t	$\mathbb{Z}_2$	$\mathbb{R}$	$\mathbb{Z}_2$	$\mathbb{R}$
3D	0.100	0.100	2 × 2 × 1	24	0.7	6	0.5	5	18	6	18
	0.050	0.049	2 × 2 × 3	72	7.8	11	6.2	7	36	11	33
	0.025	0.025	3 × 4 × 3	216	48.2	32	48.0	9	80	32	96
4D	0.100	0.097	2 × 2 × 1 × 2	192	518.0	8	2.2	9	54	8	32
	0.050	0.050	2 × 2 × 3 × 3	864	730.5	20	50.0	12	144	20	80
	0.025	–	–	–	–	60	1891.0	–	–	60	240
5D	0.100	0.099	2 × 2 × 2 × 2 × 1	1920	404.2	14	523.0	14	162	14	70
	0.050	–	–	–	–	26	526.0	–	–	26	130
	0.025	–	–	–	–	101	34 073.0	–	–	101	505

approximation task is formulated as an optimization problem, which is divided into three sub-problems to make it tractable: fitting, refining and sampling. The requirements on these sub-problems to maintain the J1 structure of the triangulation were discussed and a solution method for each sub-problem was proposed.

When approximating non-linear functions for MILP, the goal is to find approximations, which meet the error tolerance without compromising MILP solving times. Generally this requires finding approximations with only a few linear pieces and which introduce only a few auxiliary variables into the MILP problem. The main advantage of the J1 approach, compared to other piecewise-linear approximation methods from literature, is that the resulting piecewise-linear function is compatible with the logarithmic branching convex combination (Log) formulation, for which the number of auxiliary binary variables grows logarithmically with the number of linear pieces instead of linearly. This formulation has been shown to yield very fast MILP solving times, especially compared to non-logarithmic formulations. For this reason, the J1 approach is likely to yield fast MILP solving times.

The main alternative to the J1 approach is the Difference of Convex (DC) approach, which uses polytopes for the approximation (instead of a highly structured J1 triangulation). For this reason, DC generally produces piecewise linear functions with fewer linear pieces than the J1 approach. However, this does not necessarily result in smaller MILP formulations, since DC is confined to MILP formulations where the number of auxiliary variables grows linearly with the number of linear pieces (as opposed to logarithmically with the J1 approach).

To evaluate the performance of the proposed J1 approach, it was applied to a set of nine bi-variate benchmark functions. The results show that the J1 approach could approximate all functions at the required error tolerances with a reasonable number of linear pieces. Even though the number of linear pieces was generally larger than with the DC approach, the resulting MILP formulations still required about the same or, in many cases, considerably fewer auxiliary variables. This can be seen as an indicator for faster MILP solving times, particularly for approximations with tight error tolerances. However, computational studies to verify this are still required.

The J1 approach has also been tested on functions with up to 5 variables. Due to the inherent scaling of the J1 triangulation with the number of dimensions, these approximations become very large. In 4 and 5 dimensions, the algorithm was not able to find approximations that meet the tightest error tolerances. For 3D however it found approximations with smaller MILP formulations than the DC approach. These results suggest that the J1 approach is well suited for approximations in up to 3 dimensions. Even though it can technically approximate higher dimensional functions, the inherent scaling limits its usefulness.

Promising directions for future research include improving the algorithms performance for approximations in 4 and more dimensions as well as including the J1 scheme as an optimization variable. Further, the strict requirements on the triangulation could be relaxed by using a zig-zag instead of the Log formulation. This could allow to relax the constraints on the complementary triangulation in the fitting sub-problem and thus lead to approximations with fewer linear pieces.

Overall, the J1 approach was shown to be capable of approximating a wide range of bi-variate functions efficiently and also on 3 dimensional problems it produces good results. The resulting piecewise-linear function can be effectively translated to MILP using the logarithmic branching convex combination scheme, requiring fewer auxiliary variables than alternative approaches especially for tight error tolerances. Whether this results in significantly faster MILP solving times is yet to be verified and subject of ongoing research.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Felix Birkelbach reports financial support was provided by Austrian Research Promotion Agency.

Appendix A. 3D illustration of results

Figure Fig. A.6 shows the same results as Figure Fig. 4 but from an isometric perspective.

Appendix B. Gradients for the fitting sub-problem

To solve the fitting-sub-problem efficiently with gradient based algorithms, the gradient of the objective function in Eq. (8) with respect to the variables $\boldsymbol{x}_v$ and f_v has to be provided. For the sake of brevity, only the terms specific to the proposed algorithm are discussed here. Lengthy applications of the chain rule for well known functions such as the norm, reciprocals, etc... are omitted.

The following notation is used throughout this section: $S(\mathcal{T}, \boldsymbol{x})$ is the simplex (i.e. the linear piece) that contains the point $\boldsymbol{x}$. The vertices of a simplex are $V(s) = \{v_1, \dots, v_{n+1}\}$, $\boldsymbol{f}_s = [f_{v_1}, \dots, f_{v_{n+1}}]^\mathsf{T}$ is the vector of function values at the vertices and $\boldsymbol{M}_s = [\tilde{\boldsymbol{x}}_{v_1}, \dots, \tilde{\boldsymbol{x}}_{v_{n+1}}]$ is the characteristic matrix. A tilde denotes the augmented coordinate

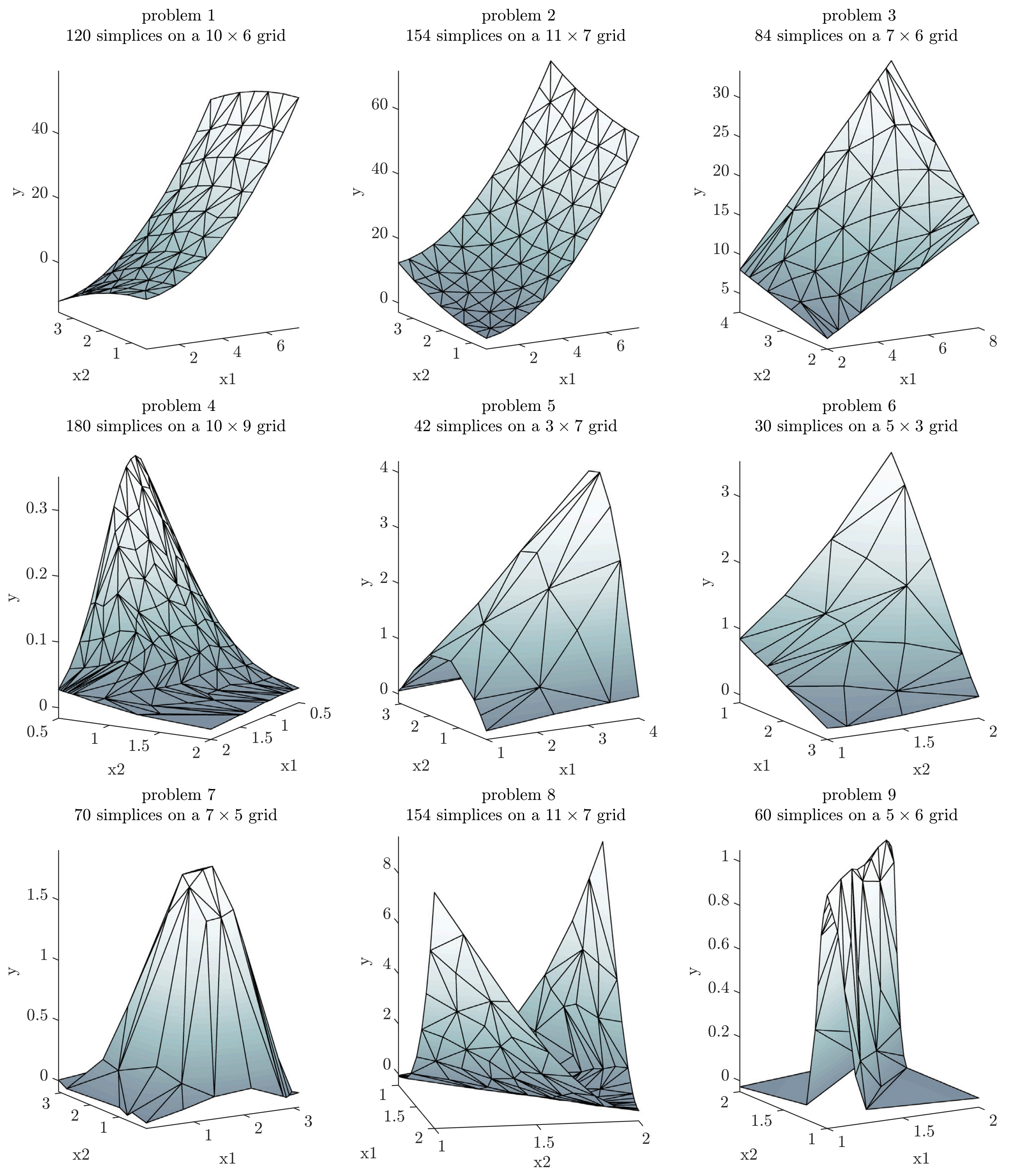

Fig. A.6. Approximations of the benchmark functions at the lowest error tolerance with the J1 approach. [3D version].

$\tilde{\boldsymbol{x}} = [\boldsymbol{x}^\top, 1]^\top$, $x_{v,i}$ is the coordinate in dimension i. $\boldsymbol{e}_i \in \mathbb{R}^{n+1}$ is the unit vector in dimension i and $\boldsymbol{e}_{v,s} \in \mathbb{R}^{n+1}$ is a vector, with 1 at the position that corresponds to the vertex v in $V(s)$. If v is not a vertex of the simplex s, $\boldsymbol{e}_{v,s}$ is all zeros. Consequently, many of the derivatives will be zero.

The value of the piecewise linear function and its derivatives are given by

$$\hat{f}(\boldsymbol{x}) = \boldsymbol{f}_s^\top M_s^{-1} \tilde{\boldsymbol{x}} \quad s = S(\mathcal{T}, \boldsymbol{x}) \tag{B.1a}$$

$$\frac{\partial}{\partial f_v} \hat{f}(\boldsymbol{x}) = \boldsymbol{e}_{v,s}^\top M_s^{-1} \tilde{\boldsymbol{x}} \quad s = S(\mathcal{T}, \boldsymbol{x}) \tag{B.1b}$$

$$\frac{\partial}{\partial x_{v,i}} \hat{f}(\boldsymbol{x}) = -\boldsymbol{f}_s^\top M_s^{-1} (\boldsymbol{e}_i \boldsymbol{e}_{v,s}^\top) M_s^{-1} \tilde{\boldsymbol{x}} \quad s = S(\mathcal{T}, \boldsymbol{x}) \tag{B.1c}$$

Note that the function value estimate $\hat{f}(\boldsymbol{x})$ is linear in the function values at the vertices f_v. Thus, determining the optimal function values at the vertices for fixed vertex coordinates is a linear problem. This can be leveraged to compute good starting values for the gradient descent algorithm.

The signed volume of a simplex s is given by

$$\text{vol}_d(s) = \frac{1}{n!} \det M_s \tag{B.2a}$$

$$\frac{\partial}{\partial f_v} \text{vol}_d(s) = 0 \tag{B.2b}$$

$$\frac{\partial}{\partial x_{v,i}} \text{vol}_d(s) = \boldsymbol{e}_{v,s}^\top M_s^{-1} \boldsymbol{e}_i \, \text{vol}(s) \tag{B.2c}$$

The sign of $\det M_s$ depends on the order of the vertices $V(s)$. The convention is that the order is such, that the determinant is positive for a simplex on an axis-parallel J1 grid.

The slope of the function on a simplex is given by

$$\frac{\partial \hat{f}_s}{\partial x_j} = \boldsymbol{f}_s^\top M_s^{-1} \boldsymbol{e}_j \tag{B.3a}$$

$$\frac{\partial}{\partial f_v} \frac{\partial \hat{f}_s}{\partial x_j} = \boldsymbol{e}_{v,s}^\top M_s^{-1} \boldsymbol{e}_j \tag{B.3b}$$

$$\frac{\partial}{\partial x_{v,i}} \frac{\partial \hat{f}_s}{\partial x_j} = -\boldsymbol{f}_s^\top M_s^{-1} (\boldsymbol{e}_i \boldsymbol{e}_{v,s}^\top) M_s^{-1} \boldsymbol{e}_j \tag{B.3c}$$

Data availability

A Matlab implementation is available at https://doi.org/10.5281/zenodo.14669069.

References

Birkelbach, F., Kasper, L., Schwarzmayr, P., Hofmann, R., 2023. Operation planning with thermal storage units using MILP: Comparison of heuristics for approximating non-linear operating behavior. In: Proceedings of ECOS 2023. Las Palmas De Gran Canaria, Spain, pp. 1345–1350. http://dx.doi.org/10.52202/069564-0122, URL: http://www.proceedings.com/069564-0122.html.

Birkelbach, F., Ruela, V.S.P., Hofmann, R., 2024. Performance of piecewise linear models in MILP unit commitment: Difference of convex vs. J1 approximation. In: Computer Aided Chemical Engineering. Vol. 53, Elsevier, pp. 115–120. http://dx.doi.org/10.1016/B978-0-443-28824-1.50020-X, URL: https://linkinghub.elsevier.com/retrieve/pii/B978044328824150020X.

Brito, B., Finardi, E., Takigawa, F., 2020. Mixed-integer nonseparable piecewise linear models for the hydropower production function in the unit commitment problem. Electr. Power Syst. Res. 182, 106234. http://dx.doi.org/10.1016/j.epsr.2020.106234, URL: https://linkinghub.elsevier.com/retrieve/pii/S0378779620300419.

Grimstad, B., Andersson, H., 2019. ReLU networks as surrogate models in mixed-integer linear programs. Comput. Chem. Eng. 131, 106580. http://dx.doi.org/10.1016/j.compchemeng.2019.106580, URL: https://linkinghub.elsevier.com/retrieve/pii/S0098135419307203.

Huchette, J., Vielma, J.P., 2019. A combinatorial approach for small and strong formulations of disjunctive constraints. Math. Oper. Res. 44 (3), 793–820. http://dx.doi.org/10.1287/moor.2018.0946, URL: https://pubsonline.informs.org/doi/10.1287/moor.2018.0946.

Huchette, J., Vielma, J.P., 2023. Nonconvex piecewise linear functions: Advanced formulations and simple modeling tools. Oper. Res. 71 (5), 1835–1856. http://dx.doi.org/10.1287/opre.2019.1973, URL: https://pubsonline.informs.org/doi/10.1287/opre.2019.1973.

Kämper, A., Holtwerth, A., Leenders, L., Bardow, A., 2021. AutoMoG 3D: Automated data-driven model generation of multi-energy systems using hinging hyperplanes. Front. Energy Res. 9, 719658. http://dx.doi.org/10.3389/fenrg.2021.719658, URL: https://www.frontiersin.org/articles/10.3389/fenrg.2021.719658/full.

Kazda, K., Li, X., 2021. Nonconvex multivariate piecewise-linear fitting using the difference-of-convex representation. Comput. Chem. Eng. 150, 107310. http://dx.doi.org/10.1016/j.compchemeng.2021.107310, URL: https://linkinghub.elsevier.com/retrieve/pii/S0098135421000880.

Kazda, K., Li, X., 2023. A linear programming approach to difference-of-convex piecewise linear approximation. European J. Oper. Res. S0377221723005647. http://dx.doi.org/10.1016/j.ejor.2023.07.026, URL: https://linkinghub.elsevier.com/retrieve/pii/S0377221723005647.

Kong, L., Maravelias, C.T., 2020. On the derivation of continuous piecewise linear approximating functions. INFORMS J. Comput. 32 (3), 531–546. http://dx.doi.org/10.1287/ijoc.2019.0949, URL: https://pubsonline.informs.org/doi/10.1287/ijoc.2019.0949.

Obermeier, A., Vollmer, N., Windmeier, C., Esche, E., Repke, J.-U., 2021. Generation of linear-based surrogate models from non-linear functional relationships for use in scheduling formulation. Comput. Chem. Eng. 146, 107203. http://dx.doi.org/10.1016/j.compchemeng.2020.107203, URL: https://linkinghub.elsevier.com/retrieve/pii/S0098135420312461.

Rebennack, S., Kallrath, J., 2015a. Continuous piecewise linear delta-approximations for bivariate and multivariate functions. J. Optim. Theory Appl. 167 (1), 102–117. http://dx.doi.org/10.1007/s10957-014-0688-2, URL: http://link.springer.com/10.1007/s10957-014-0688-2.

Rebennack, S., Kallrath, J., 2015b. Continuous piecewise linear delta-approximations for univariate functions: Computing minimal breakpoint systems. J. Optim. Theory Appl. 167 (2), 617–643. http://dx.doi.org/10.1007/s10957-014-0687-3, URL: http://link.springer.com/10.1007/s10957-014-0687-3.

Rebennack, S., Krasko, V., 2020. Piecewise linear function fitting via mixed-integer linear programming. INFORMS J. Comput. 32 (2), 507–530. http://dx.doi.org/10.1287/ijoc.2019.0890, URL: http://pubsonline.informs.org/doi/10.1287/ijoc.2019.0890.

Ruela, V., van Beurden, P., Hofmann, R., Birkelbach, F., 2024. A heuristic for fitting piecewise-linear models of bivariate functions with simplices and its application to an industrial use case. In: Proceedings of the EURO24. http://dx.doi.org/10.5281/ZENODO.12760505, URL: https://zenodo.org/doi/10.5281/zenodo.12760505.

Silva, T.L., Camponogara, E., 2014. A computational analysis of multidimensional piecewise-linear models with applications to oil production optimization. European J. Oper. Res. 232 (3), 630–642. http://dx.doi.org/10.1016/j.ejor.2013.07.040.

Todd, M., 1977. Union jack triangulations. In: Fixed Points: Algorithms and Applications. Academic Press, New York, pp. 315–336, Meeting Name: International Conference on Computing Fixed Points with Applications.

Toriello, A., Vielma, J.P., 2012. Fitting piecewise linear continuous functions. European J. Oper. Res. 219 (1), 86–95. http://dx.doi.org/10.1016/j.ejor.2011.12.030, URL: https://linkinghub.elsevier.com/retrieve/pii/S0377221711011246.

Vielma, J.P., 2015. Mixed integer linear programming formulation techniques. SIAM Rev. 57 (1), 3–57. http://dx.doi.org/10.1137/130915303, URL: http://epubs.siam.org/doi/10.1137/130915303.

Vielma, J.P., Ahmed, S., Nemhauser, G., 2010. Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions. Oper. Res. 58 (2), 303–315. http://dx.doi.org/10.1287/opre.1090.0721.

Vielma, J.P., Nemhauser, G.L., 2011. Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. Math. Program. 128 (1–2), 49–72. http://dx.doi.org/10.1007/s10107-009-0295-4.