Mixed Integer Programming for Change-point Detection

Apoorva Narula, Yao Xie, and Santanu S. Dey
H. Milton Stewart School of Industrial and Systems Engineering (ISyE),
Georgia Institute of Technology, USA
{anarula34, yao.xie, santanu.dey}@gatech.edu

Abstract

We propose a new mixed-integer programming formulation for fitting continuous piecewise linear functions. A key family of variables in this formulation is what we call as the segment assignment variables. These are indicator variables specifying which segment each data point belongs to. We prove that the projection of the linear programming relaxation of our formulation onto the segment assignment variables is integral. We compare our formulations against the most computationally efficient benchmarks in literature, both theoretically and through computational experiments on publicly available datasets. We observe that our approach achieves a significant speedup in terms of runtime relative to these benchmark formulations on larger datasets, and comparable performance on smaller ones.

1 Introduction and Literature Review

Given a dataset $\{(x_t, y_t)\}_{t=1}^T$ of T observations, we aim to fit a piecewise linear continuous function $\hat{y}(x)$ consisting of K segments while minimizing a chosen error metric between the fitted and the actual datapoints. The point of intersection of two consecutive piecewise linear segments is referred to as a break-point. This finds practical applications across various domains. In healthcare, piecewise-linear (PWL) models are used to fit local models of radiosensitivity with respect to genome-wide copy number variations, improving radiotherapy planning [Tobiasz et al., 2023]. In power systems, PWL functions are employed to approximate nonlinear power flow equations [Buason et al., 2025] and generator cost functions [Ahmadi et al., 2013]. In manufacturing, PWL fitting helps reformulate mixed-integer nonlinear programs into mixed-integer linear programs for process planning, such as in PVC plant optimization [Gao et al., 2018]. This has also been applied to evaluate the impact of policy and medical interventions over time [Wagner et al., 2002, Dehning et al., 2020]. Some classical approaches to obtain such a fitting are by Bellman and Roth [1969], Guthery [1974], Ertel and Fowlkes [1976], Bai and Perron [1998], Chen and Wang [2013] to name a few. These rely on dynamic programming and penalized likelihood methods to identify these breakpoint locations. These methods however are difficult to extend to high-dimensional settings and yield locally optimal solutions. With growing usability of mixed integer programs to large-scale optimization problems due to improved solver speeds, Bertsimas and Shioda [2007], in this work we formulate piecewise linear continuous function fitting as a mixed-integer programming (MIP) problem. Bertsimas and Shioda [2007] also introduced one of the earliest MIP formulations for fitting piecewise linear functions. However, their formulation does not enforce continuity of the fitted function.

In addition to the piecewise linear fitting problems, there also exists a substantial body of literature on developing MIP formulations for representing non-convex piecewise linear functions. Wolsey and Nemhauser [1999] propose MIP-based representations which were subsequently advanced in later works by Padberg [2000], Sherali [2001], Croxton et al. [2003]. Beyond representation, some studies have explored the use of piecewise linear (PWL) functions within optimization models. Vielma

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: MLxOR: Mathematical Foundations and Operational Integration of Machine Learning for Uncertainty-Aware Decision-Making.

et al. [2010] provide a framework for making MIP formulations for optimization problems involving piecewise linear functions. Rebennack [2016] discuss approximating nonlinear terms in mixed integer non-linear programs (MINLPs) using piecewise linear outer and inner approximations, yielding MILP reformulations with performance guarantees. PWL functions have also been employed to approximate known nonlinear functions. Lyu et al. [2025] explore using MIP to derive tight bounds on nonlinear functions through piecewise linear approximations. Rebennack and Kallrath [2015] consider the problem of fitting a piecewise linear, continuous functions which approximate a given non-convex function such that their maximum deviation from the target function does not exceed an error tolerance value.

The focus of this paper differs from the above problems, as we work on fitting piecewise linear continuous functions to a given data set where the function being approximated is unknown. This distinction is also made clear by Toriello and Vielma [2012], who propose MIP formulations for fitting continuous piecewise affine functions defined over a multi-dimensional input space. In their approach, the optimal affine function is evaluated by interpolation between computed values at fixed grid points. These fixed gridpoints are assumed to be known at prior, and are given as input to the model. Hence, finding the optimal affine function reduces to finding the optimal triangulation among the gridpoints to be used for interpolation. This makes it a combinatorial optimization problem and the optimal triangulation is determined as part of the MIP solution. While this formulation ensures global optimality for a fixed discretization provided as input, however it may not be necessary that these are always known beforehand. Cui et al. [2018] avoid explicit combinatorial partitioning by representing the fitted function as the difference of two convex piecewise-affine components, thereby defining the underlying polyhedral regions implicitly through the active affine terms. The resulting problem is nonconvex but continuous, and therefore nonconvex optimization techniques such as nonmonotone majorization–minimization and semismooth Newton method are used for optimization. However, this method does not guarantee global optimality. Rebennack and Krasko [2020] and Goldberg et al. [2021] solve the piecewise linear continuous function fitting problem and treat the breakpoints as decision variables to be optimized by the MIP. The continuity-enforcing constraints proposed by Goldberg et al. [2021] render their MIP formulation nonlinear, as they involve bilinear terms arising from products of integer and continuous variables. Rebennack and Krasko [2020] provide an alternate MIP formulation that employs additional binary variables and big-M constraints to linearize the model, yielding a purely mixed-integer linear program. A comprehensive comparative study by Warwicker and Rebennack [2022] reports that the formulation of Rebennack and Krasko [2020] remains the most computationally efficient among existing models, including that of Kong and Maravelias [2020]. Hence, in the rest of the study we compare our MIP model's performance against those by Rebennack and Krasko [2020] and Goldberg et al. [2021] as benchmarks.

The main contributions of this work is that we design a new MIP formulation for continuous piecewise linear fitting. We prove that the projection of the LP relaxation of our proposed MIP onto segment assignment variables is integral. We also perform comparative runtime analysis against existing MIP benchmarks and show a significant speed-up. The remainder of this workshop submission is organized as follows. In Section 2, we discuss the proposed MIP. In Section 3, we present computational experiments demonstrating the speedup of our formulation relative to the benchmarks. Section 4 provides a conclusion and outlines next steps for future research. For brevity, the main theoretical results are deferred to the Appendices A and B.

2 Methodology and Proposed MIP

The input to the model include the dataset $\{(x_t, y_t)\}_{t=1}^{T}$ consisting of T observations, the number of segments K, the Big-M constants $M_{1,t}^{A}$, $M_{2,t}^{A}$ and $M_{3,t}^{A}$, and the exponent $p \in \{1, 2\}$ over z_t, which allows the model to minimize either absolute error (when $p = 1$) or squared error (when $p = 2$). The decision variables include the binary variables $X_{j,t}$ and $\delta_{j,t}$, which are used to determine the segment assignment of the t^{th} data point. $\delta_{j,t} = 1$ indicates that the t^{th} data point is assigned to the j^{th} segment, where $t \in \{1, \ldots, T\}$ and $j \in \{1, \ldots, K\}$. $X_j = \{X_{j,t}\}_{t=1}^{T}$ is a monotonically decreasing T dimensional vector of binary variables, and the index where it switches from 1 to 0 marks the beginning of a new segment. The values of $\delta_{j,t}$ are determined from the $X_{j,t}$ variables by the constraint set A1. The binary variables $X_{j,t}$ are designed to be nested and monotonically decreasing across t, ensuring that data points are assigned to segments in a contiguous manner. For further clarity, the relationship between $X_{j,t}$ and $\delta_{j,t}$ is illustrated in Figure 1.

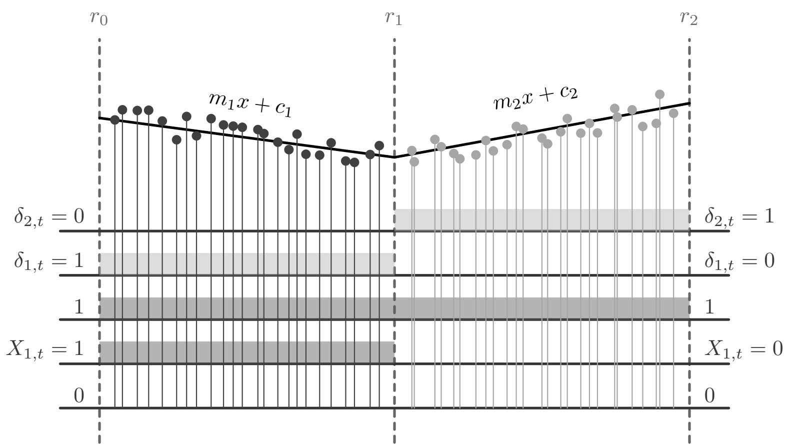

Figure 1: Relation between $X_{j,t}$ and $\delta_{j,t}$ in grayscale, shown using an illustrative example with $K = 2$.

Each segment $j \in \{1, \dots, K\}$ is defined by a linear function with slope m_j and intercept c_j, which are also decision variables. For every data point t, we introduce a decision variable $\hat{y}_t$ to represent the fitted value of x_t. To quantify the error, we define the decision variable z_t representing the absolute error for each data point. The formulation also includes $K + 1$ breakpoints as decision variables, denoted by $\{r_j\}_{j=0}^{K+1}$, which define the boundaries of the segments.

Objective function:

$$\min \sum_{t=1}^{T} z_t^p$$

Constraints:

(A1) **Segment Assignment Constraints:** Defining the segment assignment variables $\delta_{j,t}$ in terms of the nested binary variables $X_{j,t}$.

$$\delta_{j,t} = \begin{cases} X_{1,t}, & \text{if } j = 1 \\ 1 - X_{K-1,t}, & \text{if } j = K \\ X_{j,t} - X_{j-1,t}, & \text{if } j \in \{2, \dots, K-1\} \end{cases} \quad \forall j \in \{1, \dots, K\}, \forall t \in \{1, \dots, T\} \tag{A1.1}$$

$$X_{j,t} \geq X_{j,t+1} \quad \forall j \in \{1, \dots, K-1\}, \forall t \in \{1, \dots, T-1\} \tag{A1.2}$$

$$X_{j+1,t} \geq X_{j,t} \quad \forall j \in \{1, \dots, K-1\}, \forall t \in \{1, \dots, T\} \tag{A1.3}$$

(A2) **Value Assignment:**

$$m_j x_t + c_j \leq \hat{y}_t + M_{1,t}^A (1 - \delta_{j,t}) \quad \forall j \in \{1, \dots, K\}, \forall t \in \{1, \dots, T\} \tag{A2.1}$$

$$m_j x_t + c_j \geq \hat{y}_t - M_{1,t}^A (1 - \delta_{j,t}) \quad \forall j \in \{1, \dots, K\}, \forall t \in \{1, \dots, T\} \tag{A2.2}$$

(A3) **Breakpoint Localization:** All the points assigned to a segment must lie between the the starting and ending breakpoint of that segment. These constraints are similar to benchmark formulation of Goldberg et al. [2014, 2021].

$$x_t \leq r_{j+1} + M_{2,t}^A (1 - \delta_{j,t}) \quad \forall j \in \{1, \dots, K\}, \forall t \in \{1, \dots, T\} \tag{A3.1}$$

$$x_t \geq r_j - M_{3,t}^A (1 - \delta_{j,t}) \quad \forall j \in \{1, \dots, K\}, \forall t \in \{1, \dots, T\} \tag{A3.2}$$

$$r_0 = x_0, \quad r_{K+1} = x_T \tag{A3.3}$$

(A4) **Continuity:** The value of consecutive segments must be the same at the breakpoint between them, as in Goldberg et al. [2014, 2021].

$$m_{j-1} r_j + c_{j-1} = m_j r_j + c_j \quad \forall j \in \{2, \dots, K\} \tag{A4.1}$$

(A5) **Absolute Error:** Decision variable z_t captures the absolute error between y_t and the estimated $\hat{y}_t$.

$$\begin{aligned} z_t &\geq y_t - \hat{y}_t \\ z_t &\geq \hat{y}_t - y_t \end{aligned} \quad \forall t \in \{1, \ldots, T\} \tag{A5.1}$$

(A6) **Bounding Decision Variables:** Computing bounds on the decision variables is useful for computing the Big-M parameters $M^A_{1,t}$ and $M^A_{2,t}$. The bounds imposed on the slope and intercept variables, which are further discussed in Appendix A.1.

$$m_j \in [L^m, U^m], \quad c_j \in [L^c, U^c] \quad \forall j \in \{1, \ldots, K\}$$

3 Benchmarking Experiments

We compare the runtime of our MIP formulation against benchmark formulations proposed by Goldberg et al. [2021] (Basic formulation) and Rebennack and Krasko [2020] (Alternate formulation) through computational experiments on common datasets. All optimization experiments were conducted on an Apple M2 Pro processor with 12 cores. The MIP solver used is Gurobi Optimizer, version 11.0.1 (build v11.0.1rc0, mac64[arm], Darwin 24.4.0 24E263). The datasets used correspond to the historical daily closing prices of Apple Inc. (AAPL) stock. While our method has been benchmarked on additional datasets, we present a subset of results for brevity. The stock price data was obtained from the `yfinance` Python package, which has been used to download financial data from Yahoo Finance, starting from the date of January 1, 2022. The runtime results comparisons are shown in Table 1. We observe that the runtime of our MIP formulation is comparable to the benchmark formulations for smaller problem instances, i.e., when the dataset size or the number of segments to be detected is small and becomes faster for larger ones.

Table 1: Minimizing Mean Absolute Error (seconds) by formulation, and K shown in columns; rows show T.

T	**Basic**				**Alternate**				**Proposed**			
	K=2	K=3	K=4	K=5	K=2	K=3	K=4	K=5	K=2	K=3	K=4	K=5
100	0.24	0.56	2.64	7.16	**0.13**	**0.50**	3.21	8.83	0.29	0.63	**2.18**	**4.17**
200	0.46	**1.25**	15.65	1119.99	**0.41**	1.70	15.42	120.57	0.73	2.13	**13.57**	**42.29**
300	0.76	**7.80**	384.99	2000*	**0.59**	15.12	85.07	1339.09	1.17	9.22	**44.22**	**709.35**
400	1.32	18.63	269.97	2000*	**1.03**	23.36	91.56	2000*	1.43	**11.53**	**60.20**	2000*
500	1.54	11.59	2000*	2000*	**1.46**	13.06	**159.16**	2000*	1.85	**9.67**	195.77	2000*

* Solver reached the time limit (reported as $\approx$ 2000 s).

4 Conclusion and Future Research

We have proposed a new formulation that exhibits both strong theoretical properties and computational performance compared to the state-of-the-art in existing literature. In particular, for larger datasets our approach achieves a significant speedup relative to existing methods, and comparable performance on smaller ones. Future research directions include extending the formulation to address online variants of the problem, as well as solving the problem in the multidimensional setting.

Acknowledgments and Disclosure of Funding

The work of Y.X. and A.N. is partially supported by DMS-2220495, ONR N000142412278, and the Coca-Cola Foundation.

References

Hamed Ahmadi, José R Martí, and Ali Moshref. Piecewise linear approximation of generators cost functions using max-affine functions. In *2013 IEEE Power & Energy Society General Meeting*, pages 1–5. IEEE, 2013.

Jushan Bai and Pierre Perron. Estimating and testing linear models with multiple structural changes. *Econometrica*, pages 47–78, 1998.

Richard Bellman and Robert Roth. Curve fitting by segmented straight lines. *Journal of the American Statistical Association*, 64(327):1079–1084, 1969.

Dimitris Bertsimas and Romy Shioda. Classification and regression via integer optimization. *Operations research*, 55(2):252–271, 2007.

Paprapee Buason, Sidhant Misra, and Daniel K Molzahn. Sample-based piecewise linear power flow approximations using second-order sensitivities. *arXiv preprint arXiv:2501.13825*, 2025.

Danny Z Chen and Haitao Wang. Approximating points by a piecewise linear function. *Algorithmica*, 66(3):682–713, 2013.

Keely L Croxton, Bernard Gendron, and Thomas L Magnanti. A comparison of mixed-integer programming models for nonconvex piecewise linear cost minimization problems. *Management Science*, 49(9):1268–1273, 2003.

Ying Cui, Jong-Shi Pang, and Bodhisattva Sen. Composite difference-max programs for modern statistical estimation problems. *SIAM Journal on Optimization*, 28(4):3344–3374, 2018.

Jonas Dehning, Johannes Zierenberg, F Paul Spitzner, Michael Wibral, Joao Pinheiro Neto, Michael Wilczek, and Viola Priesemann. Inferring change points in the spread of covid-19 reveals the effectiveness of interventions. *Science*, 369(6500):eabb9789, 2020.

James E Ertel and Edward B Fowlkes. Some algorithms for linear spline and piecewise multiple linear regression. *Journal of the American Statistical Association*, 71(355):640–648, 1976.

Xiaoyong Gao, Zhenhui Feng, Yuhong Wang, Xiaolin Huang, Dexian Huang, Tao Chen, and Xue Lian. Piecewise linear approximation based milp method for pvc plant planning optimization. *Industrial & Engineering Chemistry Research*, 57(4):1233–1244, 2018.

Noam Goldberg, Youngdae Kim, Sven Leyffer, and Thomas D Veselka. Adaptively refined dynamic program for linear spline regression. *Computational Optimization and Applications*, 58(3):523–541, 2014.

Noam Goldberg, Steffen Rebennack, Youngdae Kim, Vitaliy Krasko, and Sven Leyffer. Minlp formulations for continuous piecewise linear function fitting. *Computational Optimization and Applications*, 79(1):223–233, 2021.

Scott B Guthery. Partition regression. *Journal of the American Statistical Association*, 69(348): 945–947, 1974.

Lingxun Kong and Christos T Maravelias. On the derivation of continuous piecewise linear approximating functions. *INFORMS Journal on Computing*, 32(3):531–546, 2020.

Bochuan Lyu, Illya V Hicks, and Joey Huchette. Building formulations for piecewise linear relaxations of nonlinear functions. *Operations Research*, 2025.

Manfred Padberg. Approximating separable nonlinear functions via mixed zero-one programs. *Operations Research Letters*, 27(1):1–5, 2000.

Steffen Rebennack. Computing tight bounds via piecewise linear functions through the example of circle cutting problems. *Mathematical Methods of Operations Research*, 84(1):3–57, 2016.

Steffen Rebennack and Josef Kallrath. Continuous piecewise linear delta-approximations for univariate functions: computing minimal breakpoint systems. *Journal of Optimization Theory and Applications*, 167(2):617–643, 2015.

Steffen Rebennack and Vitaliy Krasko. Piecewise linear function fitting via mixed-integer linear programming. *INFORMS Journal on Computing*, 32(2):507–530, 2020.

Hanif D Sherali. On mixed-integer zero-one representations for separable lower-semicontinuous piecewise-linear functions. *Operations Research Letters*, 28(4):155–160, 2001.

Joanna Tobiasz, Najla Al-Harbi, Sara Bin Judia, Salma Majid Wakil, Joanna Polanska, and Ghazi Alsbeih. Multivariate piecewise linear regression model to predict radiosensitivity using the association with the genome-wide copy number variation. *Frontiers in Oncology*, 13:1154222, 2023.

Alejandro Toriello and Juan Pablo Vielma. Fitting piecewise linear continuous functions. *European Journal of Operational Research*, 219(1):86–95, 2012.

Juan Pablo Vielma, Shabbir Ahmed, and George Nemhauser. Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions. *Operations research*, 58(2): 303–315, 2010.

Anita K Wagner, Stephen B Soumerai, Fang Zhang, and Dennis Ross-Degnan. Segmented regression analysis of interrupted time series studies in medication use research. *Journal of clinical pharmacy and therapeutics*, 27(4):299–309, 2002.

John Alasdair Warwicker and Steffen Rebennack. A comparison of two mixed-integer linear programs for piecewise linear function fitting. *INFORMS journal on computing*, 34(2):1042–1047, 2022.

Laurence A Wolsey and George L Nemhauser. *Integer and combinatorial optimization*. John Wiley & Sons, 1999.

A Appendix: Further Details for Proposed MIP

A.1 Choice of Big-M Constants

The Big-M analysis, as well as the slope and intercept bounds presented in this section, are similar to those discussed in Rebennack and Krasko [2020]. The slope for each linear segment can be bounded above (and below) by the maximum (and minimum) pairwise slope between any two distinct points in the data. Hence, we define:

$$L^m = \min_{\substack{t_1,t_2 \in \{1,\dots,T\} \\ t_1 \neq t_2}} \frac{y_{t_1} - y_{t_2}}{x_{t_1} - x_{t_2}}, \tag{A.1.1}$$

$$U^m = \max_{\substack{t_1,t_2 \in \{1,\dots,T\} \\ t_1 \neq t_2}} \frac{y_{t_1} - y_{t_2}}{x_{t_1} - x_{t_2}}. \tag{A.1.2}$$

Using these slope bounds, the intercept bounds can be computed as follows:

$$L^c = \min_{i \in \{1,\dots,K\}} \min\left(y_i - L^m x_i,\ y_i - U^m x_i\right), \tag{A.1.3}$$

$$U^c = \max_{i \in \{1,\dots,K\}} \max\left(y_i - L^m x_i,\ y_i - U^m x_i\right). \tag{A.1.4}$$

The parameter $M^A_{1,t}$ must satisfy

$$M^A_{1,t} \geq \max_j \left| m_j x_t + c_j - \hat{y}_t \right|. \tag{A.1.5}$$

Hence, we can set

$$M^A_{1,t} = \max\left(|L^m x_t| + |L^c|,\ |U^m x_t| + |L^c|,\ |L^m x_t| + |U^c|,\ |U^m x_t| + |U^c|\right). \tag{A.1.6}$$

The parameter $M_{2,t}^A$ must satisfy

$$M_{2,t}^A \geq \max_j \{r_{j+1} - x_t\} \quad \Rightarrow \quad M_{2,t}^A = x_T - x_t. \tag{A.1.7}$$

Similarly, the parameter $M_{3,t}^A$ must satisfy

$$M_{3,t}^A \geq \max_j \{x_t - r_j\} \quad \Rightarrow \quad M_{3,t}^A = x_t - x_0. \tag{A.1.8}$$

A.2 Total Unimodularity in $X_{j,t}$ and $\delta_{j,t}$ variables

A matrix $A \in \mathbb{R}^{m \times n}$ is *totally unimodular (TU)* if every square submatrix of A has determinant 0, 1, or -1. This property is useful in integer programming since if the constraint matrix of a linear program is TU and the right-hand side is integral, then every vertex of the feasible region is integral. Consequently, the optimal solution of the integer program coincides with that of its linear programming relaxation.

Theorem A.1 (Integrality of the $(A1)$ polyhedron)**.** *Let P be the polyhedron defined by the constraint set (A1) in the variables $\{X_{j,t}, \delta_{j,t}\}$ for all $t \in \{1, \ldots, T\}$ and $j \in \{1, \ldots, K\}$. Then P is integral.*

Proof. We have skipped the proof for the sake of brevity. □

B Appendix: Benchmark Mixed Integer Programming Formulations

We compare our MIP formulation with those proposed by Goldberg et al. [2014, 2021] and Rebennack and Krasko [2020]. These formulations appear to be the state-of-the-art based on comparisons as presented in Warwicker and Rebennack [2022] and Rebennack and Krasko [2020].

B.1 Basic Formulation: Goldberg et al. [2021]

The difference between our MIP and the one proposed by Goldberg et al. [2014, 2021] lies in the segment assignment constraints. The remaining components are consistent across both formulations.

(B1) **Segment Assignment Constraint:** The segment assignment variables $\delta_{j,t}$ must sum to 1 for each time step t, ensuring that each data point is assigned to a unique segment.

$$\sum_{j=1}^{K} \delta_{j,t} = 1 \qquad \forall t = 1, \ldots, T \tag{B1.1}$$

B.2 Alternate Formulation: Rebennack and Krasko [2020]

This formulation differs from our model in the segment assignment and continuity constraints. To avoid bilinear terms, it introduces auxiliary binary and continuous variables that enforce continuity of adjacent segments using slope-direction indicators. This modification allows adjacent lines to intersect in a manner that preserves continuity.

(C1) **Segment Assignment Constraints:** In addition to the basic assignment constraint, additional constraints to enforce segment contiguity are also imposed.

$$\sum_{j=1}^{K} \delta_{j,t} = 1 \qquad \forall t = 1, \ldots, T \tag{C1.1}$$

$$\delta_{j+1,t+1} \leq \delta_{j,t} + \delta_{j+1,t} \qquad \forall t = 1, \ldots, T-1; \quad j = 1, \ldots, K-1 \tag{C1.2}$$

$$\delta_{1,t+1} \leq \delta_{1,t} \qquad \forall t = 1, \ldots, T-1 \tag{C1.3}$$

$$\delta_{K,t+1} \geq \delta_{K,t} \qquad \forall t = 1, \ldots, T-1 \tag{C1.4}$$

(C2) **Continuity Constraints:** The formulation enforces continuity between segments using slope direction indicators. $\delta^+_{j,t}$ (or $\delta^-_{j,t}$) is greater than 0 if the slope is decreasing (or increasing) from one segment to another.

$$\delta_{j,t} + \delta_{j+1,t+1} + \gamma_j - 2 \leq \delta^+_{j,t} \tag{C2.1}$$

$$\delta_{j,t} + \delta_{j+1,t+1} + (1 - \gamma_j) - 2 \leq \delta^-_{j,t} \tag{C2.2}$$

Continuity Enforcement (via Big-M): Rather than explicitly modeling breakpoints r_b, the model ensures intersection between adjacent segments within a valid interval using:

$$c_{j+1} - c_j \geq x_t(m_j - m_{j+1}) - M^C_{4,t}(1 - \delta^+_{j,t}) \tag{C2.3}$$

$$c_{j+1} - c_j \leq x_{t+1}(m_j - m_{j+1}) + M^C_{4,t+1}(1 - \delta^+_{j,t}) \tag{C2.4}$$

$$c_{j+1} - c_j \leq x_t(m_j - m_{j+1}) + M^C_{4,t}(1 - \delta^-_{j,t}) \tag{C2.5}$$

$$c_{j+1} - c_j \geq x_{t+1}(m_j - m_{j+1}) - M^C_{4,t+1}(1 - \delta^-_{j,t}) \tag{C2.6}$$

Additional Variables include $\gamma_j \in \{0, 1\}$ which indicates whether the slope is decreasing ($\gamma_j = 1$) or increasing ($\gamma_j = 0$) between segments j and $j+1$, $\delta^+_{j,t}, \delta^-_{j,t} \in [0, 1]$ which are continuous variables enforcing continuity based on slope direction.

Additional Big-M parameters The parameter $M^C_{4,t}$ can be computed as follows. From the model constraints, we have

$$c_{j+1} - c_j \;\geq\; x_t(m_j - m_{j+1}) - M^C_{4,t} \quad \forall j \in \{1, \ldots, K\}. \tag{B.2.1}$$

Equivalently,

$$M^C_{4,t} \;\geq\; x_t(m_j - m_{j+1}) - c_{j+1} + c_j. \tag{B.2.2}$$

Hence, we can set

$$M^C_{4,t} = |x_t\,(U^m - L^m)| + |U^c - L^c|\,. \tag{B.2.3}$$

B.3 Comments on Projection of the Linear Programming Relaxations

We establish that the projection of the feasible region of our LP relaxation onto the subspace defined by the segment assignment variables $\delta_{j,t}$ is a *proper subset* of the corresponding projection for the formulation presented by Goldberg et al. [2014, 2021].

Theorem B.1. *Let Q denote the polytope defined by the projection of the constraint sets (A3) and (B1) onto the space of δ-variables; this corresponds to the projection of Goldberg's formulation onto the segment assignment variables. Let P denote the polytope defined by the projection of the constraint set (A1) onto the space of δ-variables; this corresponds to the projection of our formulation onto the segment assignment variables. Then we have*

$$P \subsetneq Q.$$

Proof. We have skipped the proof for the sake of brevity. □

In contrast, the formulation by Rebennack and Krasko [2020] is incomparable to ours. This is because in our case it is possible that a segment has no points assigned, whereas this is impossible in Rebennack and Krasko [2020].

B.4 Variation of Total Absolute Error, Total Squared Error with K

Theorem B.2. *The total absolute error and total squared error of the piecewise linear fitting monotonically decrease with the upper bound on number of components, K*

Proof. We have skipped the proof for the sake of brevity. □