Optimization and Engineering
https://doi.org/10.1007/s11081-024-09888-2

RESEARCH ARTICLE

Novel solution strategies for multiparametric nonlinear optimization problems with convex objective function and linear constraints

Diogo A. C. Narciso[1,2,3] · Efstratios N. Pistikopoulos[4,5]

Received: 27 September 2023 / Revised: 28 March 2024 / Accepted: 29 March 2024

Abstract

This paper expands the multiparametric quadratic programming (mp-QP) framework presented in Narciso et al. (Comput Chem Eng 164:107882, 2022. https://doi.org/10.1016/j.compchemeng.2022.107882) to the more general multiparametric nonlinear programming (mp-NLP) case. First, the vector of parameters in mp-NLP problems is recast so that a unique transformed parameter is implicitly assigned to each of the inequality constraints. Maps of critical regions in this transformed space of parameters feature a set of 1-dimensional parametric edges (two per inequality constraint), which then greatly facilitate solution calculation. In the mp-NLP case, however, parametric edges define nonlinear semi-infinite lines; this requires an adaptation to the mp-QP algorithm (deals with linear parametric edges only), to enable a suitable calculation path to the more general nonlinear case. Three routes are proposed to mp-NLPs: the first route delivers solutions in compact form (same format as in mp-QP) using a single reference point per edge; the second route delivers explicit solutions using a hybrid approach for critical region construction, where all active sets not detected in the parameters space are excluded from the solution (equivalent to first route concerning accuracy); the third route builds on the initial explicit solution and further partitions

✉ Diogo A. C. Narciso
diogo.narciso@tecnico.ulisboa.pt

1 CERENA - Centro Recursos Naturais e Ambiente, Departmento de Engenharia Química, Instituto Superior Técnico, Universidade de Lisboa, Avenida Rovisco Pais 1, 1049-001 Lisboa, Portugal

2 LEPABE - Laboratory for Process Engineering, Environment, Biotechnology and Energy, Faculty of Engineering, University of Porto, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal

3 ALICE - Associate Laboratory for Innovation in Chemical Engineering, Faculty of Engineering, University of Porto, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal

4 Texas A&M Energy Institute, Texas A&M University, College Station, TX 77843, USA

5 Artie McFerrin Department of Chemical Engineering, Texas A&M University, College Station, TX 77843, USA

the parameters space until all solution fragments satisfy an error check. Five algorithms were coded for these routes, and tested in a large range of mp-NLP problems. These strategies enable significant improvements in terms of solution accuracy, algorithm efficiency, and interpretability when compared to the state-of-the-art mp-NLP algorithms.

Keywords Multiparametric programming · Nonlinear optimization · Critical regions · Explicit parametric solutions

1 Introduction

Multiparametric programming deals with a special class of optimization problems, where a vector of parameters is defined independently of the optimization variables and bounded in a real space of parameters. These parameters are typically posed as a set of right-hand side dependencies on the definition of the inequality constraints, enabling the contraction and expansion of the feasible space. These problems are inherently infinite in size, and their solutions are commonly delivered as a set of partitions of the parameters space (the critical regions) and their optimizer functions (Pistikopoulos et al. 2020).

In muliparametric linear and quadratic programming (mp-LP and mp-QP, respectively), optimizer functions are obtained exactly from the Karush–Kuhn–Tucker (KKT) conditions for all active sets (Gal and Nedoma 1972; Dua et al. 2002). The matching critical regions are then obtained via a set of additional optimality and feasibility constraints imposed on the solution of the KKT conditions. This solution strategy is generally *infeasible* in multiparametric nonlinear programming (mp-NLP): the KKT conditions in mp-NLP problems typically define a nonlinear system of equations, which cannot be solved explicitly for the independent vector of parameters. As a result, while in mp-LP and mp-QP the solution paradigm is based on the *exact* calculation of optimizer functions and critical regions per active set, in mp-NLP, *approximation-based* strategies rely on some pre-defined parameters space partitioning strategy.

In Dua and Pistikopoulos (1999), a multiparametric outer approximation (mp-OA) algorithm is proposed, which relies on local linearizations of the objective function to deliver the solutions of mp-NLPs via mp-LP. In Johansen (2002), a multiparametric quadratic approximation (mp-QA) algorithm is presented, iterating between the solutions of NLP and mp-QP problems. In Bemporad and Filippi (2006), an approximate multiparametric (AM) algorithm is proposed, where the parameters space is divided sequentially via a set of convex hulls, and optimizer functions are obtained from interpolation until a pre-defined tolerance is satisfied. A geometric vertex search (GVS) algorithm is presented in Narciso (2009), where critical regions are also delivered as convex hulls via a careful selection of vertices to improve solution accuracy and compactness. A review of the state-of-the-art mp-NLP algorithms is presented in Domínguez et al. (2010), where an improvement to the mp-QA algorithm is also discussed.

The primary goal of mp-NLP algorithms is to deliver optimizer functions such that for all combinations of parameters in the parameters space, the estimated optimizers match as closely as possible the exact optimizers (within a pre-defined tolerance). All algorithms above achieve this with distinct levels of success: this depends mainly on the strategy used to partition the parameters space, which may favor or hinder the calculation of solutions compactly and efficiently. It is noted in Narciso (2009) that the critical regions obtained from mp-NLP algorithms frequently bear little or no resemblance with the actual optimal active sets maps. These maps are often very complex and feature critical regions where no noticeable *patterns* can be identified, which makes the task of delivering the solutions of mp-NLP problems with the lowest complexity possible and preserving the actual optimal active set maps a challenging objective.

A new solution strategy for convex mp-QP was proposed in Narciso et al. (2022), which is particularly relevant to address the problem of how to partition the parameters space in mp-NLP problems: given an mp-QP problem including p inequality constraints, it is shown that the full map of critical regions may be described in a concise form via a set of $2p$ vectors of directions in $\mathbb{R}^p$, rather than up to a total of 2^p sets of parameters space bounds defining the critical regions for as many active sets. This is achieved after transforming the original vector of parameters, which renders highly structured optimal active sets maps in the new space of parameters. These principles are also applicable to mp-NLP, and in this work, we explore how to adapt them from the mp-QP to the mp-NLP case.

The rest of the paper is organized as follows: Sect. 2 motivates the need for more efficient mp-NLP algorithms and highlights the methodology used in this work. In Sect. 3, a formal framework and five new algorithms are proposed to solve mp-NLP problems. In Sect. 4, two mp-NLP problems are solved, and a comparison with the state-of-the-art mp-NLP algorithms and a computational study on the dimension of mp-NLP problems are presented. The proposed methodology and results are discussed in Sect. 5.

2 Motivating example

Consider the following mp-NLP problem:

$$\begin{aligned}
&\min_x \ 1/2x_1^4 + 1/6x_1^3 + 2x_1^2 - 13/2x_1 + 1/4x_2^4 + 1/3x_2^3 + x_2^2 - 4x_2 \\
&s.t. \begin{bmatrix} 1 & 1/3 \\ 1 & 3 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \le \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} -1 & -1/10 \\ 1/10 & -1 \end{bmatrix}\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \qquad \begin{matrix} (c_1) \\ (c_2) \end{matrix} \\
&\qquad \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \le \begin{bmatrix} 2 \\ 1 \\ 2 \\ 4 \end{bmatrix} \\
&\qquad x \in X \subset \mathbb{R}^2 \\
&\qquad \theta \in \Theta \subset \mathbb{R}^2
\end{aligned} \tag{1}$$

where x and θ are the vectors of optimization variables and parameters, respectively. For any $x \in \mathbb{R}^2$, the objective function's Hessian is a positive diagonal matrix, from

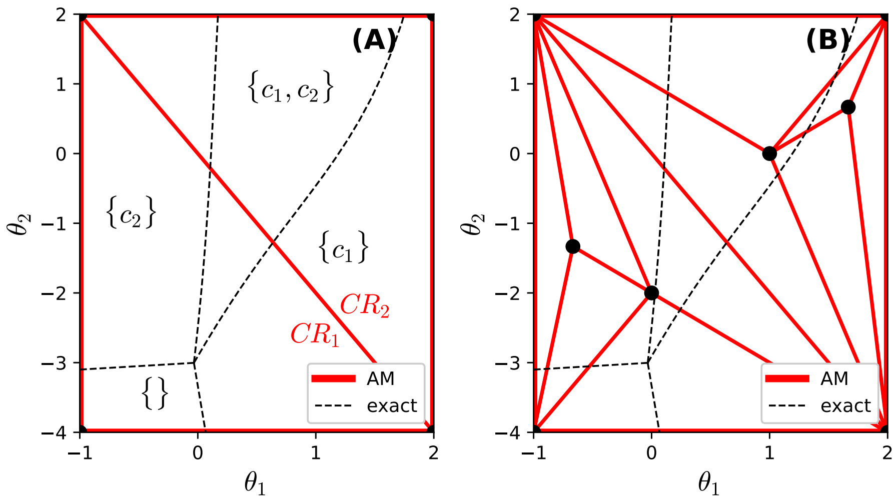

Fig. 1 Critical regions map for motivating example via AM: (**A**) initial set of convex hulls; (**B**) final set of convex hulls. The exact optimal active set map is defined via the black dotted lines

where it follows that Eq. 1 is a convex mp-NLP. The application of the AM algorithm to this problem is illustrated in Fig. 1.

Two regions (CR_1 and CR_2) are created initially using the corners of Θ (Fig. 1A), and optimizer functions are obtained via interpolation of the corresponding optimizers and parameters. The accuracy of optimizer functions is tested, and critical regions are partitioned sequentially as many times as necessary until all optimizer functions for all partitions satisfy a pre-defined tolerance (B). Note that the final map includes 10 regions and bears little resemblance to the actual optimal active sets map. This partitioning scheme depends mainly on the corners of Θ and delivers a somewhat *artificial* map of critical regions. While the AM algorithm ensures optimizer functions for all critical regions are within a pre-defined tolerance, this strategy may require a number of partitions substantially larger than the total number of active sets: these partitions may span across several of the actual regions per optimal active set; in turn, in these regions, optimizer sensitivities may be significantly different which requires the creation of many partitions to achieve optimizer accuracy and leads unavoidably to algorithm and solution complexity.

It is reported in Domínguez et al. (2010) that the improved mp-QA algorithm solves mp-NLPs more efficiently as illustrated via the solution of a benchmark problem. Yet, the GVS algorithm is the only mp-NLP algorithm known to the authors focusing specifically on building a more *natural* set of critical regions: it aims to deliver accurate optimal active set maps, via a computationally expensive partitioning scheme (applicable only to a subset of mp-NLP problems). Overall, and while the state-of-the-art mp-NLP algorithms do deliver accurate laws for optimizer calculation, we argue that the calculation of solutions minimizing algorithm complexity and more consistent with the exact optimal active set maps remains an open research subject.

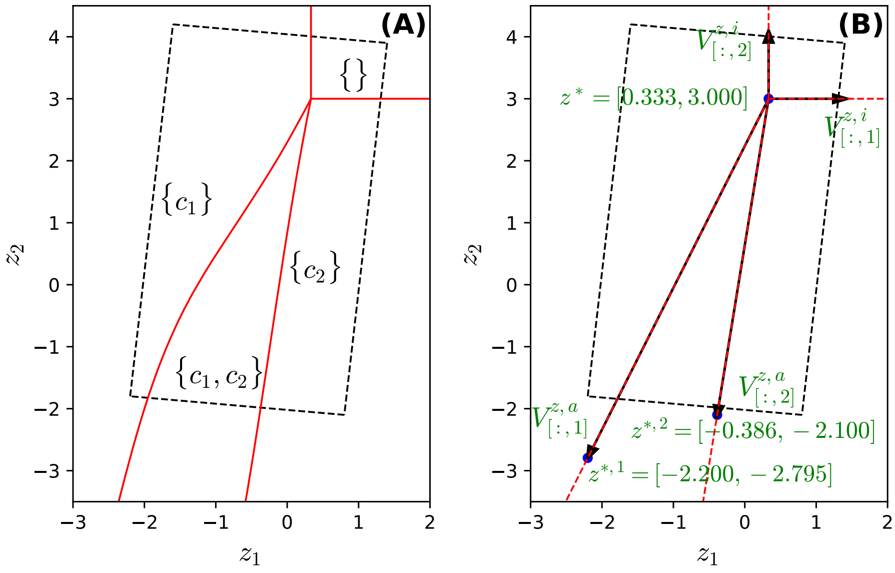

Fig. 2 Critical regions for motivating example: (**A**) exact optimal active set map; (**B**) 1-dimensional edges between critical regions estimated from optimizer sensitivities

Using the framework presented in Narciso et al. (2022), this new partitioning strategy may also be adapted to the mp-NLP case. A brief illustration is presented next; first, Eq. 1 is recast using $z = F\theta$:

$$\begin{aligned}
&\min_x \ 1/2x_1^4 + 1/6x_1^3 + 2x_1^2 - 13/2x_1 + 1/4x_2^4 + 1/3x_2^3 + x_2^2 - 4x_2 \\
&s.t. \ \begin{bmatrix} 1 & 1/3 \\ 1 & 3 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \qquad \begin{matrix} (c_1) \\ (c_2) \end{matrix} \\
&\quad \begin{bmatrix} -0.990 & 0.099 \\ 0.990 & -0.099 \\ -0.099 & -0.990 \\ 0.099 & 0.990 \end{bmatrix}\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \leq \begin{bmatrix} 2 \\ 1 \\ 2 \\ 4 \end{bmatrix} \\
&\quad x \in X \subset \mathbb{R}^2 \\
&\quad z \in Z \subset \mathbb{R}^2
\end{aligned} \tag{2}$$

where z is a vector of *transformed* parameters and Z the corresponding space of parameters. The exact map of optimal active sets in Z is depicted in Fig. 2 (A).

In mp-QP, after recasting $z = F\theta$, it suffices to calculate the objective function's global optimum (x^*), and the (constant) sensitivities of all single-constraint active sets, from where the full solution is obtained via a set of trivial algebraic operations. The same principles apply to mp-NLP; the main difference resides in the fact that the sensitivities of single-constraint active sets are <u>not constant</u> as highlighted in Fig. 3A. To circumvent this, a single *representative* optimizer is used per each of the single-constraint active sets via the corresponding KKT conditions with $z < z^*$ ($x^{*,1}$ and

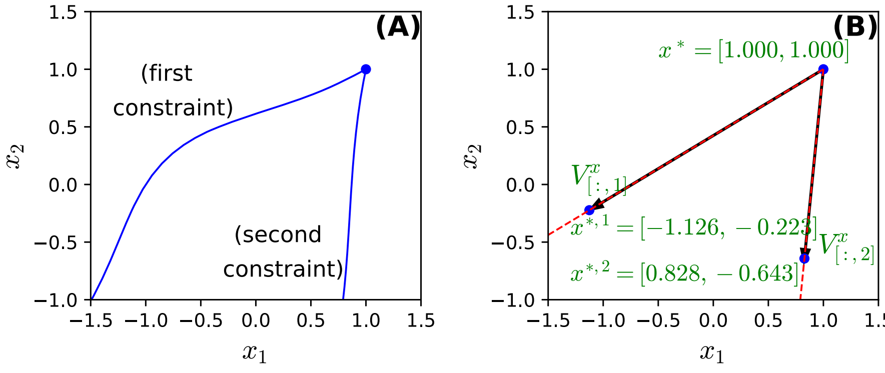

Fig. 3 Optimizers for single-constraint active sets in motivating example: (**A**) exact calculation via the KKT conditions using multiple $z < z^*$; (**B**) representative (constant) gradients using a single optimizer per active set ($x^{*,1}$ and $x^{*,2}$) and the global optimum (x^*)

$x^{*,2}$), as depicted in Fig. 3B. Constant estimations for all gradients (V^x) are obtained trivially from the reference optimizers.

The estimated 1-dimensional edges between all critical regions are then obtained from $V^{z,a} = AV^x$. Their calculation is depicted in Fig. 2B. Note that the critical regions obtained this way match very closely the optimal active set maps since this partitioning scheme explores very efficiently the topology of critical regions in Z. It also contributes to capturing optimizer sensitivities more efficiently, thus offering a promising path for mp-NLP. Section 3 presents a formal framework.

3 New solution strategies for convex mp-NLP

In this work, we address convex mp-NLP problems of the following class:

$$\begin{aligned} \min_x \;\; & f(x) \\ s.t. \;\; & Ax \leq b + F\theta \\ & P_A\theta \leq P_b \\ & x \in X \subset \mathbb{R}^n \\ & \theta \in \Theta \subset \mathbb{R}^m \end{aligned} \tag{3}$$

where x and θ are the vectors of optimization variables and parameters, respectively, and $f(x)$ is a convex nonlinear function with a unique global optimizer at x^*. $A \in \mathbb{R}^{(p\times n)}$, $b \in \mathbb{R}^{(p\times 1)}$, $F \in \mathbb{R}^{(p\times m)}$, $P_A \in \mathbb{R}^{(r\times m)}$ and $P_b \in \mathbb{R}^{(r\times 1)}$ are constant real matrices and vectors. Constant integers m, n, p, and r denote the number of parameters, optimization variables, feasible space's inequality constraints, and parameters space's bounds, respectively.

Active sets are referred to in this work via two alternative notations: (i) as a collection of abbreviated constraints of the form $\{c_j\}$, where j is any collection of constraint indexes between 1 and p, or (ii) as a binary vector y of size p, where $y_j = 0$ or 1 denotes the j^{th} constraint is inactive/active, respectively. The *empty active set* or $y^{empty} =$

[0, 0, ..., 0] ({}), and the *full active set* or $y^{full} = [1, 1, ..., 1]$ ($\{c_1, c_2, ..., c_p\}$) are defined as particular cases of this notation.

Equation 3 is recast in two steps; first, a new vector of *transformed* parameters $z = F\theta$ is defined:

$$\begin{aligned} \min_x \;& f(x) \\ s.t. \;& Ax \leq b + z \\ & P_A\theta \leq P_b \\ & z = F\theta \\ & x \in X \subset \mathbb{R}^n \\ & \theta \in \Theta \subset \mathbb{R}^m \\ & z \in Z \subset \mathbb{R}^p \end{aligned} \tag{4}$$

In those mp-NLP problems where F is a square invertible matrix, $\theta = F^{-1}z$ and an explicit definition of Z via a set of halfspaces is possible (excluding θ from Eq. 4). We refer to Eq. 4 as the transformed mp-NLP. In a second recasting step, Z is defined unbounded:

$$\begin{aligned} \min_x \;& f(x) \\ s.t. \;& Ax \leq b + z \\ & x \in X \subset \mathbb{R}^n \\ & z \in Z \subset \mathbb{R}^p \end{aligned} \tag{5}$$

We refer to Eq. 5 as the expanded mp-NLP. It follows directly from the discussion in Sections 3.1.1 and 3.1.2 in Narciso et al. (2022) that the expanded mp-NLP and the recast mp-QP problems share the same topology properties. Namely, the solutions of both problems include: (i) a unique *optimizer vertex*, (ii) p *optimizer edges*, (iii) a unique *parametric vertex*, and (iv) $2p$ *parametric edges*. In mp-NLP, the optimizer vertex is the global optimum of $f(x)$ and the optimizer edges include the full set of optimizers (x^{opt}) for all single-constraint active sets defining 1-dimensional semi-infinite lines in X; the parametric vertex and the parametric edges are obtained from the corresponding optimizers using $z = Ax^{opt} - b$ as illustrated in Figs. 2 and 3.

In mp-NLP, optimizer edges generally define *nonlinear* semi-infinite lines. Optimizer sensitivities are not constant and this is the most critical subject towards the adaptation of the newly developed mp-QP paradigm to the mp-NLP case. Building on the initial analysis proposed in Sect. 2, three new solution strategies are presented in Sects. 3.1–3.3. These enable the calculation of the solutions for the expanded and transformed mp-NLPs with *incremental* accuracy as schematically depicted in Fig. 4. Five algorithms deliver this functionality. These were coded in Python (Van Rossum and Drake 2009), and supported on the numpy (Harris et al. 2020), SciPy (Virtanen 2020) and pyomo (Bynum et al. 2021) libraries. A high-level discussion is presented in this section, including the steps of all algorithms in the form of pseudo-codes; the Python codes for all algorithms are available in file https://github.com/diogonarciso/mpNLP (to include later in Github). We remark that the goal of this paper is the *offline* calculation of critical region maps and optimizer functions for increased solution accuracy

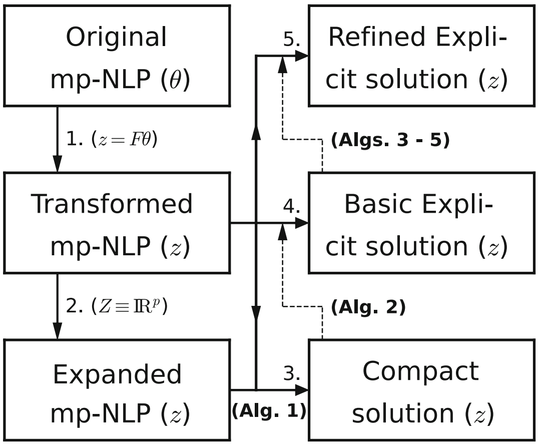

Fig. 4 New solution strategies for convex mp-NLP problems

in mp-NLP problems. A first exploration of this framework in the context of online Model Predictive Control is presented in Narciso et al. (2023); this subject will be addressed in future work.

3.1 Compact solution

This concept was proposed in Narciso et al. (2022) for mp-QP. Taking the optimizer vertex (x^*), the parametric vertex (z^*), the gradients of optimizer edges (V^x), and the gradients of parametric edges ($V^{z,i}$ - inactive constraints, and $V^{z,a}$ - active constraints), all critical regions (Eq. 6), and optimizer functions (Eq. 7), are expressed exactly in *compact* form as follows:

$$z = z^* + (V^{z,i})((y^{full} - y)l) + (V^{z,a})(yl),\ y \in \{0, 1\}, l \geq 0 \tag{6}$$

$$x^{opt} = x^* + (V^x)(yl) \tag{7}$$

where l is a nonnegative vector of multipliers denoting the extents of feasible space expansion/contraction beyond z^* via each of the p inequality constraints, and x^{opt} represents the vector of optimizers. Equation 6 fits in the class of Linear Complementarity Problems (LCP), though its derivation is distinct from the classic LCP formulation employed to obtain the solution of mp-QPs. A detailed discussion on this subject is presented in Sections 3.1 and 3.6 of Narciso et al. (2022), where the derivation of Eqs. 6 and 7 and a comparison with the classic LCP formulation are presented.

The compact solution (CS) format is also directly applicable to the expanded and transformed mp-NLP problems given their shared topology properties. The calculation of all vertices and gradients is adapted from Narciso et al. (2022) to the mp-NLP case, where obtaining estimates for V^x in Step 3 requires the most significant update. The CS is presented here as a trivial extension from mp-QP and as an initial step in the proposed framework. Its applicability in the context of mp-NLP is limited and presented in greater detail in Appendix 1 (Sect. 2).

3.2 Basic explicit solution

Explicit solutions may be obtained for any active set by setting y directly in Eqs. 6 and 7 and listing their solution fragments. An improved methodology for the calculation of the *basic explicit solution* (BES) is proposed in this section, delivering critical regions in the context of the transformed mp-NLP problem only. Sections 3.2.1–3.2.5 discuss the supporting concepts of Algorithm 2, which is presented in Sect. 3.2.6.

3.2.1 Constraints status assessment

Without prior assessment, all p inequality constraints of the transformed mp-NLP may be inactive or active in distinct regions of Θ, from where the theoretical maximum number of active sets comprising its solution amounts to 2^p. To reduce problem complexity it is desirable to conclude if any, and which, constraints are $\underline{\text{always inactive}}$ or $\underline{\text{always active}}$, thus potentially reducing significantly the total number of active sets and critical regions to assess during the calculation of the BES. This can be achieved by adapting Eq. 6 as follows:

$$\begin{aligned}
&F\theta = z^* + (V^{z,i})((y^{full} - y)l) + (V^{z,a})(yl),\ y \in \{0, 1\}, l \geq 0 \\
&y_j = 0 \\
&P_A\theta \leq P_b
\end{aligned} \tag{8}$$

where j is any integer between 1 and p, thus enforcing the corresponding parametric edge (inactive constraint) on the definition of the permissible space; $z = F\theta$ and the bounds of Θ are also enforced on this problem statement. If a solution exists for Eq. 8, there is at least one $\theta \in \Theta$, where the j^{th} constraint is inactive. Conversely, if no solution exists, there is no $\theta \in \Theta$, where the j^{th} constraint is inactive at the optimal solutions; in other words this constraint is $\underline{\text{always active}}$ in Θ.

An equivalent statement is defined to test parametric edges associated with active constraints; it suffices to update $y_j = 1$ in Eq. 8, and in this case: if no solution exists for this problem, the j^{th} constraint is $\underline{\text{always inactive}}$ in Θ. The total numbers of always active/always inactive constraints are denoted as $0 \leq N_a \leq p$ and $0 \leq N_i \leq p$, respectively. Equation 8 may be recast equivalently as a Mixed Integer Linear Programming (MILP) problem (checking feasibility to reach one of the conclusions above). Details are discussed in Sect. 4 of Appendix 1.

3.2.2 Critical region dimensionality

All critical regions are implicitly defined in Eq. 6 via a unique combination of p column vectors of $V^{z,i}$ and $V^{z,a}$, spanning them in Z. Their dimensions in this space depend exclusively on the *linear independence* between their p vectors of directions: given an active set y and the corresponding matrix of column vectors from $V^{z,i}$ and $V^{z,a}$ - denoted as $V^{z,y}$ - the dimension of its critical region in Z is obtained trivially from $rank(V^{z,y})$.

Since $V^{z,i}$ is the identity matrix of size p (all vectors are orthogonal to each other), critical region dimension may alternatively be calculated from the subset of active constraints for a given active set y. In mp-NLP, optimizer edges define nonlinear semi-infinite lines in X, and thus the estimated matrices V^x and $V^{z,a}$ via Algorithm 1 depend on the defined parameters space. A more robust way of checking dimensionality consists of checking linear independence between the relevant rows of A. This is achieved as follows: (i) calculate $rank(A^{z,y})$, where $A^{z,y}$ denotes the set of rows of A matching the active constraints of y; (ii) if $rank(A^{z,y}) = \sum_{j=1}^{p} y_j$, the corresponding critical region is full dimensional in Z.

3.2.3 A hybrid partitioning strategy

A new strategy for critical region construction is presented, making use on the one hand of the nonnegative combinations of the gradients of parametric edges as presented in Eq. 6, and on the other hand incorporating, selectively, convex hull constraints in their definition. Given an active set y, such that constraints $c_{J_1}, \ldots, c_{J_j}$ are active, and constraints $c_{K_1}, \ldots, c_{K_k}$ are inactive (where $[J_1, \ldots, J_j] \cup [K_1, \ldots, K_k] = [1, 2, \ldots, p]$ and $[J_1, \ldots, J_j] \cap [K_1, \ldots, K_k] = \emptyset$); the corresponding critical region is now expressed as follows:

$$\begin{aligned}
&z = \left(z^* s_1 + z^{*,J_1} s_2 + \cdots + z^{*,J_j} s_{j+1}\right) + \left(e^{(K_1)} t_1 + \cdots + e^{(K_k)} t_k\right)\\
&s \geq 0, t \geq 0\\
&\sum_{i=1}^{j+1} s_i = 1
\end{aligned} \tag{9}$$

where $s \in \mathbb{R}_{\geq 0}^{j+1}$ and $t \in \mathbb{R}_{\geq 0}^{k}$ span the parametric edges for active and inactive constraints of y, respectively, in place of l. The set of points $z^{*,J_1} \cdots + z^{*,J_j}$ are obtained from $z = Ax - b$, using the set of optimizers from the corresponding optimizer edges (Step 3A.2 in Algorithm 1). This partitioning strategy is entirely equivalent to the CS, except that an additional constraint ($\sum_{i=1}^{j+1} s_i = 1$) is now enforced in Eq. 9, and which limits the extents of critical regions via the set of parametric edges associated with active constraints to promote solution accuracy. An equivalent definition for Eq. 9 using matrix notation and the matching optimizer functions are presented in Eqs. 10 and 11, respectively.

$$\begin{bmatrix} z \\ 1 \end{bmatrix} = \begin{bmatrix} F\theta \\ 1 \end{bmatrix} = \left[\begin{array}{c|ccc|ccc} z^* & z^{*,J_1} & \ldots & z^{*,J_j} & e^{(K_1)} & \ldots & e^{(K_k)} \\ 1 & 1 & \ldots & 1 & 0 & \ldots & 0 \end{array}\right] \begin{bmatrix} s \\ t \end{bmatrix}, s \geq 0, t \geq 0 \tag{10}$$

$$x^{opt} = \left[x^* \middle| x^{*,J_1} \ldots x^{*,J_j} \middle| 0 \ldots 0\right] \begin{bmatrix} s \\ t \end{bmatrix} \tag{11}$$

where matrices in Eqs. 10 and 11 are denoted as M^{CR} and M^{OF}, respectively. A graphical illustration of the partitioning strategy via Eq. 10 and a discussion on the theoretical consistency between Eqs. 10 and 11 are presented in Sect. 5 of Appendix 1.

3.2.4 Optimal active set detection

The following LP enables to conclude if a given active set y defines a critical region in Θ:

$$\begin{aligned} &\min_{s,t,\theta} \begin{bmatrix} 1 \dots 1 \end{bmatrix} \theta \\ &s.t. \begin{bmatrix} F\theta \\ 1 \end{bmatrix} = \left[\begin{array}{c|ccc|ccc} z^* & z^{*,J_1} & \dots & z^{*,J_j} & e^{(K_1)} & \dots & e^{(K_k)} \\ 1 & 1 & \dots & 1 & 0 & \dots & 0 \end{array}\right] \begin{bmatrix} s \\ t \end{bmatrix} \\ &P_A\theta \leq P_b \\ &[s|t]^T \in \mathbb{R}^{p+1}_{>0} \\ &\theta \in \Theta \subset \mathbb{R}^m \end{aligned} \tag{12}$$

where the equality constraints are set from the corresponding critical region of y, via Eq. 10. If Eq. 12 returns a feasible solution, there is at least one $\theta \in \Theta$ where active set y is detected as the optimal solution.

3.2.5 Parametric edges scope

The selection of the reference optimizers and parameters in Step 3A.2 of Algorithm 1 ($x^{*,j}$ and $z^{*,j}$ with $j = 1, \dots, p$) becomes more important in the context of Eq. 10. For instance, in Fig. 2B, the convex combinations enabled via Eq. 10 for all active sets cover the full extent of the space of parameters (depicted in Fig. 5), thus making the selected reference points an adequate choice. In some mp-NLP problems, this may not be the case; this is illustrated with a second example in Fig. 7B, where a small triangle of the transformed space around [0, 0] is not covered via any of the permitted convex combinations for active set $\{c_1, c_2\}$, as expected. To accommodate for this scenario, Step 3A.2 in Algorithm 1 includes a user-defined vector of slacks δ, which for all $j = 1, \dots, p$, *negatively increments* z_j^{min}, thus ensuring that the reference points calculated in this step cover the full space of parameters via their critical regions (Fig. 7A).

Finding δ such that it covers as <u>tightly</u> as possible the parameters space presents several challenges. We argue that a simpler approach towards setting δ is a better compromise between solution accuracy and algorithm complexity: for all $j = 1, \dots, p$, $\delta_j = \alpha(z_j^{min} - z_j^*)$, where α denotes a *relative* increment parameter with respect to the initial estimation of z_j^{min} ($\alpha \approx 0 - 0.1$).

3.2.6 Critical region mapping

The calculation of explicit solutions in Multiparametric Programming is generally based on the exploration of Θ, where a set of critical regions are calculated in sequence until the so-called *rest space* is empty (Dua et al. 2002), Pistikopoulos et al. (2002). A brute force approach would consist of enumerating all their $\mathbf{2^p}$ active set and delivering the matching critical regions via Eq. 10, which would generally result in a very expensive option. The proposed strategy for optimal active set detection is based on the seminal work presented in Gupta et al. (2011), where a combinatorial strategy was developed for mp-QP; this strategy was extended to the mp-NLP case in Mate et al. (2020) and now adapted to the optimal active sets identification problem in the context of the transformed parameters space.

Equation 6 enables a more efficient mapping as hinted in Sects. 3.2.1–3.2.5. First, by defining the $2p$ MILPs for the $2p$ parametric edges via Eq. 8, the number of candidate active sets reduces to $\mathbf{2^{p-N_i-N_a}}$: all candidate active sets are such that the N_i and N_a *fixed* coordinates of y are set the constant values equal to 0 and 1, respectively. When all $2p$ MILPs are infeasible, Θ defines an infeasible region and in this case, the BES includes **0** active sets.

The number of candidates may be further decreased by excluding all active sets necessarily defining low-dimensional critical regions. In line with the discussion in Sect. 3.2.2, this occurs for any given active set where the number of active constraints exceeds the number of optimization variables. An active set *enumeration* strategy is proposed to exclude them, where the $p - N_i - N_a$ *free* coordinates of y are selectively set to 0 or 1, and where up to $n - N_a$ of these coordinates are equal to 1. The enumeration begins with $i = 0$ free coordinates equal to 1; i is then incremented sequentially to deliver the full set of candidate active sets, which amounts to a total of $\mathbf{\sum_{i=0}^{n-N_a} C_i^{p-N_i-N_a}}$. These active sets are then tested such that only those defining <u>full-dimensional</u> critical regions <u>detected</u> in Θ are listed in the BES. All steps are summarized in Algorithm 2.

Algorithm 2

```
Step 1: Initialize vectors $u^i$ and $u^a$ with all coordinates equal to 0, and vector $y^{ref}$ with all
  coordinates equal to -1$^{(a)}$ (all vectors of size $p$).
Step 2: For all $j = 1, ..., p$, solve Eq. 8 with $y_j = 0$; if infeasible, set $u_j^a = 1$. $N_a = \sum_{j=1}^{p} u_j^a$.
Step 3: For all $j = 1, ..., p$, solve Eq. 8 with $y_j = 1$; if infeasible, set $u_j^i = 1$. $N_i = \sum_{j=1}^{p} u_j^i$.
Step 4: If $N_a = p$ and $N_i = p$, terminate: no critical region exists ($\Theta$ defines an infeasible region).
  Else, for all $j = 1, ..., p$: (i) if $u_j^a = 1$, set $y_j^{ref} = 1$, or (ii) if $u_j^i = 1$, set $y_j^{ref} = 0$.
Step 5: Enumerate all $\sum_{i=0}^{n-N_a} C_i^{p-N_i-N_a}$ candidate active sets$^{(b)}$. Set $j = 1$.
Step 6: Set $y$ to the jth active set in the list of candidates.
Step 7: Calculate $rank(A^y)$; if $rank(A^y) = \sum_{j=1}^{p} y_j$, calculate the corresponding critical region via
  Eq. 10$^{(c)}$. Else, go to Step 9.
Step 8: Solve the LP in Eq. 12; if a feasible solution is found, add the jth active set and its critical
  region to the BES. Else, go to Step 9.
Step 9: If $j < \sum_{i=0}^{n-N_a} C_i^{p-N_i-N_a}$, set $j = j + 1$ and go to Step 6.
Step 10: Else, define the general optimizer functions via Eq. 14 (details in Sect. 3.2.7).
```

$^{(a)}$ Notation for y^{ref}. -1: inactive or active; 0: always inactive; 1: always active, constraints in Θ.

$^{(b)}$ All fixed coordinates of all candidate active sets are set equal to the matching coordinates of y^{ref}. All candidate active sets are enumerated sequentially based on the number of active constraints: first, the unique active set where $i = 0$ free components of y are set to 1; then, the $p - N_i - N_a$ active sets where $i = 1$ free components of y are set to 1, and so on.

$^{(c)}$ Active sets defining critical regions of dimension $p - 1$ are also saved. All candidate active sets listed in Step 6 are first compared against these low-dimensional active sets; any candidate active set sharing the same active constraints of one of these low-dimensional active sets is also associated with a low-dimensional critical region, and excluded from the BES (skip Steps 7 and 8).

3.2.7 Offline and online solutions

Instead of recording critical regions in accordance with the notation in Eq. 10, storing their inverse matrices in line with Eq. 13 is presented as an alternative. From an offline perspective, the first option is preferable since it dismisses the calculation of an inverse matrix per detected active set. In an online context, the second option enables a faster route for critical region identification, since this becomes a simpler function evaluation problem.

$$\begin{bmatrix} s \\ t \end{bmatrix} = \left[\begin{array}{c|ccc|ccc} z^{*} & z^{*,J_1} & \dots & z^{*,J_j} & e^{(K_1)} & \dots & e^{(K_k)} \\ 1 & 1 & \dots & 1 & 0 & \dots & 0 \end{array}\right]^{-1} \begin{bmatrix} F\theta \\ 1 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{13}$$

The notation used for matrices M^{CR} (Eq. 10) and M^{OF} (Eq. 11) is such that all active/inactive constraints are captured in the central/right blocks of these matrices, respectively (sensitive/insensitive transformed parameters). In practice, all columns of M^{CR} and M^{OF} are sorted and saved in the original order; vector $[s|t]]^T$ is sorted accordingly and denoted as γ. This enables a general and computationally inexpensive representation of all optimizer functions for all active sets using Eq. 14:

$$x^{opt} = \left[x^{*} \middle| x^{*,1} \dots x^{*,p} \right] \gamma \tag{14}$$

3.3 Refined explicit solution

The *refined explicit solution* (RES) builds on the BES to improve solution accuracy. The proposed strategy comprises three stages: first, a set of additional points per edge is calculated (Algorithm 3); then, a set of more precise partitions is obtained from these points (Algorithm 4); at last, these partitions are broken down in smaller partitions/critical regions, if necessary until their optimizer functions deliver optimizers within a pre-defined error tolerance (Algorithm 5).

3.3.1 Edges refinement

All edges are implicitly defined in the CS and the BES as single line segments. These are now halved and tested sequentially using as many segments as necessary via a set of additional points to improve their accuracy. Their calculation is summarized in Algorithm 3.

Algorithm 3

```
Step 1: Define the list of $p$ single constraint active sets. Set $j = 1$ (active set: $\{c_1\}$).
Step 2: If $u^i_j = 1$ go to Step 7$^{(a)}$.
Step 3: Initialize the list of intervals to process with: $z_j^{min} - \delta_j \le z_j \le z_j^*$.
Step 4: While there is at least one interval to process, take the first interval in this list (denoted as
  $z_j^{lb} \le z_j \le z_j^{ub}$) and:
    Step 4.1: Calculate $z_j^{center} = (z_j^{lb} + z_j^{ub})/2$, and solve the KKT conditions for active set $\{c_j\}$ at
  $z_j = z_j^{center}$, to obtain optimizer $x^{*,exact}$.
    Step 4.2: Calculate the estimated optimizers at $z_j^{center}$: $x^{*,estim} = (x^{lb} + x^{ub})/2^{(b)}$.
    Step 4.3: Calculate the approximation error $\epsilon = \sum_{j=1}^{n} (x_j^{*,exact} - x_j^{*,estim})^2$.
    Step 4.4: Remove the first interval from the list of intervals.
    Step 4.5: If $\epsilon > \zeta^{edges}$, where $\zeta^{edges}$ is a pre-defined error tolerance: (i) add $x^{*,exact}$ to the $j^{\text{th}}$
  optimizer edge, (ii) break the current interval in two intervals: $z_j^{lb} \le z_j \le z_j^{center}$ and
  $z_j^{center} \le z_j \le z_j^{ub}$, and (iii) add the two intervals to the list of intervals. Go to Step 4 (while loop).
  Step 5: For all optimizers added to the $j^{\text{th}}$ optimizer edge in Step 4.5, calculate the matching
  reference points $z = Ax^{*,exact} - b$, and add them to the $j^{\text{th}}$ parametric edge (active constraint).
Step 6: If $j < p$, set $j = j + 1$ (active set: $\{c_j\}$) and go to Step 2.
Step 7: For all $j = 1, \ldots, p$, order the full set of representative points for the $j^{\text{th}}$ optimizer and
  parametric edges from the highest to the lowest $z_j$.
```

$^{(a)}$ From Sect. 3.2.1, if this condition holds, the current edge is not used in the calculation of the BES, and its refinement is unnecessary in this case.
$^{(b)}$ Formally, the optimizer function would be calculated first via interpolation of x^{lb} (at z_j^{lb}) and x^{ub} at (z_j^{ub}), and then $x^{*,estim}$ obtained at z_j^{center}. The interpolation step is dismissed and the optimizer obtained directly via x^{lb} and x^{ub}.

3.3.2 Initial set of critical regions

Making use of the points calculated in Sect. 3.3.1, a new set of critical regions matching more closely the actual optimal active set map are obtained, where each active set now includes one or more critical regions. All steps are summarized in Algorithm 4.

Algorithm 4

```
Step 1: If no new points are added in Algorithm 3, terminate: the initial set of critical regions matches
  the BES.
Step 2: Define the list of all $N_{as}$ active sets identified in the BES. Set $j = 1$ (first active set).
Step 3: Define the list of all reference points for the $j^{\text{th}}$ active set from the points in the associated
  parametric edges, excluding those points at their last positions$^{(a)}$; list them as "unprocessed"$^{(b)}$.
Step 4: Define the first critical region via Eq. 10, using $z^*$ and the vectors of representative points per
  each of the relevant parametric edges at their second positions $^{(c)}$. Update $z^*$ to the "processed"
  status. Set the current positions of all parametric edges to their second positions.
Step 5: While there are any "unprocessed" reference points:
    Step 5.1: Create a new region using a reference point per edge at their current positions and add a
  point from one of the associated parametric edges at the next available position$^{(d)}$.
Step 5.2: At the same parametric edge used in Step 5.1 (ii), update the status of the point at the
  current position to "processed", and increment this position by 1. Go to Step 5 (while loop).
Step 6: If $j < N_{as}$, set $j = j + 1$ (next active set in the list) and go to Step 3.
```

$^{(a)}$ All points in edges are ordered via Step 8 of Algorithm 3: for all $j = 1, \ldots, p$, z^* is at the first position, and $z_j^{min} - \delta_j$ at the last position.
$^{(b)}$ The "processed" and "unprocessed" statuses reflect if points at edges are available or not, respectively, for the purposes of creating critical regions.
$^{(c)}$ All gradients for all associated inactive constraints are included accordingly in Eq. 10.
$^{(d)}$ To promote a balanced construction of partitions, the parametric edge including the highest number of "unprocessed" points is selected in this step.

3.3.3 Critical regions check and refinement

Using the full set of initial critical regions delivered in Algorithm 4, two checks are made: (i) detecting which are included in the parameters space, and (ii) testing the accuracy of their optimizer functions. Accordingly, critical regions are excluded or broken down into smaller partitions until optimizers are estimated within a pre-defined tolerance. Algorithm 5 summarizes all calculations.

Algorithm 5

```
Step 1: Define the list of all $N_{cr}$ initial critical regions identified in Algorithm 4. Set $j = 1$ (first critical region).
Step 2: Initialize the list of partitions to check in the current iteration ($j$) using the $j^{\text{th}}$ critical region in the list of critical regions.
Step 3: While there is at least one partition to check, take the first partition in this list and:
    Step 3.1: Check if the partition is included in the parameters space via Eq. 12. If not, remove the partition from the list of partitions to check and go to Step 3 (while loop).
    Step 3.2: Calculate $z^{center}$, as the average of all reference points used in its definition, and solve the KKT conditions for its active set at $z = z^{center}$, to obtain optimizer $x^{*,exact}$.
    Step 3.3: Define its optimizer functions using the matching optimizers via Equation 11.
    Step 3.4: Calculate the estimated optimizer from the optimizer functions at $z^{center}$, $x^{*,estim}$, as the average of the optimizers used their definition (a).
    Step 3.5: Calculate the approximation error $\epsilon = \sum_{j=1}^{n}(x_j^{*,exact} - x_j^{*,estim})^2$.
    Step 3.6: Remove the first partition from the list of partitions.
    Step 3.7: If $\epsilon > \zeta^{partitions}$, where $\zeta^{partitions}$ is a pre-defined error tolerance: (i) break the current partition in $\beta + 1$ partitions(b) via all the combinations of $\beta$ of its reference points with $z^{center}$, and (iii) add these partitions to the list of partitions to check. Go to Step 3 (while loop).
    Step 3.8: Else, if $\epsilon \leq \zeta^{partitions}$, add the partition/critical region and its optimizer functions to the RES. Go to Step 3 (while loop).
Step 4: If $j < N_{cr}$, set $j = j + 1$ (next initial critical region in the list) and go to Step 2.
```

(a) See note (b) of Algorithm 3 for details.
(b) Where β denotes the number of active constraints in the current active set/partition.

Several user-defined parameters are defined in Algorithms 1-5 (e.g. ζ^{edges}); some guidelines on setting these parameters and a discussion on their impact on the solutions of mp-NLPs is presented in Appendix 1.

3.4 Computational complexity

The computational complexity of Algorithms 1–5 is summarized in Table 1. This is expressed via the number of auxiliary optimization problems required per algorithm, which contribute most significantly to their execution. It is not possible to quantify complexity exactly in all steps, since these algorithms depend on the definition of the parameters space, tolerances, and problem nonlinearity: a set of *exact* and *estimated* upper bounds are listed where applicable. A more detailed discussion on the subject is presented in Section 7 of Appendix 1.

Algorithm 1 is likely the lightest mp-NLP algorithm developed to date, where complexity scales linearly with the number of constraints instead of the more common exponential dependency reported in the field (Pistikopoulos et al. 2020). Algorithm 2 requires the validation of all critical regions which amounts to roughly the number of optimal active sets detected in Θ (N_{as}). In Algorithm 3 all edges are refined which includes at least the assessment of a middle point per edge. Algorithm 5 checks and breaks all partitions where applicable, which includes at least the set of initial critical regions (N_{cr}) derived in Algorithm 4.

We remark that the number of NLPs solved as listed in Table 1 originate from the KKT conditions; these define systems of nonlinear equations and not formal optimization problems (subject to a set of inequality constraints). Their solutions are cheaper

Table 1 Computational complexity of Algorithms 1–5: upper bounds

Algorithm	LPs	NLPs	MILPs
1	p	$p+1$	0
2	$\sum_{i=0}^{n-N_a} C_i^{p-N_i-N_a}$	0	$2p$
3	0	$\approx 10p$	0
4	0	0	0
5	$\approx 10N_{cr}$	$\approx 10N_{cr}$	0

to obtain than solving an equivalent NLP (setting θ in Eq. 3, finding the optimal active set and solving for x).

4 Results

4.1 Motivating example

The motivating example (Eq. 1) is now solved in accordance with the steps highlighted in Fig. 4 to deliver the CS, BES and RES. The transformed and expanded mp-NLPs are defined in Eqs. 2 and 15, respectively. The following settings on user-defined parameters for algorithm execution are employed: $\Delta z = 0.05$, $\delta = [0, 0]$, $\zeta^{edges} = \zeta^{partitions} = 10^{-2}$. Detailed, step-by-step calculations of all solutions including graphical insights are presented in Sect. 8 of Appendix 1.

$$\begin{aligned} \min_x \ & 1/2x_1^4 + 1/6x_1^3 + 2x_1^2 - 13/2x_1 + 1/4x_2^4 + 1/3x_2^3 + x_2^2 - 4x_2 \\ s.t. \ & \begin{bmatrix} 1 & 1/3 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \le \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \qquad \begin{matrix} (c_1) \\ (c_2) \end{matrix} \\ & x \in X \subset \mathbb{R}^2 \\ & z \in Z \subset \mathbb{R}^2 \end{aligned} \tag{15}$$

4.1.1 Compact solution

From Algorithm 1: $x^* = [1.000, 1.000]^T$ (Step 1) and $z^* = [0.333, 3.000]^T$ (Step 2). A set of auxiliary optimizers are calculated in Step 3, enabling the calculation of optimizer edges' gradients: $V^x = [[-2.126, -1.223]^T, [-0.172, -1.643]^T]$. Then, $V_i^z = [[1, 0], [0, 1]]$ (Step 4), and $V_a^z = [[-2.533, -5.794]^T, [-0.720, -5.100]^T]$ (Step 5). The CS is delivered accordingly in Eqs. 16 and 17 (Step 6). The map of critical regions is depicted in Fig. 2B, where all regions extend to infinity (beyond the regions shown in the map).

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 0.333 \\ 3.000 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} (1-y_1)l_1 \\ (1-y_2)l_2 \end{bmatrix} + \begin{bmatrix} -2.533 & -0.720 \\ -5.794 & -5.100 \end{bmatrix} \begin{bmatrix} y_1 l_1 \\ y_2 l_2 \end{bmatrix}, y \in \{0, 1\}, l \ge 0 \tag{16}$$

Table 2 Basic explicit solution for motivating example

Critical region	M^{CR}
$\{\}$	$\begin{bmatrix} 0.333 & 1.000 & 0.000 \\ 3.000 & 0.000 & 1.000 \\ 1 & 0 & 0 \end{bmatrix}$
$\{\mathbf{c_1}\}$	$\begin{bmatrix} 0.333 & -2.200 & 0.000 \\ 3.000 & -2.795 & 1.000 \\ 1 & 1 & 0 \end{bmatrix}$
$\{\mathbf{c_2}\}$	$\begin{bmatrix} 0.333 & 1.000 & -0.386 \\ 3.000 & 0.000 & -2.100 \\ 1 & 0 & 1 \end{bmatrix}$
$\{\mathbf{c_1}, \mathbf{c_2}\}$	$\begin{bmatrix} 0.333 & -2.200 & -0.386 \\ 3.000 & -2.795 & -2.100 \\ 1 & 1 & 1 \end{bmatrix}$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^{opt} = \begin{bmatrix} 1.000 \\ 1.000 \end{bmatrix} + \begin{bmatrix} -2.126 & -0.172 \\ -1.223 & -1.643 \end{bmatrix} \begin{bmatrix} y_1 l_1 \\ y_2 l_2 \end{bmatrix} \tag{17}$$

4.1.2 Basic explicit solution

From Algorithm 2: $u^i = u^a = [0, 0]$ and $y^{ref} = [-1, -1]$ (Step 1). Four MILPs are defined and solved in Steps 2 and 3 for inactive and active parametric edges; all MILPs return a feasible solution, from where $u^i = u^a = [0, 0]$, and $N_a = N_i = 0$. Since $N_a = N_i = 0 \neq 2 = p$, there is at least one critical region in Θ (not infeasible); $y^{ref} = [-1, -1]$ (Step 4). A total of $\sum_{i=0}^{2} C_i^2 = 1 + 2 + 1 = 4$ active sets are defined, including $\{\}$, $\{c_1\}$, $\{c_2\}$ and $\{c_1, c_2\}$ (Step 5). All active sets are tested in Steps 6 - 9; all matching critical regions are full-dimensional and defined via Eq. 10 (Step 7). Equation 12 confirms that all critical regions are included in Θ and added to the BES (Step 8). The general optimizer functions are obtained in Step 10 (Eq. 18).

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^{opt} = \left[\begin{array}{c|cc} 1.000 & -1.126 & 0.828 \\ 1.000 & -0.223 & -0.643 \end{array}\right] \gamma \tag{18}$$

All critical regions in the BES are listed in Table 2 and depicted in Fig. 5; note that regions only extend to infinity via the set of parametric edges associated with inactive constraints.

4.1.3 Refined explicit solution

Edges refinement:
From Algorithm 3: the list of all single constraint active sets includes: $\{c_1\}$ and $\{c_2\}$ (Step 1). $u^i = [0, 0]$, and no edge is dismissed from the remaining refinement steps (Step 2). The two edges are then broken down sequentially in Steps 3 and 4: two additional optimizers are added to the first optimizer edge and none is required for the second edge. In Step 5, the corresponding points for the first parametric edge are

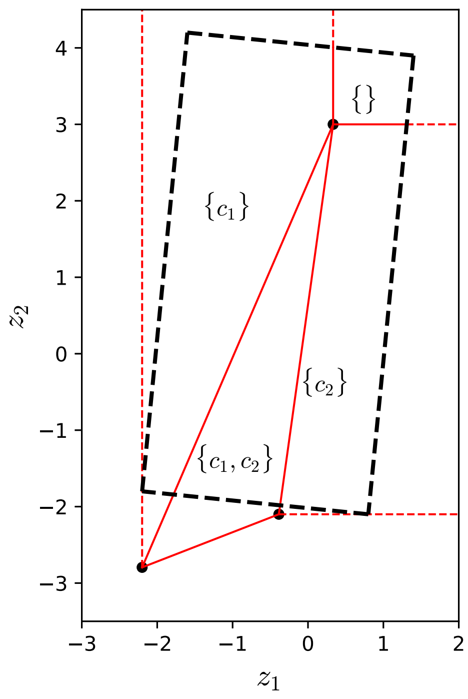

Fig. 5 Map of critical regions for motivating example (BES). The transformed parameters space is bounded by the black dotted lines

Table 3 Reference points included in the optimizer and parametric edges of motivating example

	x_1	x_2	z_1	z_2
Vertex	1.000	1.000	0.333	3.000
$\{c_1\}$	− 0.125	0.574	− 0.933	0.598
	− 0.676	0.329	− 1.567	− 0.690
	− 1.126	− 0.223	− 2.200	− 2.795
$\{c_2\}$	0.828	− 0.643	− 0.386	− 2.100

also added to improve its accuracy. All points are sorted in Step 7. The complete set of points per edge is summarized in Table 3.

Initial set of critical regions:

From Algorithm 4: the termination condition in Step 1 is not satisfied; the list of all active sets validated in the BES include: $\{\}$, $\{c_1\}$, $\{c_2\}$ and $\{c_1, c_2\}$ (Step 2). All initial partitions are created in Steps 3-6. In the case of active sets $\{\}$ and $\{c_2\}$, since no points were added to the second edge, their partitions match those of the BES. Concerning active sets $\{c_1\}$ and $\{c_1, c_2\}$, the new points listed in Table 3 are also used to calculate the initial critical regions. These are illustrated in Fig. 6, where a sequential increment on the position of the first edge delivers their critical regions.

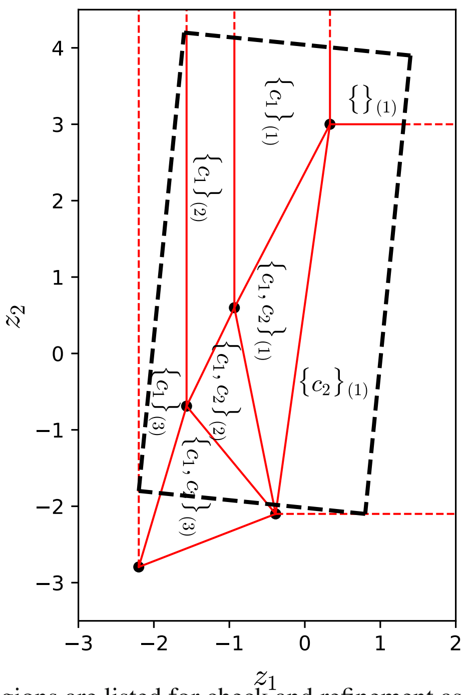

Fig. 6 Map of initial critical regions for motivating example (RES)

Critical regions check and refinement:
From Algorithm 5: eight initial critical regions are listed for check and refinement as depicted in Fig. 6 (Step 1). All critical regions are checked in Step 3.1 and confirmed to be included in Θ; $x^{*,exact}$, the optimizer functions, $x_j^{*,estim}$ and ϵ are calculated for all critical regions in Steps 3.2–3.5, with $\epsilon < \zeta^{partitions} = 10^{-2}$. The condition in Step 3.7 is not applicable to any region; all critical regions and their optimizer functions are added to the RES in Step 3.8. No additional critical region partitioning is required: the final set of critical regions matches the initial regions delivered in Algorithm 4. The RES is presented in Table 4.

4.2 Benchmark mp-NLP

The following benchmark mp-NLP problem is presented in Pistikopoulos et al. (2007), Narciso (2009), Domínguez et al. (2010):

Table 4 Refined explicit solution for motivating example

Critical region	M^{CR}	M^{OF}
$\{\}_1$	$\begin{bmatrix} 0.333 & 1.000 & 0.000 \\ 3.000 & 0.000 & 1.000 \\ 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 1.000 & 0.000 & 0.000 \end{bmatrix}$
$\{\mathbf{c_1}\}_1$	$\begin{bmatrix} 0.333 & -0.933 & 0.000 \\ 3.000 & 0.598 & 1.000 \\ 1 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1.000 & -0.125 & 0.000 \\ 1.000 & 0.574 & 0.000 \end{bmatrix}$
$\{\mathbf{c_1}\}_2$	$\begin{bmatrix} -0.933 & -1.567 & 0.000 \\ 0.598 & -0.690 & 1.000 \\ 1 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} -0.125 & -0.676 & 0.000 \\ 0.574 & 0.329 & 0.000 \end{bmatrix}$
$\{\mathbf{c_1}\}_3$	$\begin{bmatrix} -1.567 & -2.200 & 0.000 \\ -0.690 & -2.759 & 1.000 \\ 1 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} -0.676 & -1.126 & 0.000 \\ 0.329 & -0.223 & 0.000 \end{bmatrix}$
$\{\mathbf{c_2}\}_1$	$\begin{bmatrix} 0.333 & 1.000 & -0.386 \\ 3.000 & 0.000 & -2.100 \\ 1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1.000 & 0.000 & 0.828 \\ 1.000 & 0.000 & -0.643 \end{bmatrix}$
$\{\mathbf{c_1}, \mathbf{c_2}\}_1$	$\begin{bmatrix} 0.333 & -0.386 & -0.933 \\ 3.000 & -2.100 & 0.598 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1.000 & 0.828 & -0.125 \\ 1.000 & -0.643 & 0.574 \end{bmatrix}$
$\{\mathbf{c_1}, \mathbf{c_2}\}_2$	$\begin{bmatrix} -0.933 & -0.386 & -1.567 \\ 0.598 & -2.100 & -0.690 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} -0.125 & 0.828 & -0.676 \\ 0.574 & -0.643 & 0.329 \end{bmatrix}$
$\{\mathbf{c_1}, \mathbf{c_2}\}_3$	$\begin{bmatrix} -1.567 & -0.386 & -2.200 \\ -0.676 & -2.100 & -2.759 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} -0.676 & 0.828 & -1.126 \\ 0.329 & -0.643 & -0.223 \end{bmatrix}$

$$\begin{aligned} \min_{x}\ & x_1^3 + 2x_1^2 - 5x_1 + x_2^2 - 3x_2 - 6 \\ s.t.\ & \begin{bmatrix} 2 & 1 \\ 1/2 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 5/2 \\ 3/2 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \quad \begin{matrix} (c_1) \\ (c_2) \\ (c_3) \\ (c_4) \end{matrix} \\ & \begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \\ & x \in X \subset \mathbb{R}^2 \\ & \theta \in \Theta \subset \mathbb{R}^2 \end{aligned} \tag{19}$$

The transformed and expanded mp-NLP problems are defined via the trivial recasting steps highlighted earlier (omitted for brevity). Parameters $\Delta z = 0.05$, $\delta = [0.05, 0.05, 0.05, 0.05]$, $\zeta^{edges} = 10^{-5}$ and $\zeta^{partitions} = 10^{-6}$ are set for algorithm execution. The objective function in Eq. 19 is not convex: the Hessian matrix is positive definite only when $x_1 > -2/3$. All solutions are subject to this requirement, and their validity is discussed in sequence.

4.2.1 Compact solution

The CS is delivered via Algorithm 1 and presented in Eqs. 20 and 21:

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_3 \end{bmatrix} = \begin{bmatrix} 0.573 \\ 0.393 \\ -0.786 \\ -1.500 \end{bmatrix} + \begin{bmatrix} 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \end{bmatrix} \begin{bmatrix} (1-y_1)l_1 \\ (1-y_2)l_2 \\ (1-y_3)l_3 \\ (1-y_4)l_4 \end{bmatrix} + \begin{bmatrix} -0.623 & -0.516 & 0.100 & 0.050 \\ -0.393 & -0.443 & 0.025 & 0.050 \\ 0.153 & 0.049 & -0.050 & 0.000 \\ 0.316 & 0.419 & 0.000 & -0.050 \end{bmatrix} \begin{bmatrix} y_1 l_1 \\ y_2 l_2 \\ y_3 l_3 \\ y_4 l_4 \end{bmatrix}, y \in \{0,1\}, l \geq 0 \quad (20)$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^{opt} = \begin{bmatrix} 0.786 \\ 1.500 \end{bmatrix} + \begin{bmatrix} -0.153 & -0.049 & 0.050 & 0.000 \\ -0.316 & -0.419 & 0.000 & 0.050 \end{bmatrix} \begin{bmatrix} y_1 l_1 \\ y_2 l_2 \\ y_3 l_3 \\ y_4 l_4 \end{bmatrix} \quad (21)$$

When $z_1 \ll 0.573$ or $z_2 \ll 0.393$, $x_1 < -2/3$. The convexity requirement does not hold in this case, and the CS is not valid in the context of the expanded mp-NLP (Z defined unbounded). On the other hand, all optimizers for all $z \in Z$ in the transformed mp-NLP satisfy $x_1 > -2/3$, thus preserving the convexity requirement and ensuring the validity of the CS in this context.

4.2.2 Basic explicit solution

Algorithm 2 delivers the BES. In this case, the MILPs for the 3rd and 4th active parametric edges return infeasible: constraints c_3 and c_4 are always inactive in Θ, from where only active sets $\{\}$, $\{c_1\}$, $\{c_2\}$ and $\{c_1, c_2\}$ are relevant. The list of critical regions is presented in Table 5; the general optimizer function applicable to all regions is presented in Eq. 22.

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^{opt} = \left[\begin{array}{c|cccc} 0.786 & 0.633 & 0.737 & 0.836 & 0.786 \\ 1.500 & 1.184 & 1.081 & 1.500 & 1.550 \end{array}\right] \gamma \quad (22)$$

4.2.3 Refined explicit solution

In the edge refinement stage (Algorithm 3), the 3rd and 4th edges are dismissed ($u_3^i = u_4^i = 0$). The 1^{st} and 2^{nd} edges are tested, and no additional points are required. In this case, the set of critical regions in the BES suffices (all steps in Algorithm 4, except Step 1 are dismissed). All regions are tested in the last stage (Algorithm 5). Only the initial region for active set $\{c_1\}$ requires improvement and is broken down in two smaller partitions. The optimizer functions for $\{c_1, c_2\}$ are obtained exactly (defined precisely as the interception of the two active inequality constraints). The RES is listed in Table 6 and the map of critical regions is shown in Fig. 7.

Table 5 Basic explicit solution for benchmark problem

Critical region	$1\mathbf{M}^{\mathbf{CR}}$
$\{\}$	$\begin{bmatrix} 0.573 & 1.000 & 0.000 & 0.000 & 0.000 \\ 0.393 & 0.000 & 1.000 & 0.000 & 0.000 \\ -0.786 & 0.000 & 0.000 & 1.000 & 0.000 \\ -1.500 & 0.000 & 0.000 & 0.000 & 1.000 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$
$\{\mathbf{c_1}\}$	$\begin{bmatrix} 0.573 & -0.050 & 0.000 & 0.000 & 0.000 \\ 0.393 & 0.000 & 1.000 & 0.000 & 0.000 \\ -0.786 & -0.633 & 0.000 & 1.000 & 0.000 \\ -1.500 & -1.184 & 0.000 & 0.000 & 1.000 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix}$
$\{\mathbf{c_2}\}$	$\begin{bmatrix} 0.573 & 1.000 & 0.056 & 0.000 & 0.000 \\ 0.393 & 0.000 & -0.050 & 0.000 & 0.000 \\ -0.786 & 0.000 & -0.737 & 1.000 & 0.000 \\ -1.500 & 0.000 & -1.081 & 0.000 & 1.000 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix}$
$\{\mathbf{c_1}, \mathbf{c_2}\}$	$\begin{bmatrix} 0.573 & -0.050 & 0.056 & 0.000 & 0.000 \\ 0.393 & 0.000 & -0.050 & 0.000 & 0.000 \\ -0.786 & -0.633 & -0.737 & 1.000 & 0.000 \\ -1.500 & -1.184 & -1.081 & 0.000 & 1.000 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix}$

Table 6 Refined explicit solution for benchmark problem

Critical region	$1\mathbf{M}^{\mathbf{CR}}$	$1\mathbf{M}^{\mathbf{OF}}$
$\{\}_{(1)}$	$\begin{bmatrix} 0.573 & 1.000 & 0.000 & 0.000 & 0.000 \\ 0.393 & 0.000 & 1.000 & 0.000 & 0.000 \\ -0.786 & 0.000 & 0.000 & 1.000 & 0.000 \\ -1.500 & 0.000 & 0.000 & 0.000 & 1.000 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0.786 & 0.000 & 0.000 & 0.000 & 0.000 \\ 1.500 & 0.000 & 0.000 & 0.000 & 0.000 \end{bmatrix}$
$\{\mathbf{c_1}\}_{(1)}$	$\begin{bmatrix} 0.573 & 0.261 & 0.000 & 0.000 & 0.000 \\ 0.393 & 0.195 & 1.000 & 0.000 & 0.000 \\ -0.786 & -0.711 & 0.000 & 1.000 & 0.000 \\ -1.500 & -1.340 & 0.000 & 0.000 & 1.000 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0.786 & 0.710 & 0.000 & 0.000 & 0.000 \\ 1.500 & 1.340 & 0.000 & 0.000 & 0.000 \end{bmatrix}$
$\{\mathbf{c_1}\}_{(2)}$	$\begin{bmatrix} 0.261 & -0.050 & 0.000 & 0.000 & 0.000 \\ 0.195 & 0.000 & 1.000 & 0.000 & 0.000 \\ -0.711 & -0.633 & 0.000 & 1.000 & 0.000 \\ -1.340 & -1.184 & 0.000 & 0.000 & 1.000 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0.710 & 0.633 & 0.000 & 0.000 & 0.000 \\ 1.340 & 1.184 & 0.000 & 0.000 & 0.000 \end{bmatrix}$
$\{\mathbf{c_2}\}_{(1)}$	$\begin{bmatrix} 0.573 & 1.000 & 0.056 & 0.000 & 0.000 \\ 0.393 & 0.000 & -0.050 & 0.000 & 0.000 \\ -0.786 & 0.000 & -0.737 & 1.000 & 0.000 \\ -1.500 & 0.000 & -1.081 & 0.000 & 1.000 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0.786 & 0.000 & 0.737 & 0.000 & 0.000 \\ 1.500 & 0.000 & 1.081 & 0.000 & 0.000 \end{bmatrix}$
$\{\mathbf{c_1}, \mathbf{c_2}\}_{(1)}$	$\begin{bmatrix} 0.573 & -0.050 & 0.056 & 0.000 & 0.000 \\ 0.393 & 0.000 & -0.050 & 0.000 & 0.000 \\ -0.786 & -0.633 & -0.737 & 1.000 & 0.000 \\ -1.500 & -1.184 & -1.081 & 0.000 & 1.000 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0.786 & 0.633 & 0.737 & 0.000 & 0.000 \\ 1.500 & 1.184 & 1.081 & 0.000 & 0.000 \end{bmatrix}$

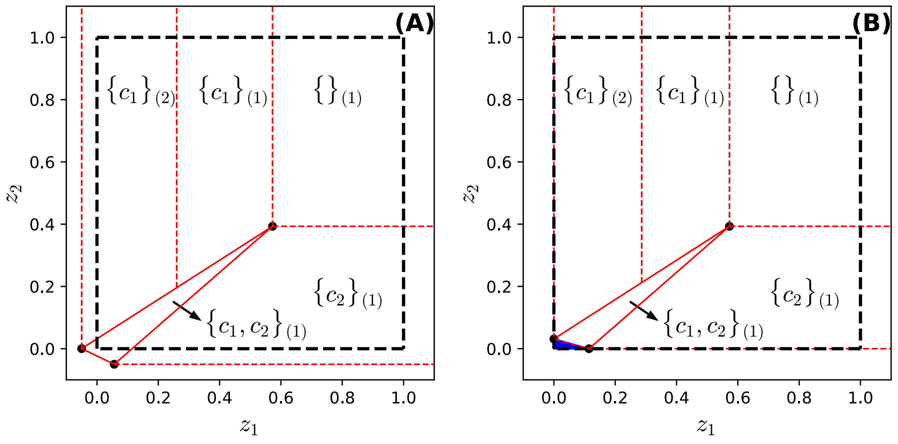

Fig. 7 Map of critical regions for benchmark mp-NLP (RES): (**A**) $\delta = [0.05, 0.05, 0.05, 0.05]$: the full parameters space is covered by one of the defined regions; (**B**) $\delta = [0, 0, 0, 0]$: illustrates the significance of δ to this example, and how a small region highlighted in blue would not be included in any of the regions calculated. Both maps display only axes z_1 and z_2: for all $z \in Z$, constraints c_3 and c_4 are always inactive enabling a 2D critical region representation

Table 7 Solution accuracy of benchmark problem via mp-NLP algorithms

Algorithm	Optimal value function	Optimization variables	Parameter space partition
mp-OA	$\surd$	X	X
AM	$\surd$	$\surd$	X
mp-QA	$\surd$	$\surd$	$\surd$
GVS	$\surd$	$\surd$	$\surd$
CS	$\surd$	$\surd$	$\surd$
BES	$\surd$	$\surd$	$\surd$
RES	$\surd$	$\surd$	$\surd$

4.2.4 Comparison with state-of-the art mp-NLP algorithms

The solution of the benchmark problem is reported in Domínguez et al. (2010) using the four state-of-the-art mp-NLP algorithms highlighted in the Introduction: (i) mp-OA (ii) AM, (iii) mp-QA, and (iv) GVS. This includes all or some representative solution fragments and the corresponding maps of critical regions. For brevity, we focus only on solution accuracy and computational complexity as reported in Domínguez et al. (2010) and compare them with Algorithm 1 (CS), Algorithm 2 (BES), and Algorithms 3–5 (RES) as shown in Tables 7 and 8, respectively.

Since the objective function of the benchmark problem displays a mild nonlinearity, the CS on its own is a very accurate solution and requires minimal computational effort in this case. Specifically, for all $z \in Z$, feasible solutions are obtained, with negligible optimality loss, and where the critical regions' map in Fig. 7 is very similar to those reported for mp-QA and GVS in Domínguez et al. (2010). The additional refinements

Table 8 Computational complexity of benchmark problem via mp-NLP algorithms

Algorithm	mp-LPs	mp-QPs	LPs	NLPs	MILPs
mp-OA	12	0	0	276	0
AM	0	0	0	71	0
mp-QA	0	1	0	9	0
GVS	0	0	0	418	0
CS	0	0	2	$3^{(a)}$	0
BES	0	0	4	0	8
RES	0	0	5	$9^{(a)}$	0
(total)	0	0	11	$12^{(a)}$	8

$^{(a)}$ NLPs via KKT conditions. Details in Sect. 3.4

in BES and RES deliver explicit solutions with marginal accuracy gains with respect to the CS, where computational complexity is similar to mp-QA, though not requiring the solution of any auxiliary mp-QP problem.

4.3 Computational study

Two sets of mp-NLPs were created and solved to obtain additional insights into the performance of Algorithms 1–5. In all mp-NLPs $n = p$; the 2 sets of examples include 10 mp-NLPs with increasing size: $p = 1, \ldots, 10$. In the first set, all mp-NLPs were created such that all (2^p) active sets define a critical region in the parameter space; in the second set of examples, all constraints are inactive, except a single constraint, thus yielding in all mp-NLPs a total of 2 solution fragments. All mp-NLPs are included in https://github.com/diogonarciso/mpNLP. Given the large variability in solution complexity for mp-NLP problems of similar dimensions, the authors believe that this approach facilitates a fairer comparison of performance. Figure 8 displays the results on computational performance; all examples were solved using a 2.80GHz 11th Gen Intel(R) Core(TM) i7-1165G7 machine (16Gb of RAM).

The computational times to obtain the CS are the same in the 2 sets of examples. This is expected since regardless of the number of active sets included in the solution, only p LPs and $p + 1$ NLPs must be solved. Complexity increases in an almost linear fashion with p; this is not strictly the case, since in these sets of examples the larger is p the larger is also the LPs and NLPs to be solved. Obtaining the BES requires the most significant effort in comparison with both the CS and RES. This is a result of solving the $2p$ the MILPs, an aspect of Algorithm 2 which can still be improved for performance. Yet, it is clear that the computational dependency with p is also of an almost linear nature (larger MILPs also require more computational time). In the second set, it is visible that computation times become increasingly shorter than their couterparts in the first set of examples: in this case only 2 LPs must be solved, and thus alleviating the computtional burden. The computational times to obtain the RES are sharply different between the first and second set: in the first case all 2^p active sets include a critical region in the parameter space and an exponential dependency with p is observed; this case exceeds the computational times required for the BES at

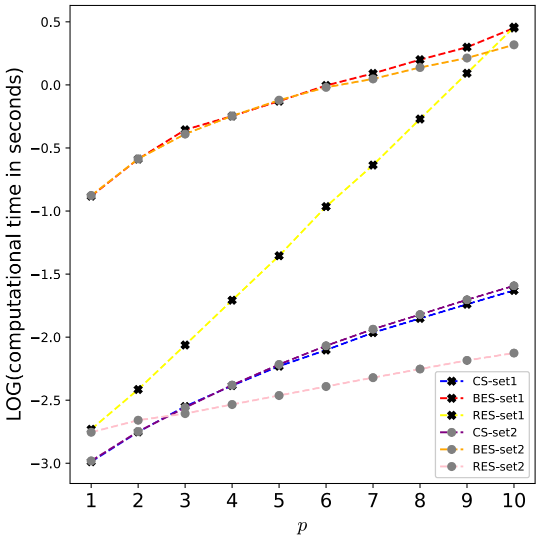

Fig. 8 Computational times required to solve the 2 sets of 10 mp-NLPs

around $p = 10$. In the second set, Algorithms 3–5 explore more efficiently the outputs of the $2p$ MILPS in Algorithm 2: since only 2 active sets are part of the solutions in all examples, computational times are much smaller than their couterparts in the first set, and which suggest a nearly linear dependency with p.

5 Concluding remarks

A new solution strategy for mp-NLP is proposed in this work. The critical step enabling these developments is parameter transformation $z = F\theta$: in the transformed parameters space, all critical regions share a unique vertex and are bounded by a set of $2p$ 1-dimensional edges, defining the backbone of critical region maps. This enables the development of an enhanced partitioning strategy where critical region maps are obtained more consistently with the actual optimal active sets detected in the parameters space, optimizer sensitivities are captured more accurately, and overall promoting low algorithm and solution complexity.

While the CS enables the calculation of optimizers for all $z \in \mathbb{R}^p$, this solution is of limited applicability in the context of the expanded mp-NLP; since optimizer edges are nonlinear, significant deviations to the actual optimal optimizers are expected when $z \ll z^*$. The expanded mp-NLP is here presented mainly as a *theoretical* extension of the mp-QP case in Narciso et al. (2022). In practical terms, this work is focused mainly on obtaining the solution of the transformed mp-NLP, which is a more realistic objective for this class of problems.

Three routes are presented to deliver the solution to transformed mp-NLP problems, which are built incrementally for improved accuracy. The CS is the first of these, where optimizer sensitivities are estimated from a set of representative points within the parameters space to promote accuracy. This is generally a good first estimation, provided that optimizer edges are mildly nonlinear. The BES enables a fully declarative presentation of all solution fragments, more consistent with the standard practice in the field. The solution includes all full-dimensional critical regions detected in the parameters space, now subject to convex-hull restrictions (Eq. 9) to prevent $z \ll z^*$. A single optimizer function delivers the optimizers for all critical regions. We remark that the BES offers important insights from the analysis of their maps of critical regions, as illustrated in Fig. 5: in {}, no constraint is active, and thus the matching optimizer function is insensitive to z_1 and z_2; its critical region extends to infinity since all their points denote an *extension* of the feasible space beyond z^* with no impact on the associated optimizers. An equivalent analysis may be undertaken for all active sets in mp-NLP problems of any size.

The key drawback of the CS and BES resides in the fact that no steps are enforced to check and improve solution accuracy. The RES deals with this limitation, by first collecting a set of additional points per edge (if required) and creating an initial set of partitions from these points. These are then assessed and broken down into smaller partitions to improve accuracy. Only the set of sensitive parameters is used in this process for all critical regions, which favors low algorithm and solution complexity while offering the same kind of interpretation of critical regions as discussed above.

The present research path was initiated in Narciso (2009), where the author examines the limitations of the state-of-the-art mp-NLP algorithms. It is pointed out that the construction of critical region maps in these algorithms generally bears little relevance to the actual active set maps, and thus contributes to solution complexity. The GVS algorithm was the first attempt at solving this problem. All other mp-NLP algorithms also derive solutions while partitioning the original parameters space (Θ), and thus can not explore the convenient transformation $z = F\theta$. This is the critical paradigm shift presented in this work. It enables more efficient routes for the calculation of mp-NLP problems and a more intuitive analysis of their solutions, where each coordinate of z relates directly to a contraction/expansion of the feasible space via the corresponding inequality constraint.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s11081-024-09888-2.

Acknowledgements Not applicable.

Author Contributions Conceptualization: DN and EP; formal analysis and investigation: DN; methodology: DN; software: DN; validation: DN; supervision: EP; writing—original draft preparation: DN and EP; writing—review and editing: DN and EP

Funding Open access funding provided by FCT|FCCN (b-on). This work was supported by national funds through FCT/MCTES (PIDDAC): LEPABE, UIDB/00511/2020 (DOI: 10.54499/UIDB/00511/2020) and UIDP/00511/2020 (DOI: 10.54499/UIDP/00511/2020), ALiCE, LA/P/0045/2020 (DOI: 10.54499/LA/P/0045/2020) and CERENA, UIDB/04028/2020. The suppot of the Texas A&M Energy Institute is also gratefully acknowledged.

Availability of data and materials Supplementary materials for this work include Appendix 1 and the Python codes in https://github.com/diogonarciso/mpNLP

Declarations

Conflict of interest The authors declare that they have no Conflict of interest.

Ethics approval and Consent to participate Not applicable.

Consent for publication The authors consent to publish this paper in Springer's Optimization and Engineering journal.

References

Bemporad A, Filippi C (2006) An algorithm for approximate multiparametric convex programming. Comput Optim Appl 35:87–108. https://doi.org/10.1007/s10589-006-6447-z

Bynum ML, Hackebeil GA, Hart WE et al (2021) Pyomo: optimization modelling in Python, vol 67. Springer, Berlin. https://doi.org/10.1007/978-3-030-68928-5

Domínguez LF, Narciso DAC, Pistikopoulos EN (2010) Recent advances in multiparametric nonlinear programming. Comput Chem Eng 34:707–716. https://doi.org/10.1016/j.compchemeng.2009.10.012

Dua V, Pistikopoulos EN (1999) Algorithms for the solution of multiparametric mixed-integer nonlinear optimization problems. Ind Eng Chem Res 38:3976–3987. https://doi.org/10.1021/ie980792u

Dua V, Bozinis A, Pistikopoulos EN (2002) A multiparametric programming approach for mixed-integer quadratic engineering problems. Comput Chem Eng 26:715–733. https://doi.org/10.1016/S0098-1354(01)00797-9

Gal T, Nedoma J (1972) Multiparametric linear programming. Math Prog Stud, 18, 406–422. https://www.jstor.org/stable/2629358

Gupta A, Bhartiya S, Nataraj PSV (2011) A novel approach to multiparametric quadratic programming. Automatica 47:2112–2117. https://doi.org/10.1016/j.automatica.2011.06.019

Harris CR, Millman KJ, van der Walt SJ et al (2020) Array programming with NumPy. Nature 585:357–362. https://doi.org/10.1038/s41586-020-2649-2

Johansen TA (2002) On multi-parametric nonlinear programming and explicit nonlinear model predictive control. In: Proceedings of the 41st IEEE conference on decision and control. https://ieeexplore.ieee.org/document/1184260

Mate S, Bhartiya S, Nataraj PSV (2020) Multiparametric nonlinear MPC: a region free approach. IFAC-PapersOnLine 53(2):11374–11379. https://doi.org/10.1016/j.ifacol.2020.12.548

Narciso D (2009) Developments in multiparametric parametric programming and control. PhD thesis, Department of Chemical Engineering and Chemical Technology Imperial College of Science, Technology and Medicine London, U.K

Narciso DAC, Pappas I, Martins FG, Pistikopoulos EN (2022) A new solution strategy for multiparametric quadratic programming. Comput Chem Eng 164:107882. https://doi.org/10.1016/j.compchemeng.2022.107882

Narciso DAC, Kenefake D, Akundi SS, Martins FG, Pistikopoulos EN (2023) A new framework and online solution engines for multiparametric Model Predictive Control. Comput Aided Chem Eng 52:1229–1234

Pistikopoulos EN, Dua V, Bozinis NA, Bemporad A, Morari M (2002) On-line optimization via off-line parametric optimization tools. Comput Chem Eng 26:175–185. https://doi.org/10.1016/S0098-1354(01)00739-6

Pistikopoulos EN, Diangelakis NA, Oberdieck R (2020) Multi-parametric optimization and control. Wiley, Hoboken. https://onlinelibrary.wiley.com/doi/book/10.1002/9781119265245

Pistikopoulos EN, Georgiadis MC, Dua V (2007) Multi-parametric programming: theory, algorithms, and applications, vol. 1. Wiley-VCH, Weinheim. https://doi.org/10.1002/9783527631216

Van Rossum G, Drake, FL (2009) Python 3 reference manual. CreateSpace, Scotts Valley. https://doi.org/10.5555/1593511

Virtanen P et al (2020) SciPy 1.0: fundamental algorithms for scientific computing in python. Nat Methods 17:261–272. https://doi.org/10.1038/s41592-019-0686-2