Contemporary Mathematics

Projection Methods: Swiss Army Knives for Solving Feasibility and Best Approximation Problems with Halfspaces

Heinz H. Bauschke and Valentin R. Koch

Abstract. We model a problem motivated by road design as a feasibility problem. Projections onto the constraint sets are obtained, and projection methods for solving the feasibility problem are studied. We present results of numerical experiments which demonstrate the efficacy of projection methods even for challenging nonconvex problems.

1. Introduction and motivation

1.1. The abstract formulation of the problem. Throughout this paper, we assume that $n \in \{2, 3, \dots\}$ and that

$$X = \mathbb{R}^n \text{ with standard inner product } \langle \cdot, \cdot \rangle \text{ and induced norm } \| \cdot \|. \tag{1}$$

We also assume we are given n strictly increasing breakpoints on the real line:

$$t = (t_1, \dots, t_n) \in X \quad \text{such that } t_1 < \cdots < t_n. \tag{2}$$

Our goal is to

$$\text{find a vector } x = (x_1, \dots, x_n) \in X \tag{3a}$$

such that

$$(t_1, x_1), \dots, (t_n, x_n) \text{ satisfies a given set of constraints.} \tag{3b}$$

1991 *Mathematics Subject Classification.* Primary 65K05, 90C25; Secondary 41A65, 49M45, 90C05.

Key words and phrases. best approximation, convex set, curve fitting, Douglas–Rachford splitting algorithm, Dykstra's method, feasibility, halfspace, interpolation, linear inequalities, method of cyclic projections, projection, road design, superiorization.

The authors thank an anonymous referee as well as Yair Censor and Gabor Herman for their careful reading, their constructive comments, and for referring us to [**9, 14, 27, 28, 33, 40, 49, 52, 54, 57, 59**]. HHB was partially supported by the Natural Sciences and Engineering Research Council of Canada (Individual Discovery Grant and Accelerator Supplement) and by the Canada Research Chair Program.

Note that for every $x \in X$, the pair $(t,x) \in X \times X$ induces a corresponding piecewise linear function or *linear spline* (see [**34**] and [**61**])

$$f_{(t,x)}\colon [t_1,t_n] \to \mathbb{R}\colon \tau \mapsto x_i + (x_{i+1}-x_i)\frac{\tau - t_i}{t_{i+1}-t_i}, \tag{4}$$

where $\tau \in [t_i, t_{i+1}]$ and $i \in \{1,\ldots,n-1\}$, which passes through the points $\big\{(t_i,x_i) \in \mathbb{R}^2 \mid i \in I\big\}$.

The set of constraints mentioned in (3b) will involve the function $f_{(t,x)}$. Let us list several types of constraints which are motivated in Section 1.2 below:

- **interpolation constraints:** For a given subset I of $\{1,\ldots,n\}$, the entries x_i are prescribed: $(\forall i \in I)\ f_{(t,x)}(t_i) = y_i$. This is an *interpolation problem* for the points $\big\{(t_i,y_i) \in \mathbb{R}^2 \mid i \in I\big\}$.
- **slope constraints:** For a given subset I of $\{1,\ldots,n-1\}$ and for every $i \in I$, the *slope*

$$s_i := \frac{x_{i+1}-x_i}{t_{i+1}-t_i} \tag{5}$$

  of $f_{(t,x)}|_{[t_i,t_{i+1}]}$ must lie in a given subset of $\mathbb{R}$.
- **curvature constraints:** For a given subset I of $\{1,\ldots,n-2\}$ and for every $i \in I$, $|s_{i+1}-s_i|$, the distance between the slopes s_i and s_{i+1} of two adjacent intervals $[t_i,t_{i+1}]$ and $[t_{i+1},t_{i+2}]$, must lie in a given subset of $\mathbb{R}$.

1.2. A concrete instance in road design. Problem (3) introduced above and its solutions have several direct applications in engineering and computer-assisted design. For instance, an engineer may want to verify the feasibility of a design, or adapt the design according to the constraints. Examples drawn from Computer-Assisted Design (CAD) include designs for roadway profiles, pipe layouts, fuel lines in automotive designs such as cars and airplanes, overhead power lines, chairlifts, cable cars, and duct networks.

Our primary motivation for this work is automated design of *road alignments*. A road alignment is represented by the centerline of the road, which is idealized as a (generally) nonlinear, smooth curve in $\mathbb{R}^3$. To facilitate construction drawings, civil engineers reduce the three-dimensional road design to two two-dimensional parts, horizontal and vertical.

The horizontal alignment is the plan (or map) view of the road. In the vertical view, the *ground profile* $g\colon [t_1,t_n] \to \mathbb{R}$ shows the elevation values of the existing ground along the centerline (see the brown curve in Figure 1). Since earthwork operations such as cuts and fills are expensive items in road construction, a civil engineer tries to find a road profile represented by a linear spline $f_{(t,x)})$ that follows g as closely as possible.

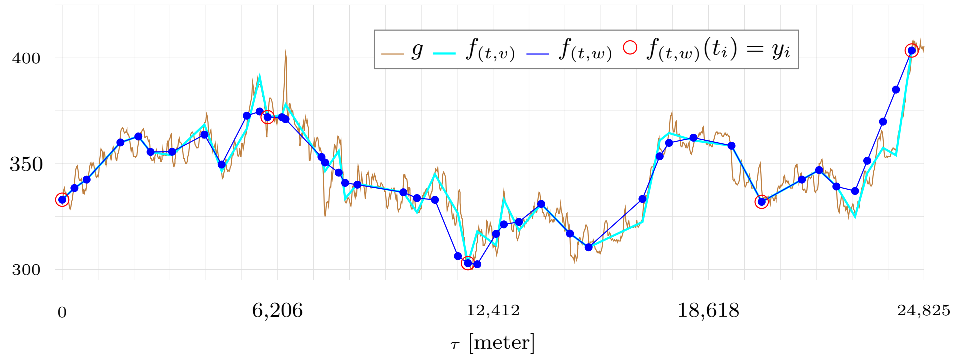

FIGURE 1. Vertical profiles of ground profile g, initial road design $f_{(t,v)}$, and final road design $f_{(t,w)}$ for a highway design near Kelowna, B.C., Canada. The solution was found with the ParDyk algorithm (Algorithm 4.5 below) using a design speed of 130 km/h, and a maximum slope of 4%. (These engineering constraints translate into a specific set of slope and interpolation constraints.)

Design constraints imposed on $f_{(t,x)}$ by the engineer or by civil design standards such as those published by the *American Association of State Highway and Transportation Officials (AASHTO)* [**1**] include the following:

- At a certain station t_i, the engineer may have to fix the elevation x_i to allow for construction of an intersection with an existing road that crosses the new road at t_i. This corresponds to the (mathematically *affine*) interpolation constraint mentioned in Section 1.1.
- For safety reasons and to ensure good traffic flow, AASHTO requires that the slope between two stations t_i and t_{i+1} is bounded above and below. These are the (mathematically *convex*) slope constraints of Section 1.1.
- In a road profile, engineers fit a (usually parabolic) curve at the point of intersection of two line segments. The curvature depends on the grade change of the line segments and influences the vertical acceleration of a vehicle, as well as the stopping sight distance. AASHTO requires bounds on the slope change. This corresponds to the (mathematically *convex*) curvature constraint in Section 1.1.
- In some cases, the engineer requires a minimum drainage grade to allow flow and avoid catchment of storm water. These (mathematically challenging *nonconvex*) slope constraints are discussed in Section 5.1 below.

We denote the starting spline for a road profile (see the cyan curve in Figure 1) by

$$f_{(t,v)}, \quad \text{where } t = (t_1, \ldots, t_n) \in X \text{ and } v = (v_1, \ldots, v_n) \in X. \tag{6}$$

In practice, the starting spline could simply be the connected line segments for the interpolation constraint, or it could be generated from the ground profile g by using Bellman's method (see [**15**] and [**55**] for details). In either case, we assume that t is given and fixed, and that we need to decide whether or not $f_{(t,v)}$ is feasible with respect to the aforementioned constraints. If v leads to a feasible spline, then we are done; otherwise, we wish to find $w \in X$ such that the new road spline $f_{(t,w)}$ (see the blue curve in Figure 1) satisfies the design constraints. Ideally, $f_{(t,w)}$ is close to the ground profile represented by (t, v). Finally, if there is no $w \in X$ making the problem feasible, then we would like to detect this through some suitable measure.

1.3. Main results and organization of the paper. We now summarize the main contributions of this paper.

- In principle, there are numerous constraints to deal with for problem (3) in the context of road design. Fortunately, the constraints have a lot of structure we can take advantage of and we demonstrate that *the constraints parallelize which allows to reduce the problem to six constraint sets* (see Section 2.5) each of which admits a closed form projection formula (see (12), (21), and (31), (99)).
- We provide a selection of *state-of-the-art projection methods, superiorization algorithms, and best approximation algorithms* (see Sections 3 and 4), and adapt them to the road design problem.
- We *present various observations on the algorithms and their relationships* (see Remark 3.14, Remark 4.7, Remark 4.15, Remark 4.16 and Example 5.1.)
- We report on broad *numerical experiments* (see Section 6) introducing for the first time *performance profiles* for projection methods.
- Based on the numerical experiments, we recommend CycIP, which is an intrepid form of the method of cyclic projections, as an overall good choice for solving feasibility and best approximation problems.

The remainder of the paper is organized as follows. Section 2 contains a detailed analysis of the projection operators encountered in the road design problem. We take advantage of aggregating constraints and derive simple formulas. Projection methods for feasibility problems are reviewed in Section 3. Because we are working with more than two constraint sets, we adapt to this situation by working in a product space if needed. If more than just a feasible road design is desired, then the engineer has to consider optimization algorithms. We review two types of such methods (superiorization and best approximation algorithms) and adapt them to our problem in Section 3.5 and Section 4 respectively. Nonconvex constraints are investigated in Section 5. We report on numerical experiments in Section 6. The final Section 7 concludes the paper.

For notation and general references on the mathematics underlying projection methods, we refer the reader to the books [**6**], [**19**], [**26**], [**35**], [**43**], and [**44**].

2. Constraints and projection operators

2.1. The projection onto a general convex set. In this section, we make the constraints encountered in road design mathematically precise. Almost all of these constraints turn out to lead to sets that are *convex* and closed. Recall that a set C is *convex* if it contains all line segments between each pair taken from C:

$$(\forall c_0 \in C)(\forall c_1 \in C)(\forall \lambda \in [0,1]) \quad (1-\lambda)c_0 + \lambda c_1 \in C. \tag{7}$$

If C is a nonempty closed convex subset of X, then for every $x \in X$, the optimization problem

$$d(x, C) := \min_{c \in C} \|x - c\|, \tag{8}$$

which concerns the computation of the *distance* $d(x, C)$ from x to the set C, has a *unique* solution, denoted by $P_C x$ and called the *projection* of x onto C. The vector $P_C x$ is characterized by two properties, namely

$$P_C x \in C \quad \text{and} \quad (\forall c \in C) \quad \langle c - P_C x, x - P_C x \rangle \leq 0. \tag{9}$$

(For a proof, see, e.g., [**6**, Theorem 3.14]). The induced operator $P_C \colon X \to C$ is the *projection operator* or *projector* onto C. There are several examples that allow us to write down the projector in closed form or to approximate P_C provided C is the intersection of finitely many simple closed convex sets (see, e.g., [**6**, Chapters 28 and 29].) In the road design application, it is fortunately possible to obtain explicit formulas; in the following subsections, we will make these formulas as convenient as possible for software implementation. Let us do this for each of the three types of constraints.

2.2. Interpolation constraints. We assume that I is a set such that

$$\{1, n\} \subseteq I \subseteq \{1, 2, \ldots, n\}, \quad \text{and} \quad y = (y_i) \in \mathbb{R}^I \tag{10}$$

is given. Set

$$Y := \big\{x = (x_1, \ldots, x_n) \in X \mid (\forall i \in I) \;\; x_i = y_i\big\}. \tag{11}$$

The closed set Y is an *affine subspace*, i.e., $(\forall y \in Y)(\forall z \in Y)(\forall \lambda \in \mathbb{R})\ (1-\lambda)y+\lambda z \in Y$; in particular, Y is convex. For convenience, we record the explicit formula for the projection onto Y.

Proposition 2.1 (interpolation constraint projector). *The projector onto Y is given by*

$$P_Y \colon X \to X \tag{12a}$$

$$(x_1, x_2, \ldots, x_n) \mapsto (c_1, c_2, \ldots, c_n), \quad \textit{where} \quad c_i = \begin{cases} y_i, & \textit{if } i \in I; \\ x_i, & \textit{if } i \in \{1, 2, \ldots, n\} \smallsetminus I. \end{cases} \tag{12b}$$

2.3. Slope constraints.

2.3.1. *A useful special case.* We start with a simple special case that will also be useful in handling our general slope constraints. To this end, let $i \in \{1, 2, \ldots, n-1\}$. The constraint set S_i imposes that the absolute value of the slope $f_{(t,x)}$ for the interval $[t_i, t_{i+1}]$ is bounded above, i.e., there exists $\alpha_i \geq 0$ such that

$$S_i := \big\{x = (x_1, \ldots, x_n) \in \mathbb{R}^n \mid |x_{i+1} - x_i| \leq \alpha_i\big\}. \tag{13}$$

Indeed, if the actual maximum absolute slope is $\sigma_i \geq 0$, then, setting $\alpha_i = \sigma_i |t_{i+1} - t_i|$, we see that

$$\left|\frac{x_{i+1} - x_i}{t_{i+1} - t_i}\right| \leq \sigma_i \Leftrightarrow |x_{i+1} - x_i| \leq \alpha_i \Leftrightarrow \pm(x_{i+1} - x_i) \leq \alpha_i. \tag{14a}$$

$$\Leftrightarrow \pm\langle e_{i+1} - e_i, x\rangle \leq \alpha_i \tag{14b}$$

$$\Leftrightarrow -\alpha_i \leq \langle e_{i+1} - e_i, x\rangle \leq \alpha_i, \tag{14c}$$

where e_i and e_{i+1} denote the standard unit vectors in X (with the number 1 in position i and $i + 1$, respectively, and zeros elsewhere). The last characterization reveals that S_i is the intersection of two halfspaces whose boundary hyperplanes are orthogonal to $e_{i+1} - e_i$. In particular, S_i is a closed convex subset of X. Using e.g., [**6**, Example 28.17], we obtain for every $x \in X$,

$$P_{S_i}x = \begin{cases} x + \dfrac{-\alpha_i - \langle e_{i+1} - e_i, x\rangle}{\|e_{i+1} - e_i\|^2}(e_{i+1} - e_i), & \text{if } \langle e_{i+1} - e_i, x\rangle < -\alpha_i; \\ x, & \text{if } -\alpha_i \leq \langle e_{i+1} - e_i, x\rangle \leq \alpha_i; \\ x + \dfrac{\alpha_i - \langle e_{i+1} - e_i, x\rangle}{\|e_{i+1} - e_i\|^2}(e_{i+1} - e_i), & \text{if } \langle e_{i+1} - e_i, x\rangle > \alpha_i. \end{cases} \tag{15}$$

This formula shows that $(P_{S_i}x)_j = x_j$ for every $j \in I \smallsetminus \{i, i+1\}$. Thus, the only entries that possibly change after executing P_{S_i} are in positions i and $i+1$; after some simplification, we obtain for these entries the formula

$$\big((P_{S_i}x)_i, (P_{S_i}x)_{i+1}\big) = \begin{cases} \tfrac{1}{2}(x_i + x_{i+1} + \alpha_i, x_i + x_{i+1} - \alpha_i), & \text{if } x_i - x_{i+1} > \alpha_i; \\ (x_i, x_{i+1}), & \text{if } |x_{i+1} - x_i| \leq \alpha_i; \\ \tfrac{1}{2}(x_i + x_{i+1} - \alpha_i, x_i + x_{i+1} + \alpha_i), & \text{if } x_{i+1} - x_i > \alpha_i. \end{cases} \tag{16}$$

We note that

$$x \notin S_i \quad \Rightarrow \quad (P_{S_i}x)_i \neq x_i \text{ and } (P_{S_i}x)_{i+1} \neq x_{i+1}; \tag{17}$$

furthermore, if $\alpha_i = +\infty$, i.e., no slope constraint, then (16) is valid as well.

2.3.2. *The general case.* Now we turn to the general case. We assume the existence of a vector $a = (\alpha_i) \in \mathbb{R}^{n-1}_+$ such that the constraint set is

$$S = \bigcap_{i \in \{1,\ldots,n-1\}} S_i = \big\{(x_1, \ldots, x_n) \in \mathbb{R}^n \mid (\forall i \in \{1, \ldots, n-1\}) \;\; |x_{i+1} - x_i| \leq \alpha_i\big\}, \tag{18}$$

where S_i is defined in (13). While we obtained an explicit formula to deal with a single slope constraint (see (16)) we are unaware of a corresponding formula for P_S. Furthermore, since P_{S_i} possibly modifies the vector in positions i and $i+1$ (but not elsewhere), we *cannot* use (16) for the sets S_i and S_{i+1} concurrently because their projections possibly modify positions $(i, i+1)$ and $(i+1, i+2)$ (see (17)), but not necessarily in a consistent manner at position $i+1$! However, by combining the $n-1$ slope constraints according to parity of indices, i.e., by setting

$$S_{\text{even}} := \bigcap_{i \in \{1,\ldots,n-1\} \cap (2\mathbb{N})} S_i, \quad \text{and} \quad S_{\text{odd}} := \bigcap_{i \in \{1,\ldots,n-1\} \cap (1+2\mathbb{N})} S_i, \tag{19}$$

we see that

$$S = S_{\text{even}} \cap S_{\text{odd}} \tag{20}$$

can be written as the intersection of just *two* constraint sets! Furthermore, (16) yields the *fully parallel* update formulas:

PROPOSITION 2.2 (convex slope constraint projector). *For every $x \in X$, the projectors onto S_{even} and S_{odd} are given by*

$$P_{S_{\text{even}}}x = \big(x_1, (P_{S_2}x)_2, (P_{S_2}x)_3, (P_{S_4}x)_4, (P_{S_4}x)_5, \ldots\big) \in X, \tag{21a}$$

where the last entry in (21a) *is x_n if n is even; and*

$$P_{S_{\text{odd}}}x = \big((P_{S_1}x)_1, (P_{S_1}x)_2, (P_{S_3}x)_3, (P_{S_3}x)_4, \ldots\big) \in X, \tag{21b}$$

where the last entry in (21b) *is x_n if n is odd.*

The constraints making up the aggregated slope constraint are very special polyhedra, namely *"strips"*, i.e., the intersection of two halfspaces with opposing normal vectors. This motivates the technique, which originates in [**51**] (see also [**20**], [**21**], [**27**], [**49**], and [**50**]), of not just projecting onto these sets but rather *inside* them: either we reflect *into* the strip or (if we are too distant from the strip) we jump to the corresponding midpoint of the strip. Let us record the formula for this operator.

PROPOSITION 2.3 (intrepid slope constraint projectors). *The intrepid counterpart of* (16)*, is*[1]

(22)
$$(x_i, x_{i+1}) \mapsto \begin{cases} \frac{1}{2}(x_i + x_{i+1}, x_i + x_{i+1}), & \text{if } |x_i - x_{i+1}| > 2\alpha_i; \\ (x_i, x_{i+1}), & \text{if } |x_{i+1} - x_i| \le \alpha_i; \\ (x_{i+1}, x_i) + \operatorname{sgn}(x_{i+1} - x_i)(-\alpha_i, \alpha_i), & \text{if } \alpha_i < |x_i - x_{i+1}| < 2\alpha_i. \end{cases}$$

These operators lead to the intrepid counterpart of (21)*.*

2.4. Curvature constraints.

2.4.1. *A useful special case.* Again, let us start with a simple special case that will also be useful in handling our general curvature constraints. To this end, let $i \in \{1, \ldots, n-2\}$. The constraint requires that the difference of consecutive slopes is bounded above and below. Hence there exists $\gamma_i \in \mathbb{R}$ and $\delta_i \in \mathbb{R}$ such that $\gamma_i \ge \delta_i$ and

$$C_i := \left\{ x = (x_1, \ldots, x_n) \in \mathbb{R}^n \;\middle|\; \gamma_i \ge \frac{x_{i+2} - x_{i+1}}{t_{i+2} - t_{i+1}} - \frac{x_{i+1} - x_i}{t_{i+1} - t_i} \ge \delta_i \right\}. \tag{23}$$

Set

$$\tau_i := t_{i+1} - t_i, \quad \tau_{i+1} := t_{i+2} - t_{i+1}, \quad \text{and} \quad u_i := \tau_{i+1} e_i - (\tau_i + \tau_{i+1}) e_{i+1} + \tau_i e_{i+2}. \tag{24}$$

Then $\tau_i > 0$, $\tau_{i+1} > 0$, and for every $x \in X$,

$$x \in C_i \quad \Leftrightarrow \quad \gamma_i \tau_{i+1} \tau_i \ge \langle u_i, x \rangle \ge \delta_i \tau_{i+1} \tau_i. \tag{25}$$

Again, we see that C_i is a closed *convex* polyhedron and by e.g. [**6**, Example 28.17], we obtain

$$P_{C_i} \colon x \mapsto \begin{cases} x + \dfrac{\delta_i \tau_i \tau_{i+1} - \langle u_i, x \rangle}{\|u_i\|^2} u_i, & \text{if } \langle u_i, x \rangle < \delta_i \tau_i \tau_{i+1}; \\ x, & \text{if } \delta_i \tau_i \tau_{i+1} \le \langle u_i, x \rangle \le \gamma_i \tau_i \tau_{i+1}; \\ x + \dfrac{\gamma_i \tau_i \tau_{i+1} - \langle u_i, x \rangle}{\|u_i\|^2} u_i, & \text{if } \gamma_i \tau_i \tau_{i+1} < \langle u_i, x \rangle. \end{cases} \tag{26}$$

Since $u_i \in \operatorname{span}\{e_i, e_{i+1}, e_{i+2}\}$, it follows that

$$\big\{ j \in \{1, \ldots, n\} \mid x_j \ne (P_{C_i} x)_j \big\} \subseteq \{i, i+1, i+2\}. \tag{27}$$

---

[1] Here sgn denotes the signum function.

2.4.2. *The general case.* Now we turn to the general curvature constraints. We assume the existence of a vector $c = (\gamma_i) \in \mathbb{R}^{n-2}$ and $d = (\delta_i) \in \mathbb{R}^{n-2}$ such that the constraint set is

(28a)
$$C = \bigcap_{i \in \{1,\ldots,n-2\}} C_i$$
(28b)
$$= \bigg\{ (x_1, \ldots, x_n) \in X \;\bigg|\; (\forall i \in \{1, \ldots, n-2\})\ \gamma_i \geq \frac{x_{i+2} - x_{i+1}}{t_{i+2} - t_{i+1}} - \frac{x_{i+1} - x_i}{t_{i+1} - t_i} \geq \delta_i \bigg\}.$$

Because of (27), we can and do aggregate these $n-2$ constraints into *three* constraint sets that allow projections in closed form. To this end, we set

$$(\forall j \in \{1,2,3\}) \quad C_{[j]} := \bigcap_{i \in \{1,\ldots,n-2\} \cap (j+3\mathbb{N})} C_i \tag{29}$$

so that

$$C = C_{[1]} \cap C_{[2]} \cap C_{[3]}. \tag{30}$$

Combined with (26), we obtain the following.

Proposition 2.4 (curvature constraint projector). *For every $x \in X$, the projectors onto $C_{[1]}$, $C_{[2]}$, and $C_{[3]}$ are given*

(31a)
$$P_{C_{[1]}} x = \big((P_{C_1}x)_1, (P_{C_1}x)_2, (P_{C_1}x)_3, (P_{C_4}x)_4, (P_{C_4}x)_5, (P_{C_4}x)_6, \ldots \big),$$
(31b)
$$P_{C_{[2]}} x = \big(x_1, (P_{C_2}x)_2, (P_{C_2}x)_3, (P_{C_2}x)_4, (P_{C_5}x)_5, (P_{C_5}x)_6, (P_{C_5}x)_7, \ldots \big),$$
(31c)
$$P_{C_{[3]}} x = \big(x_1, x_2, (P_{C_3}x)_3, (P_{C_3}x)_4, (P_{C_3}x)_5, (P_{C_6}x)_6, (P_{C_6}x)_7, (P_{C_6}x)_8, \ldots \big),$$

respectively, where each P_{C_i} is given by (26).

We now record the intrepid counterpart.

Proposition 2.5 (intrepid curvature constraint projectors). *The intrepid counterpart of* (26) *is*

(32)
$$x \mapsto \begin{cases} x - \dfrac{\langle u_i, x\rangle - \tau_i\tau_{i+1}(\delta_i + \gamma_i)/2}{\|u_i\|^2} u_i, & \text{if } \langle u_i, x\rangle < \tau_i\tau_{i+1}(3\delta_i - \gamma_i)/2; \\[2ex] x - 2\dfrac{\langle u_i, x\rangle - \delta_i\tau_i\tau_{i+1}}{\|u_i\|^2} u_i, & \text{if } \tau_i\tau_{i+1}(3\delta_i - \gamma_i)/2 \leq \langle u_i, x\rangle < \delta_i\tau_i\tau_{i+1}; \\[2ex] x, & \text{if } \delta_i\tau_i\tau_{i+1} \leq \langle u_i, x\rangle \leq \gamma_i\tau_i\tau_{i+1}; \\[2ex] x - 2\dfrac{\langle u_i, x\rangle - \gamma_i\tau_i\tau_{i+1}}{\|u_i\|^2} u_i, & \text{if } \gamma_i\tau_i\tau_{i+1} < \langle u_i, x\rangle \leq \tau_i\tau_{i+1}(3\gamma_i - \delta_i)/2; \\[2ex] x - \dfrac{\langle u_i, x\rangle - \tau_i\tau_{i+1}(\delta_i + \gamma_i)/2}{\|u_i\|^2} u_i, & \text{if } \tau_i\tau_{i+1}(3\gamma_i - \delta_i)/2 < \langle u_i, x\rangle. \end{cases}$$

These counterparts induce the intrepid variant of (31).

2.5. The convex feasibility problem. The *convex feasibility problem* motivated by road design is to find a point in

$$C := Y \cap S_{\text{even}} \cap S_{\text{odd}} \cap C_{[1]} \cap C_{[2]} \cap C_{[3]}, \tag{33}$$

where the sets on the right side are defined in (11), (19), and (29). Note that we have explicit formulas available for all six projectors (see Propositions 2.1, 2.2, and 2.4). Moreover, we have intrepid variants of the projectors onto the five[2] constraint sets S_{even}, S_{odd}, $C_{[1]}$, $C_{[2]}$, and $C_{[3]}$ (see Propositions 2.3 and 2.5).

3. Feasibility problems and projection methods

In this section, we present a selection of classic projection methods for solving feasibility problems. To this end, let $m \in \{2, 3, \ldots\}$ and $C_1, \ldots, C_m$ be nonempty closed convex subsets of X.

3.1. The m-set feasibility problem and its reduction to two sets. Our aim is to

$$\text{find } x \in C := C_1 \cap \cdots \cap C_m \neq \varnothing, \tag{34}$$

or, even more ambitiously, to compute

$$P_C v \tag{35}$$

for some given point $v \in X$, or some intermediate between feasibility and best approximation. Of course, we have the concrete scenario (33) of Section 2.5, where $m = 6$, in mind; however, the discussion in this section is not limited to that particular instance.

Projection methods solve (34) by generating a sequence of vectors by using the projectors $P_{C_1}, \ldots, P_{C_m}$ onto the individual sets. (For further background material on projection methods, see, e.g., [**6**], [**19**], and [**26**].) As is illustrated by Section 2, these projectors are available for a variety of constraints appearing in practical applications. Before we proceed to catalogue the algorithms, we note that some algorithms (see, e.g., Algorithm 3.12 below) work intrinsically only with *two* constraint sets. Because m in our application is not large, this turns out to be not a handicap at all as we can reformulate (34) in a product space in the following fashion: In the *Hilbert product space*

$$\mathbf{X} := X^m, \tag{36}$$

equipped with

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{m} \langle x_i, y_i \rangle \quad \text{and} \quad \|\mathbf{x}\| = \sqrt{\sum_{i=1}^{m} \|x_i\|^2}, \tag{37}$$

where $\mathbf{x} = (x_1, \ldots, x_m)$ and $\mathbf{y} = (y_1, \ldots, y_m)$ belong to $\mathbf{X}$, we consider the *Cartesian product* of the constraints together with the *diagonal* in $\mathbf{X}$, i.e.,

$$\mathbf{C} := C_1 \times \cdots \times C_m \quad \text{and} \quad \mathbf{D} := \big\{(x, \ldots, x) \in \mathbf{X} \mid x \in X\big\}. \tag{38}$$

Then for every $x \in X$, we have the key equivalence

$$x \in C \quad \Leftrightarrow \quad (x, \ldots, x) \in \mathbf{C} \cap \mathbf{D}; \tag{39}$$

[2] There is no fundamentally different intrepid variant of P_Y because the interior of the affine subspace Y is empty.

thus, the original m-set feasibility problem (34) in X reduces to *two-set* feasibility problem for the sets $\mathbf{C}$ and $\mathbf{D}$ in $\mathbf{X}$. This approach is viable because the projectors onto $\mathbf{C}$ and $\mathbf{D}$ are given explicitly (see, e.g., [**6**, Propositions 28.3 and 28.13]) by

$$P_{\mathbf{C}}\colon (x_1,\ldots,x_m)\mapsto (P_{C_1}x_1,\ldots,P_{C_m}x_m) \tag{40a}$$

and by

$$P_{\mathbf{D}}\colon (x_1,\ldots,x_m)\mapsto (y,\ldots,y),\quad \text{where } y=\tfrac{1}{m}(x_1+\cdots+x_m), \tag{40b}$$

respectively.

3.2. Projection methods and . . . Swiss Army knives! Projection methods use the projectors onto the given constraint sets in some fashion. Because the squared distance functions, $d^2(\cdot,C_i)$, are Fréchet differentiable with derivative $2(\operatorname{Id}-P_{C_i})$ (see, e.g., [**6**, Corollary 12.30]) projection methods are *first-order methods* — these are known to allow at best for linear convergence. Not surprisingly, they are not always competitive with special-purpose solvers [**45**]; however, when projection methods succeed (see [**20**] for a compelling set of examples), then they have a variety of *very attractive features*:

- Projection methods are *easy to understand.* This is important in industry, where mathematical/algorithmic considerations are only one part of an engineer's job. The engineer will not typically be familiar with the latest research developments in all branches of relevant mathematics. On the other hand, the idea of a projection method is often immediately grasped by drawing some sketches.
- Projection methods are *easy to implement.* In Section 2, we have seen various formulas for projection methods. These involve simple operations from linear algebra and are easily programmed.
- Projection methods are *easy to maintain.* Once implemented, the code for these methods is typically small and in fact smaller than other pieces of code dealing with data input/output. This makes maintenance straightforward.
- Projection methods are *easy to deploy.* Because of typically small memory requirements, it makes them much easier to deploy on low memory computers like mobile devices. Also, the code base required for projection methods is in most cases significantly smaller than the size of libraries for linear or nonlinear optimization solver software. Thus, projection methods satisfy some of the key requirements for *embedded optimization* [**17**], where the solution of one method is used within an encompassing algorithm.
- Projection methods are *inexpensive to implement.* Because of typically straight forward implementations, there is no need for commercial optimization solver software.
- Projection methods can be *very fast.* If the iterations in a projection method can be executed quickly, then for certain classes of problems projection methods can become very competitive with traditional optimization algorithms. In Section 6.6 below, we illustrate the potential of projection methods when compared to algorithms for linear programming or even mixed-integer linear programming.

In summary, *projection methods possess the same essential characteristics of Swiss Army knives*: they are flexible, lightweight, simple and very convenient provided they get the job done. If the saw included with the swiss army knife in your pocket cuts the branch of the tree, then there is no need for you to either get a big saw out of the garage or to buy a chainsaw from the hardware store! The road design feasibility problem analyzed in this paper adds a new compelling success story.

3.3. A catalogue of projection methods. In this subsection, we provide a list of projection methods. Each of these methods produces a sequence that converges — sometimes after applying a suitable operator — to a point in C provided that $C \neq \varnothing$ (and perhaps an additional assumption is satisfied). Basic references are the books [**6**], [**19**], [**26**], [**35**], [**43**], and [**44**]; if these books are insufficient, we include further pointers in accompanying remarks.

Note that numerous generalizations of projection methods are known. These typically involve additional parameters (e.g., weights and relaxation parameters). Because we shall compare these methods numerically, we have to restrict our attention to the most basic instance of each method.

In the following, we restrict our attention to the most basic instance of each method.

While these methods are generally not able to find $P_C v$, they may lead to feasible solutions that are "fairly close" to $P_C v$ provided the starting point is chosen to be v.

We start with the method of cyclic projections, which has a long history (see, e.g., [**35**]).

ALGORITHM 3.1 (cyclic projections (CycP)). *Set $x_0 = v$ and update*

$$(\forall k \in \mathbb{N}) \quad x_{k+1} := T x_k, \quad \textit{where} \quad T := P_{C_m} P_{C_{m-1}} \cdots P_{C_1}. \tag{41}$$

A modern variant of CycP replaces the projectors in (41) by intrepid counterparts when available.

ALGORITHM 3.2 (cyclic intrepid projections (CycIP)). *Set $x_0 = v$ and update*

$$(\forall k \in \mathbb{N}) \quad x_{k+1} := T x_k, \quad \textit{where} \quad T := R_m R_{m-1} \cdots R_1, \tag{42}$$

where each R_i is either an intrepid counterpart of P_{C_i} (if available), or P_{C_i} itself (if there is no intrepid counterpart).

REMARK 3.3 (CycIP). CycIP (also known as ART3)[3] is available for (33) because we have the intrepid counterparts of the projectors for the slope and curvature constraints at our disposal (see Propositions 2.3 and 2.5); for the interpolation constraint, we utilize the original projector (see also Footnote 2). This method is fairly

[3] In an earlier version [**10**] of this paper, we referred to CycIP as CycP+. However, as pointed out by Gabor Herman, ART3+ is a modification of ART3 where the *order* of the constraints considered is adaptively changed to speed up convergence. This is particularly important when the number of constraints, m, is huge. In our road design application we have only six constraints; thus, for simplicity, we implemented ART3 rather than ART3+ because the numerical results carried out with noncyclic orders and randomized versions along the lines of Strohmer and Vershynin [**62**] were comparable. To avoid confusion with ART3+, we therefore changed the name of Algorithm 3.2 from CycP+ to CycIP.

recent[4]; see [**20**], [**21**], [**28**] and [**51**] (for constructions based on halfspaces and subgradient projectors as well as accelerations), and also [**11**] (where one of the sets is an *obtuse cone* K so that the intrepid projector is actually the reflector $2P_K - \mathrm{Id}$).

While Algorithm 3.1 is *sequential*, the following method is *parallel*:

ALGORITHM 3.4 (parallel projections (ParP)). *Set $x_0 = v$ and update*

$$(\forall k \in \mathbb{N}) \quad x_{k+1} := Tx_k, \quad \textit{where} \quad T := \tfrac{1}{m}\big(P_{C_1} + \cdots + P_{C_m}\big). \tag{43}$$

REMARK 3.5 (ParP). In view of (40), it is interesting to note that ParP is equivalent to iterating $P_{\mathbf{D}}P_{\mathbf{C}}$, i.e., to applying CycP to the subsets $\mathbf{C}$ and $\mathbf{D}$ of $\mathbf{X}$. See also [**60**, Corollary 2.6] and [**2**, Section 6].

The next method can be seen as a hybrid of CycP and ParP.

ALGORITHM 3.6 (string-averaging projections (SaP)). *Set $x_0 = v$ and update*

$$(\forall k \in \mathbb{N}) \quad x_{k+1} := Tx_k, \quad \textit{where} \quad T := \tfrac{1}{m}\big(P_{C_1} + P_{C_2}P_{C_1} + \cdots + P_{C_m}\cdots P_{C_2}P_{C_1}\big). \tag{44}$$

REMARK 3.7 (SaP). For further information, see [**24**] and [**25**] (as well as [**3**, Example 2.14] and [**14**, Section 8]).

ALGORITHM 3.8 (extrapolated parallel projections (ExParP)). *Set $x_0 = v$ and update, for every $k \in \mathbb{N}$,*

$$x_{k+1} := Tx_k, \ \textit{where} \ Tx := \begin{cases} x + \dfrac{\sum_{i=1}^m \|x - P_{C_i}x\|^2}{\|\sum_{i=1}^m (x - P_{C_i}x)\|^2} \displaystyle\sum_{i=1}^m (P_{C_i}x - x), & \textit{if } x \notin C; \\ x, & \textit{otherwise.} \end{cases} \tag{45}$$

REMARK 3.9 (ExParP). This method is actually an instance of the *subgradient projection method* applied to the function $x \mapsto \sum_{i=1}^m d^2(x, C_i)$; see [**30**] for further information.

ALGORITHM 3.10 (extrapolated alternating projections (ExAltP)). *Assume that C_1 is an affine subspace — this is the case in our application when we choose the interpolation constraint. Set $x_0 = v$, and let $k \in \mathbb{N}$. Let I_k be a nonempty subset of $\{1, \ldots, m\}$ containing exactly m_k indices such that for each $j \in \{2, \ldots, m\}$, j belongs to I_k frequently. Given x_k, set*

$$z_k = P_{C_1}x_k, \ \ p_k = P_{C_1}\bigg(\frac{1}{m_k}\sum_{i \in I_k} P_{C_i}z_k\bigg), \tag{46a}$$

$$\mu_k = \begin{cases} \dfrac{\sum_{i \in I_k} \|z_k - P_{C_i}z_k\|^2}{m_k\|p_k - z_k\|^2}, & \textit{if } z_k \notin \displaystyle\bigcap_{i \in I_k} C_i; \\ 1, & \textit{otherwise,} \end{cases} \tag{46b}$$

and then update

$$x_{k+1} = z_k + \mu_k(p_k - z_k). \tag{46c}$$

REMARK 3.11 (ExAltP). See [**7**, Algorithm 3.5] for further information on this method. In our implementation, we chose $I_k = \{2, \ldots, m\}$ so that $m_k = m - 1$.

---

[4] However, the idea behind CycIP goes back to a 1975 paper by Herman [**49**].

The next method is different from the previous ones in two aspects: First, it truly operates in $\mathbf{X}$ and thus has increased storage requirements — fortunately, m is small in our application so this is of no concern when dealing with (33). Second, the sequence of interest is actually different and derived from another sequence that governs the iteration.

ALGORITHM 3.12 (Douglas–Rachford (D–R)). *Set* $\mathbf{x}_0 = (v, \dots, v) \in \mathbf{X} = X^m$. *Given* $k \in \mathbb{N}$ *and* $\mathbf{x}_k = (x_{k,1}, \dots, x_{k,m}) \in \mathbf{X}$, *update to* $\mathbf{x}_{k+1} = (x_{k+1,1}, \dots, x_{k+1,m})$, *where*

$$\bar{x}_k = \frac{1}{m}\sum_{i=1}^{m} x_{k,i} \tag{47a}$$

and

$$(\forall i \in \{1, \dots, m\}) \;\; x_{k+1,i} = x_{k,i} - \bar{x}_k + P_{C_i}(2\bar{x}_k - x_{k,i}). \tag{47b}$$

The sequence of interest is not $(\mathbf{x}_k)_{k\in\mathbb{N}}$ *but rather* $(\bar{x}_k)_{k\in\mathbb{N}}$.

REMARK 3.13 (general Douglas–Rachford algorithm). Let us briefly sketch how the update formula (47) is derived from the general splitting version of D–R, which aims to minimize the sum $f + g$ of proper lower semicontinuous convex functions $f\colon X \to \left]-\infty, +\infty\right]$ and $g\colon X \to \left]-\infty, +\infty\right]$. Given $x_0 \in X$, the algorithm proceeds via

$$(\forall k \in \mathbb{N}) \quad x_{k+1} = Tx_k, \;\; \text{where} \;\; T = \operatorname{Prox}_g(2\operatorname{Prox}_f - \mathrm{Id}) + \mathrm{Id} - \operatorname{Prox}_f, \tag{48}$$

and where Prox_f denotes the *proximal mapping*[5] (or *proximity operator*) of f, i.e., $\operatorname{Prox}_f(y)$ is the unique minimizer of the function $x \mapsto \frac{1}{2}\|x-y\|^2 + f(x)$; the sequence to monitor is $(\operatorname{Prox}_f x_k)_{k\in\mathbb{N}}$. Now assume that A and B are two nonempty closed convex subsets of X. The *indicator function* ι_A of A takes the value 0 on A, and $+\infty$ outside A, and analogously for ι_B. We set $f = \iota_B$ and $g = \iota_A$. It is then clear that

$$\operatorname{Prox}_f = P_B \quad \text{and} \quad \operatorname{Prox}_g = P_A \tag{49}$$

and that (48) turns into

$$(\forall k \in \mathbb{N}) \quad x_{k+1} = Tx_k, \;\; \text{where} \;\; T = P_A(2P_B - \mathrm{Id}) + \mathrm{Id} - P_B \tag{50}$$

Applying this in $\mathbf{X}$ (with $A = \mathbf{C}$ and $B = \mathbf{D}$) and recalling (40), we obtain (47). Viewed directly in $\mathbf{X}$, we obtain the iteration

$$\mathbf{x}_{k+1} = \mathbf{T}\mathbf{x}_k, \;\; \text{where} \;\; \mathbf{T} := P_{\mathbf{C}}(2P_{\mathbf{D}} - \mathbf{Id}) + \mathbf{Id} - P_{\mathbf{D}} = \frac{\mathbf{Id} + (2P_{\mathbf{C}} - \mathbf{Id})(2P_{\mathbf{D}} - \mathbf{Id})}{2}, \tag{51a}$$

and we monitor the sequence

$$(P_{\mathbf{D}}\mathbf{x}_k)_{k\in\mathbb{N}}. \tag{51b}$$

For convergence results, see [**6**, Chapter 26], [**37**], and [**56**].

REMARK 3.14 (D–R vs alternating direction method of multipliers (ADMM)). ADMM is a very popular method [**16**] that can also be adapted to solve feasibility problems for two sets. Suppose we wish to find a point in $A \cap B$, where A and B

[5] We note in passing that Prox_f is also identical to the resolvent of the subdifferential operator ∂f. For more on the asymptotic behavior of the composition of two resolvents, see [**9**].

are two nonempty closed convex subsets of X. Given $u_0 \in X$ and $b_0 \in X$, ADMM generates three sequences $(a_k)_{k\geq 1}$, $(b_k)_{k\in\mathbb{N}}$, and $(u_k)_{k\in\mathbb{N}}$ via

$$(\forall k \in \mathbb{N}) \quad a_{k+1} := P_A(b_k - u_k),\ b_{k+1} := P_B(a_{k+1} + u_k),\ u_{k+1} := u_k + a_{k+1} - b_{k+1}. \tag{52}$$

On the other hand, D–R for this problem, with starting point $x_0 \in X$, produces two sequences $(x_k)_{k\in\mathbb{N}}$ and $(y_k)_{k\in\mathbb{N}}$ via

$$(\forall k \in \mathbb{N}) \quad y_k := P_B x_k,\ x_{k+1} := P_A(2y_k - x_k) + x_k - y_k. \tag{53}$$

Now assume that

$$x_0 = b_0 \in B \text{ and } u_0 = 0. \tag{54}$$

Then $y_0 = P_B x_0 = x_0$, $x_1 = P_A(2y_0 - x_0) + x_0 - y_0 = P_A x_0 = P_A(b_0 - 0) = P_A(b_0 - u_0) = a_1 = a_1 + u_0$ and $y_1 = P_B x_1 = P_B(a_1 + u_0) = b_1$. Furthermore, assume that for some $k \geq 1$, we have $x_k = a_k + u_{k-1}$ and $y_k = b_k$. Then $2y_k - x_k = 2b_k - (u_{k-1} + a_k) = b_k - a_k + b_k - u_{k-1} = u_{k-1} - u_k + b_k - u_{k-1} = b_k - u_k$ and $x_k - y_k = u_{k-1} + a_k - b_k = u_k$; in turn, this implies $x_{k+1} = P_A(2y_k - x_k) + x_k - y_k = P_A(b_k - u_k) + u_k = a_{k+1} + u_k$ and $y_{k+1} = P_B x_{k+1} = P_B(a_{k+1} + u_k) = b_{k+1}$. It follows inductively that[6]

$$(x_k)_{k\geq 1} = (a_k + u_{k-1})_{k\geq 1} \ \text{ and } \ (y_k)_{k\geq 1} = (b_k)_{k\geq 1}. \tag{55}$$

In this sense,

$$\text{D–R and ADMM } \textit{are equivalent.} \tag{56}$$

See also [**16, 32, 39, 41**] for further information on ADMM and related methods. Furthermore, if A and B are linear subspaces, then $(x_k)_{k\in\mathbb{N}}$, the sequence governing DR, has the update rule

$$x_{k+1} = (P_A P_B + P_{A^\perp} P_{B^\perp}) x_k \tag{57}$$

since $P_A(2P_B - \mathrm{Id}) + \mathrm{Id} - P_B = P_A(2P_B - P_B - P_{B^\perp}) + P_{B^\perp} = P_A(P_B - P_{B^\perp}) + P_A P_{B^\perp} + P_{A^\perp} P_{B^\perp} = P_A P_B + P_{A^\perp} P_{B^\perp}$. Using, e.g., [**8**, Corollary 3.9], one can further show that $\mathrm{Fix}(P_A P_B + P_{A^\perp} P_{B^\perp}) = (A \cap B) + (A^\perp \cap B^\perp)$.

3.4. Summary of feasibility algorithms.

Name	Acronym	Formula	Monitor
Cyclic Projections	CycP	(41)	$(x_k)_{k\in\mathbb{N}}$
Cyclic Intrepid Projections	CycIP	(42)	$(x_k)_{k\in\mathbb{N}}$
Parallel Projections	ParP	(43)	$(x_k)_{k\in\mathbb{N}}$
String-averaging Projections	SaP	(44)	$(x_k)_{k\in\mathbb{N}}$
Extrapolated Parallel Projections	ExParP	(45)	$(x_k)_{k\in\mathbb{N}}$
Extrapolated Alternating Projections	ExAltP	(46)	$(x_k)_{k\in\mathbb{N}}$
Douglas–Rachford	D–R	(47)	$(\bar{x}_k)_{k\in\mathbb{N}}$

Note that all algorithms in this table proceed by iterating an operator, and monitoring either the iterates directly, or some simple version thereof. This is a key point when we revisit these algorithms in the next section.

[6] In fact, this argument works much more generally when the projectors P_A and P_B are replaced by arbitrary *resolvents* and b_0 is a fixed point of the second resolvent.

3.5. Superiorization: between feasibility and best approximation. The algorithms considered so far are designed to solve the *feasibility problem* (34). Let $v \in X \smallsetminus C$. We shall discuss algorithms for finding $P_C v$ in Section 4 below. This *best approximation problem* is equivalent to[7] solving the optimization problem

$$\min_{x \in C} \tfrac{1}{2}\|x - v\|^2. \tag{58}$$

The new paradigm of *superiorization* (see, e.g., [**22, 52**], and [**33, 40, 54, 57, 59**] for applications) lies between feasibility and this best approximation problem. It is not quite trying to solve (58); rather, the objective is to find a feasible point that is a superior to one returned by a feasibility algorithm. To explain this in detail, we assume that $T\colon X \to X$ satisfies

$$\operatorname{Fix} T = C; \tag{59}$$

hence, (58) is equivalent to

$$\min_{x \in \operatorname{Fix} T} \tfrac{1}{2}\|x - v\|^2. \tag{60}$$

Applying the superiorization approach to (60), we obtain the following abstract algorithm:

(61)

```
Algorithm: Superiorization of T
Data: v ∈ X, ε > 0
Result: x_k
k ← 0
x_0 ← v
θ ← 1
while d(x_k) > ε do
    if ‖x_k − v‖ > 0 then
        x̃ ← x_k + θ(x_k − v)/‖x_k − v‖
    else
        x̃ ← x_k
    end
    θ ← θ/2
    if ‖x̃ − v‖ ≤ ‖x_k − v‖ and d(Tx̃) < d(x_k)
    then
        x_{k+1} ← Tx̃
    end
    k ← k + 1
end
```

Note that $d\colon X \to \mathbb{R}_+$ is a performance function satisfying $d(x) = 0$ if and only if $x \in C$. (In Section 6, we use (112).) With the exception of D–R, each algorithm in Section 3.4 has a *superiorized* counterpart. (It is not clear how D–R should be superiorized because the fixed point set of the operator T governing the iteration is generally different from C, the set of interest.) We denote the superiorized

[7] We work here with a squared version of (8) because the objective function is then differentiable.

counterpart of CycP by sCycP, and analogously for the other algorithms. For remarks on the numerical performance of these algorithms, see Section 6.4.

4. Best approximation algorithms

4.1. The problem and notation. We continue to assume that $m \in \{2,3,\ldots\}$ and that $C_1,\ldots,C_m$ are closed convex subsets of X such that

$$C := C_1 \cap \cdots \cap C_m \neq \varnothing. \tag{62}$$

Let $v \in X$. We wish to determine

$$P_C v. \tag{63}$$

Before we present a selection of pertinent best approximation algorithms, let us fix some notation for ease of use. It is notationally convenient to occasionally work with cyclic remainders

$$[1]_m = 1, [2]_m = 2, \ldots, [m]_m = m, [m+1]_m = 1, \ldots, \tag{64}$$

taken from $\{1,\ldots,m\}$. We also require the operator Q defined by [8]

$$\begin{aligned} &Q\colon X \times X \times X \to X \\ &(x,y,z) \mapsto \begin{cases} z, & \text{if } \rho = 0 \text{ and } \chi \geq 0; \\ x + \Big(1 + \frac{\chi}{\nu}\Big)(z-y), & \text{if } \rho > 0 \text{ and } \chi\nu \geq \rho; \\ y + \frac{\nu}{\rho}\Big(\chi(x-y) + \mu(z-y)\Big), & \text{if } \rho > 0 \text{ and } \chi\nu < \rho, \end{cases} \end{aligned} \tag{65}$$

where $\chi := \langle x-y, y-z\rangle$, $\mu := \|x-y\|^2$, $\nu := \|y-z\|^2$, and $\rho := \mu\nu - \chi^2$. An analogous formula holds for $\mathbf{Q}\colon \mathbf{X} \times \mathbf{X} \times \mathbf{X} \to \mathbf{X}$.

4.2. A catalogue of best approximation methods. As in Section 3, we present a list of best approximation methods based on projectors and comments. Unless stated otherwise, each of these methods produces a main/governing sequence that is converging to $P_C v$.

Algorithm 4.1 (Halpern–Wittmann (H–W)). *Set $x_0 = v$ and update*

$$(\forall k \in \mathbb{N}) \quad x_{k+1} := \tfrac{1}{k+1} v + \tfrac{k}{k+1} P_{C_m} P_{C_{m-1}} \cdots P_{C_1} x_k. \tag{66}$$

Remark 4.2 (H–W). This algorithm was introduced by Halpern [**47**] while Wittmann [**64**] proved convergence for the choice of parameters presented in Algorithm 4.1. Many variants have been proposed and studied.

Algorithm 4.3 (Cyclic Dykstra algorithm (CycDyk)). *Set $x_0 := v$, $q_{-(m-1)} := q_{-(m-2)} := \cdots := q_{-1} := q_0 := 0$, and update*

$$(\forall k \in \mathbb{N}) \quad x_{k+1} := P_{C_{[k+1]_m}}(x_k + q_{k+1-m}) \ \text{ and } \ q_{k+1} := x_k + q_{k+1-m} - x_{k+1}. \tag{67}$$

Remark 4.4 (CycDyk). For convergence proofs, see, e.g., [**18**] or [**6**, Theorem 29.2]. See also [**31**, Section 3] for connections to the *forward–backward method.*

The following method operates in $\mathbf{X}$.

[8] If $\rho = 0$ and $\chi < 0$, then the output of Q is undefined — this corresponds to the case when $C = \varnothing$.

ALGORITHM 4.5 (Parallel Dykstra algorithm (ParDyk)). *Set* $(\mathbf{y}_0, \mathbf{z}_0) = (v, \ldots, v, 0, \ldots, 0) \in \mathbf{X} \times \mathbf{X}$*, and let* $(\mathbf{y}_k, \mathbf{z}_k) = (y_{k,1}, \ldots, y_{k,m}, z_{k,1}, \ldots, z_{k,m}) \in \mathbf{X} \times \mathbf{X}$ *be given. Then the next iterate is* $(\mathbf{y}_{k+1}, \mathbf{z}_{k+1}) = (y_{k+1,1}, \ldots, y_{k+1,m}, z_{k+1,1}, \ldots, z_{k+1,m})$*, where the sequence* $(\bar{x}_k)_{k\in\mathbb{N}}$ *to monitor is*

$$\bar{x}_k = \frac{1}{m}\sum_{i=1}^{m} y_{k,i} \tag{68a}$$

and the update formulas are

$$(\forall i \in \{1, \ldots, m\}) \quad y_{k+1,i} := P_{C_i}(z_{k,i} + \bar{x}_k) \ \ \text{and} \ \ z_{k+1,i} := z_{k,i} + \bar{x}_k - y_{k+1,i}. \tag{68b}$$

REMARK 4.6 (ParDyk). ParDyk is CycDyk applied to the subsets $\mathbf{C}$ and $\mathbf{D}$ of $\mathbf{X}$; see, e.g., [**2**, Theorem 6.1].

REMARK 4.7 (Dykstra vs ADMM). Let A and B be two nonempty closed convex subsets of X, and let $v \in X$. CycDyk for finding $P_{A\cap B}w$ generates sequences $(a_k)_{k\geq 1}$, $(b_k)_{k\in\mathbb{N}}$, $(p_k)_{k\in\mathbb{N}}$, and $(q_k)_{k\in\mathbb{N}}$ as follows[9]: Set $b_0 := v$, $p_0 := q_0 := 0$, and for every $k \in \mathbb{N}$, update

$$a_{k+1} := P_A(b_k + p_k), \qquad p_{k+1} := b_k + p_k - a_{k+1}, \tag{69a}$$

$$b_{k+1} := P_B(a_{k+1} + q_k), \qquad q_{k+1} := a_{k+1} + q_k - b_{k+1}. \tag{69b}$$

On the other hand, given $u_0 \in X$ and $y_0 \in X$, ADMM for finding a point in $A\cap B$ — not necessarily $P_{A\cap B}w$ — generates three sequences $(x_k)_{k\geq 1}$, $(y_k)_{k\in\mathbb{N}}$, and $(u_k)_{k\in\mathbb{N}}$ via

$$(\forall k \in \mathbb{N}) \quad x_{k+1} := P_A(y_k - u_k),\ y_{k+1} := P_B(x_{k+1} + u_k),\ u_{k+1} := u_k + x_{k+1} - y_{k+1}. \tag{70}$$

Now let us assume that

$$B \text{ is a linear subspace} \tag{71}$$

and that $u_0 \in B^\perp$. Then it is well known that the Dykstra update (69) simplifies to

$$a_{k+1} := P_A(b_k + p_k),\ \ b_{k+1} := P_B(a_{k+1}),\ \ p_{k+1} := p_k + b_k - a_{k+1}, \tag{72}$$

because the sequence $(q_k)_{k\in\mathbb{N}}$ lies in $B^\perp$ and thus becomes "invisible" when computing $(b_k)_{k\in\mathbb{N}}$ due to the linearity of P_B. Turning to ADMM, we observe that $(u_k)_{k\in\mathbb{N}}$ lies in $B^\perp$ which simplifies the update for y_{k+1} to $y_{k+1} := P_B(x_{k+1})$. Setting $(v_k)_{k\in\mathbb{N}} := -(u_k)_{k\in\mathbb{N}}$, we see that (70) turns into

$$(\forall k \in \mathbb{N}) \quad x_{k+1} := P_A(y_k + v_k),\ y_{k+1} := P_B(x_{k+1}),\ v_{k+1} := v_k + y_{k+1} - x_{k+1}. \tag{73}$$

Comparing (72) to (73), we see that the update formulas look *almost* identical: The only difference lies in the update formulas of the auxiliary sequences $(p_k)_{k\in\mathbb{N}}$ and $(v_k)_{k\in\mathbb{N}}$ — the former works with b_k while the latter incorporates immediately the more recent update y_{k+1}. However, the resulting sequences and hence algorithms appear to be *different*[10]. Let us now further specialize by additionally assuming

[9] Here the sequence $(q_k)_{k\in\mathbb{N}}$ from (67) is split into subsequences corresponding to odd and even terms for easier readability.

[10] It is stated on [**16**, page 34f] that (in our notation) "(73) is exactly Dykstra's alternating projections method ... which is far more efficient than the classical method [of alternating projections] that does not use the dual variable v.". This statement appears to be at least ambiguous because the crucial starting points are not specified. See also [**58**].

that

$$A \text{ is a linear subspace.} \tag{74}$$

Then the Dykstra update (72) does not require the sequence $(p_k)_{k\in\mathbb{N}}$ anymore (because it lies in $A^\perp$ and it thus plays no role in the generation of $(a_k)_{k\geq 1}$), and it further simplifies to

$$a_{k+1} := P_A(b_k), \ \ b_{k+1} := P_B(a_{k+1}). \tag{75}$$

Hence, as is well known, Dykstra turns into CycP, which is also known as *von Neumann's alternating projection method* in this special case. On the other hand, these additional assumptions do not seem to simplify (73). In fact, the behaviour of Dykstra can starkly differ from ADMM even in this setting: Suppose that $X=\mathbb{R}^2$, that $A=\mathbb{R}\cdot(1,1)$ and $B=\mathbb{R}\times\{0\}$. Then the sequence $(b_k)_{k\in\mathbb{N}}$ with starting point $b_0 := v := (1,0)$ turns out to be $(2^{-k},0)_{k\in\mathbb{N}}$. In contrast (see also Footnote 10), we compute the following ADMM updates, where $y_0 := v := (1,0)$ and $v_0 := (0,0)$: $x_1 = P_A(y_0+v_0) = P_A(1,0) = \frac{1}{2}(1,1)$, $y_1 = P_B(x_1) = \frac{1}{2}(1,0)$, $v_1 = v_0 + y_1 - x_1 = \frac{1}{2}(0,-1)$, $x_2 = P_A(y_1+v_1) = P_A(\frac{1}{2}(1,-1)) = (0,0)$, and $y_2 = P_B(x_2) = (0,0)$. Since $x_2 = y_2 \in A\cap B$, the algorithm terminates whenever feasibility is the implemented stopping criterion.

Algorithm 4.8 (Haugazeau's algorithm with cyclic projections (hCycP)). *Set $x_0 = v$, and update*[11]

$$(\forall k\in\mathbb{N}) \quad x_{k+1} := Q(x_0, x_k, P_{C_{[k+1]_m}} x_k). \tag{76}$$

Remark 4.9 (hCycP). For convergence proofs, see [**48**] or [**6**, Corollary 29.8]. The general pattern for methods that undergo a Haugazeau-type modification is (see [**5**] for details)

$$(\forall k\in\mathbb{N}) \quad x_{k+1} := Q(x_0, x_k, T_{\mathrm{i}(k)} x_k), \tag{77}$$

where $(T_i)_{i\in I}$ is a family of operators and $\mathrm{i}\colon \mathbb{N}\to I$ selects which operator is drawn at iteration k. This gives rise to many variants; in the following, we shall focus on a representative selection.

Here is a Haugazeau-type modification of ParP (Algorithm 3.4):

Algorithm 4.10 (hParP). *Set $x_0 := v$ and update*

$$(\forall k\in\mathbb{N}) \quad x_{k+1} := Q(x_0, x_k, Tx_k), \quad \text{where } T = \tfrac{1}{m}\big(P_{C_1} + \cdots + P_{C_m}\big). \tag{78}$$

The following Haugazeau-type modification of D–R operates in the product space $\mathbf{X}$.

Algorithm 4.11 (hD–R). *Let $\mathbf{T}\colon \mathbf{X}\to\mathbf{X}$ be the operator governing D–R (see* (51a)*), and set $\mathbf{x}_0 = (v,\ldots,v)\in\mathbf{X}$. Given $k\in\mathbb{N}$ and $\mathbf{x}_k = (x_{k,1},\ldots,x_{k,m})\in\mathbf{X}$, update the governing sequence by*

$$\mathbf{x}_{k+1} := \mathbf{Q}(\mathbf{x}_0, \mathbf{x}_k, \mathbf{T}\mathbf{x}_k), \tag{79a}$$

and monitor

$$\bar{x}_k := \frac{1}{m}\sum_{i=1}^m x_{k,i}. \tag{79b}$$

[11] Recall (64).

REMARK 4.12 (hD–R). To show that $\bar{x}_k \to P_C(v)$, use [**4**, Example 8.2].

The next algorithm is a variant of D–R tailored for best approximation. It also operates in the product space $\mathbf{X}$.

ALGORITHM 4.13 (baD–R). *Set* $\mathbf{x}_0 = (v, \ldots, v) \in \mathbf{X}$. *Given* $k \in \mathbb{N}$ *and* $\mathbf{x}_k = (x_{k,1}, \ldots, x_{k,m}) \in \mathbf{X}$, *update to* $\mathbf{x}_{k+1} = (x_{k+1,1}, \ldots, x_{k+1,m})$, *where*

$$\bar{x}_k = \frac{1}{m}\sum_{i=1}^{m} x_{k,i} \tag{80a}$$

and

$$(\forall i \in \{1, \ldots, m\}) \;\; x_{k+1,i} = x_{k,i} - \bar{x}_k + P_{C_i}\big((v + 2\bar{x}_k - x_{k,i})/2\big). \tag{80b}$$

4.3. Remarks on baD–R. We comment on various aspects of baD–R and start with its genesis.

REMARK 4.14 (baD–R). Let A and B be nonempty closed convex subsets of X, and let $v \in X$. Revisit Remark 3.13, in which we showed how the feasibility version of D–R, Algorithm 3.12, arises from the general problem of minimizing $f + g$, by choosing $f = \iota_B$ and $g = \iota_A$. To explain Algorithm 4.13, we keep $f = \iota_B$ for which

$$\operatorname{Prox}_f = P_B. \tag{81}$$

However, this time we take $g\colon x \mapsto \frac{1}{2}\|x - v\|^2 + \iota_A(x)$. Then the (unique) minimizer of $f + g$ is indeed $P_{A\cap B}v$. Let x and y be in X. Then $y = \operatorname{Prox}_g x \Leftrightarrow x \in (\operatorname{Id} + \partial g)(y) = 2y - v + N_A(y) \Leftrightarrow \frac{1}{2}(x + v) \in y + \frac{1}{2}N_A(y) = y + N_A(y)$ and so $y = P_A(\frac{1}{2}(x + v))$; that is,

$$\operatorname{Prox}_g\colon x \mapsto P_A\big(\tfrac{1}{2}(x + v)\big). \tag{82}$$

In view of (81) and (82), the general update formula for the Douglas–Rachford splitting method, (48), becomes

$$(\forall k \in \mathbb{N}) \quad x_{k+1} = Tx_k, \;\; \text{where } \; T\colon x \mapsto P_A((2P_Bx - x + v)/2) + x - P_Bx. \tag{83}$$

Applying this in $\mathbf{X}$ (with $A = \mathbf{C}$ and $B = \mathbf{D}$) and recalling (40), we obtain (80).

REMARK 4.15 (baD–R = CycDyk = CycP in the doubly affine case). Let us investigate (83) with starting point $v \in B$ further. First, we rewrite it as

$$x_0 := v \in B, \text{ and } (\forall k \in \mathbb{N})\; y_k := P_Bx_k,\; x_{k+1} := x_k - y_k + P_A(y_k + (v - x_k)/2). \tag{84}$$

We now assume additionally that

$$A \text{ and } B \text{ are linear subspaces of } X. \tag{85}$$

We claim that for every $k \geq 1$,

$$x_k = P_Av - P_BP_Av + P_AP_BP_Av \mp \cdots + P_A(P_BP_A)^{k-1}v, \tag{86}$$

which implies

$$y_k = (P_BP_A)^k v \tag{87}$$

because the differences in (86) all lie in $B^\perp$. Observe that $y_0 = P_Bx_0 = P_Bv = v$ since $v \in B$. It follows that $x_1 = x_0 - y_0 + P_A(y_0 + (v - x_0)/2) = v - v + P_A(v + (v - v)/0) = P_Av$ which shows that (86) holds when $k = 1$. Now assume that (86) holds for some $k \geq 1$. Then (87) also holds. Furthermore,

$$x_k - y_k = P_Av - P_BP_Av \pm \cdots - (P_BP_A)^k v \tag{88}$$

and

$$v - x_k = (\operatorname{Id} - P_A)v + (\operatorname{Id} - P_A)P_B P_A v + \cdots (\operatorname{Id} - P_A)(P_B P_A)^{k-1} v \in A^{\perp}; \tag{89}$$

the latter identity and (87) yield $P_A(y_k + (v - x_k)/2) = P_A y_k = P_A(P_B P_A)^k v$. This verifies (86) with k replaced by $k+1$; therefore, by induction, (86) and hence (87) hold true for all $k \geq 1$. In view of (87) and (75), we observe that baD–R = CycP = CycDyk in sense that the sequences that arise after projection onto B are all identical. Finally, a translation argument reduces the doubly affine case to the doubly linear case.

Remark 4.16 (baD–R $\neq$ CycDyk in general). Suppose that A is a nonempty closed convex subset of X and that B is a subspace of X. Starting both CycDyk and baD–R at v, we obtain for $k \in \mathbb{N}$ the update rules

$$b_0 := v, p_0 := 0, a_{k+1} := P_A(b_k + p_k), p_{k+1} := b_k + p_k - a_{k+1}, b_{k+1} := P_B a_{k+1} \tag{90}$$

and

$$x_0 := v, y_k := P_B x_k, x_{k+1} := x_k - y_k + P_A(y_k + (v - x_k)/2), \tag{91}$$

respectively. One computes that $a_1 = P_A v = x_1$, $p_1 = v - P_A v$, and

$$b_1 = P_B P_A v = y_1. \tag{92}$$

It thus follows that $a_2 = P_A(P_B P_A v + v - P_A v)$ and $x_2 = P_{B^{\perp}} P_A v + P_A(P_B P_A v + (v - P_A v)/2))$. Consequently,

$$b_2 = P_B P_A(P_B P_A v + v - P_A v) \quad \text{and} \quad y_2 = P_B P_A(P_B P_A v + (v - P_A v)/2)). \tag{93}$$

These two vectors certainly appear to be different — let us now exhibit a simple example where they actually are different. We work in the Euclidean plane and thus assume that $X = \mathbb{R}^2$. Set $A := \big\{(\xi, \eta) \mid \xi^2 + (\eta - 1)^2 \leq 1\big\}$, which has the projection formula

$$P_A \colon \mathbb{R}^2 \to \mathbb{R}^2 \colon (\xi, \eta) \mapsto \begin{cases} (\xi, \eta), & \text{if } \xi^2 + (\eta - 1)^2 \leq 1; \\ (0, 1) + \dfrac{(\xi, \eta - 1)}{\sqrt{\xi^2 + (\eta - 1)^2}}, & \text{otherwise,} \end{cases} \tag{94}$$

and set $B := \mathbb{R} \times \{0\}$. Then

$$P_B \colon \mathbb{R}^2 \to \mathbb{R}^2 \colon (\xi, \eta) \mapsto (\xi, 0). \tag{95}$$

Set $v = (1, 0)$. Then (92) turns into $b_1 = y_1 = (\sqrt{2}/2, 0)$ while (93) becomes

$$b_2 = \left(\frac{2}{\sqrt{22 - 8\sqrt{2}}}, 0\right) \approx (0.61180, 0) \neq (0.59718, 0) = \left(\frac{1}{2} \frac{\sqrt{2} + 2}{\sqrt{11 - 2\sqrt{2}}}, 0\right) = y_2. \tag{96}$$

Let us mention in passing that similar examples exist in the product space setting of Section 3.1 when some of the constraint sets are balls.

4.4. Summary of best approximation algorithms.

Name	Acronym	Formula	Monitor
Halpern–Wittmann	H–W	(66)	$(x_k)_{k\in\mathbb{N}}$
Cyclic Dykstra	CycDyk	(67)	$(x_k)_{k\in\mathbb{N}}$
Parallel Dykstra	ParDyk	(68)	$(x_k)_{k\in\mathbb{N}}$
Haugazeau-like Cyclic Projections	hCycP	(76)	$(x_k)_{k\in\mathbb{N}}$
Haugazeau-like Parallel Projections	hParP	(78)	$(x_k)_{k\in\mathbb{N}}$
Haugazeau-like Douglas–Rachford	hD–R	(79)	$(\bar{x}_k)_{k\in\mathbb{N}}$
best approximation with Douglas–Rachford	baD–R	(80)	$(\bar{x}_k)_{k\in\mathbb{N}}$

5. Nonconvexity

In this section, we provide some remarks on the possible absence of convexity.

5.1. Nonconvex slope constraints. We now consider a case with nonconvex constraints. This type of constraints does occur frequently in applications; however, the body of convergence results is sparse and, to the best of our knowledge, all results are *local*, i.e., convergence of the iterates is guaranteed only when the starting point is already sufficiently close to the set of solutions. (See, e.g., [**12, 13**] and the references therein.) We use the convex constraints from Section 2 with one crucial modification: The slope constraints (13) is tightened by additionally imposing a *minimum* absolute value slope, i.e., we assume the existence of two vectors $a = (\alpha_i)$ and $b = (\beta_i)$ in $\mathbb{R}^{n-1}_+$ such that

$$(\forall i \in I) \quad S_i := \big\{x = (x_1, \ldots, x_n) \in \mathbb{R}^n \mid \alpha_i \geq |x_{i+1} - x_i| \geq \beta_i\big\}. \tag{97}$$

Note that if $\beta_i > 0$, then S_i is *not convex*. Analogously to (18), we aggregate these sets to obtain the general nonconvex constraint set

$$S = \bigcap_{i\in\{1,\ldots,n-1\}} S_i. \tag{98}$$

Let $i \in \{1, \ldots, n-1\}$ and $x \in X$. Arguing as in Section 2.3.1, we obtain the counterpart of (16) and see that $P_{S_i}x$ is different from x at most in positions i and $i+1$, and[12]

$$\begin{aligned}
&\big((P_{S_i}x)_i, (P_{S_i}x)_{i+1}\big) \\
&= \begin{cases}
\frac{1}{2}(x_i + x_{i+1} + \alpha_i, x_i + x_{i+1} - \alpha_i), & \text{if } x_{i+1} < x_i - \alpha_i; \\
(x_i, x_{i+1}), & \text{if } x_i - \alpha_i \leq x_{i+1} \leq x_i - \beta_i; \\
\frac{1}{2}(x_i + x_{i+1} + \beta_i, x_i + x_{i+1} - \beta_i), & \text{if } x_i - \beta_i < x_{i+1} \leq x_i; \\
\frac{1}{2}(x_i + x_{i+1} - \beta_i, x_i + x_{i+1} + \beta_i), & \text{if } x_i \leq x_{i+1} < x_i + \beta_i; \\
(x_i, x_{i+1}), & \text{if } x_i + \beta_i \leq x_{i+1} \leq x_i + \alpha_i; \\
\frac{1}{2}(x_i + x_{i+1} - \alpha_i, x_i + x_{i+1} + \alpha_i), & \text{if } x_i + \alpha_i < x_{i+1}.
\end{cases}
\end{aligned} \tag{99}$$

[12] The astute reader will note that when $\beta_i > 0$ and $x_i = x_{i+1}$, then

$$\big((P_{S_i}x)_i, (P_{S_i}x)_{i+1}\big) = \big\{\tfrac{1}{2}(x_i + x_{i+1} + \beta_i, x_i + x_{i+1} - \beta_i), \tfrac{1}{2}(x_i + x_{i+1} - \beta_i, x_i + x_{i+1} + \beta_i)\big\}$$

is not single-valued; indeed, in view of the nonconvexity of S_i and the famous Bunt–Motzkin Theorem (see, e.g., [**6**, Corollary 21.13]), this is to be expected. For the actual implementation, one chooses an arbitrary element from this set.

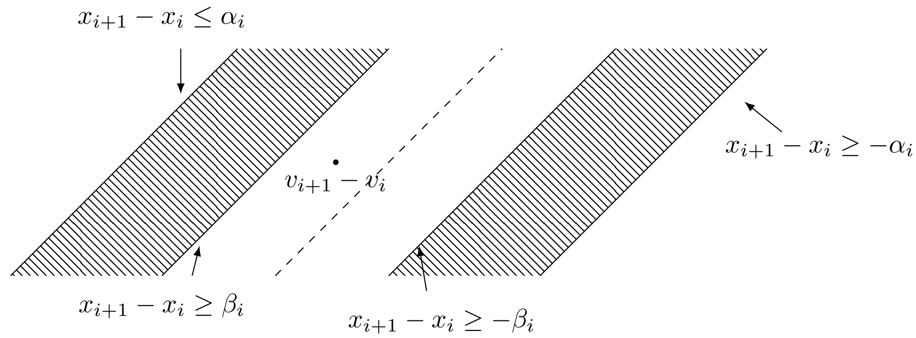

FIGURE 2. General slope constraint in the nonconvex case.

This formula allows us to deal with the general case as in Section 2.3.2; thus, we obtain the counterpart of Proposition 2.2.

Let us also record the *intrepid counterpart* of (99):

(100)

$$\begin{cases} \tfrac{1}{2}\big(x_i + x_{i+1} + \tfrac{1}{2}(\alpha_i+\beta_i), x_i + x_{i+1} - \tfrac{1}{2}(\alpha_i+\beta_i)\big), \text{ if } x_{i+1} < x_i + (\beta_i - 3\alpha_i)/2; \\ (x_{i+1}+\alpha_i, x_i - \alpha_i), \text{ if } x_i + (\beta_i - 3\alpha_i)/2 \leq x_{i+1} < x_i - \alpha_i; \\ (x_i, x_{i+1}), \text{ if } x_i - \alpha_i \leq x_{i+1} \leq x_i - \beta_i; \\ (x_{i+1}+\beta_i, x_i - \beta_i), \text{ if } x_i - \beta_i < x_{i+1} \leq x_i + \min\{0, (\alpha_i - 3\beta_i)/2\}; \\ \tfrac{1}{2}\big(x_i + x_{i+1} + \tfrac{1}{2}(\alpha_i+\beta_i), x_i + x_{i+1} - \tfrac{1}{2}(\alpha_i+\beta_i)\big), \text{ if } x_i + \dfrac{\alpha_i - 3\beta_i}{2} < x_{i+1} \leq x_i; \\ \tfrac{1}{2}\big(x_i + x_{i+1} - \tfrac{1}{2}(\alpha_i+\beta_i), x_i + x_{i+1} + \tfrac{1}{2}(\alpha_i+\beta_i)\big), \text{ if } x_i < x_{i+1} \leq x_i + \dfrac{3\beta_i - \alpha_i}{2}; \\ (x_{i+1}-\beta_i, x_i + \beta_i), \text{ if } x_i + \max\{0, (3\beta_i - \alpha_i)/2\} \leq x_{i+1} < x_i + \beta_i; \\ (x_i, x_{i+1}), \text{ if } x_i + \beta_i \leq x_{i+1} \leq x_i + \alpha_i; \\ (x_{i+1}-\alpha_i, x_i + \alpha_i), \text{ if } x_i + \alpha_i < x_{i+1} \leq x_i + (3\alpha_i - \beta_i)/2; \\ \tfrac{1}{2}\big(x_i + x_{i+1} - \tfrac{1}{2}(\alpha_i+\beta_i), x_i + x_{i+1} + \tfrac{1}{2}(\alpha_i+\beta_i)\big), \text{ if } x_i + (3\alpha_i - \beta_i)/2 < x_{i+1}. \end{cases}$$

There are at least 8 cases; however, the two cases in the middle arise if and only if $3\beta_i > \alpha_i$.

Having recorded all required formulas, one can now experiment with the performance of the corresponding algorithms. In the physics community, Veit Elser has championed especially D–R with great success (see, e.g., [**38**] and [**46**]). Because of its central importance, we consider in the following subsection a curious cycling behaviour of D–R.

5.2. Representation of nonconvex constraints and cycling for D–R. Suppose momentarily that C_1, C_2, C_3 are nonempty closed convex subsets of X such that $C = C_1 \cap C_2 \cap C_3 \neq \varnothing$. Assume further that $C_1 \cap C_2$ is still simple enough to admit a formula for the projector $P_{C_1\cap C_2}$. The projection methods considered find points in C; it does not matter if we work with either the original three sets C_1, C_2, C_3 or with $C_1 \cap C_2$ and C_3. This situation changes dramatically if we are dealing with *nonconvex* constraints. Performance of projection methods crucially depends on the representation of the constraints.

In fact, the example developed next was discovered numerically. It forced us to deal with nonconvex constraints differently and led to formula (99).

EXAMPLE 5.1 (cycling for D–R). Suppose that $X = \mathbb{R}^2$, that $\alpha \geq \beta > 0$, and set

$$C_1 := \big\{(x_1,x_2) \mid |x_1 - x_2| \leq \alpha\big\} \quad\text{and}\quad C_2 := \big\{(x_1,x_2) \mid |x_1 - x_2| \geq \beta\big\}. \tag{101}$$

Clearly, $C_1 \cap C_2 \neq \varnothing$; in fact, (99) allows us to record a formula for $P_{C_1\cap C_2}$. Set

$$\varepsilon := \beta/5, \tag{102}$$

take an arbitrary $\xi \in \mathbb{R}$, and set

$$\mathbf{x}_0 = (x_{0,1}, x_{0,2}) = (\xi, \xi-\varepsilon, \xi-2\varepsilon, \xi+\varepsilon) \in \mathbb{R}^2 \times \mathbb{R}^2. \tag{103}$$

Then the sequences $(\mathbf{x}_k)_{k\in\mathbb{N}}$ and $(\bar{x}_k)_{k\in\mathbb{N}}$ generated by (47) do not converge; indeed, they cycle as follows:

$$(\forall k \in \mathbb{N}) \quad \mathbf{x}_{2k} = (\xi, \xi-\varepsilon, \xi-2\varepsilon, \xi+\varepsilon), \;\; \mathbf{x}_{2k+1} = (\xi-\varepsilon, \xi, \xi+\varepsilon, \xi-2\varepsilon) \tag{104}$$

and

$$(\forall k \in \mathbb{N}) \quad \bar{x}_{2k} = (\xi-\varepsilon, \xi), \;\; \bar{x}_{2k+1} = (\xi, \xi-\varepsilon). \tag{105}$$

PROOF. It is clear that $\bar{x}_0 = (\xi-\varepsilon, \xi)$. Observe that

$$(\forall(\xi,\eta) \in \mathbb{R}^2) \quad P_{C_2}(\xi,\eta) = \begin{cases} (\xi,\eta), & \text{if } |\xi-\eta| \geq \beta; \\ \frac{1}{2}(\xi+\eta-\beta, \xi+\eta+\beta), & \text{if } |\xi-\eta| < \beta \text{ and } \xi \leq \eta; \\ \frac{1}{2}(\xi+\eta+\beta, \xi+\eta-\beta), & \text{if } |\xi-\eta| < \beta \text{ and } \xi \geq \eta. \end{cases} \tag{106}$$

By definition,

$$\begin{aligned} x_{1,1} &= x_{0,1} - \bar{x}_0 + P_{C_1}(2\bar{x}_0 - x_{0,1}) && \text{(107a)}\\ &= (\xi, \xi-\varepsilon) - (\xi-\varepsilon, \xi) + P_{C_1}(2(\xi-\varepsilon,\xi) - (\xi,\xi-\varepsilon)) && \text{(107b)}\\ &= (\varepsilon, -\varepsilon) + P_{C_1}(\xi-2\varepsilon, \xi+\varepsilon). && \text{(107c)} \end{aligned}$$

Now $|(\xi-2\varepsilon)-(\xi+\varepsilon)| = 3\varepsilon = (3/5)\beta < \beta \leq \alpha$, so P_{C_1} does not modify $(\xi-2\varepsilon, \xi+\varepsilon)$ and we conclude that

$$x_{1,1} = (\varepsilon, -\varepsilon) + (\xi-2\varepsilon, \xi+\varepsilon) = (\xi-\varepsilon, \xi). \tag{108}$$

Next,

$$\begin{aligned} x_{1,2} &= x_{0,2} - \bar{x}_0 + P_{C_2}(2\bar{x}_0 - x_{0,2}) && \text{(109a)}\\ &= (\xi-2\varepsilon, \xi+\varepsilon) - (\xi-\varepsilon, \xi) + P_{C_2}(2(\xi-\varepsilon,\xi) - (\xi-2\varepsilon,\xi+\varepsilon)) && \text{(109b)}\\ &= (-\varepsilon, \varepsilon) + P_{C_2}(\xi, \xi-\varepsilon). && \text{(109c)} \end{aligned}$$

Now $|\xi - (\xi-\varepsilon)| = \varepsilon = \beta/5 < \beta$, so (106) yields

$$P_{C_2}(\xi, \xi-\varepsilon) = \tfrac{1}{2}(2\xi-\varepsilon+\beta, 2\xi-\varepsilon-\beta) = (\xi+2\varepsilon, \xi-3\varepsilon). \tag{110}$$

It follows that $x_{1,2} = (-\varepsilon,\varepsilon) + (\xi+2\varepsilon, \xi-3\varepsilon) = (\xi+\varepsilon, \xi-2\varepsilon)$. Altogether,

$$\mathbf{x}_1 = (x_{1,1}, x_{1,2}) = (\xi-\varepsilon, \xi, \xi+\varepsilon, \xi-2\varepsilon) \quad\text{and}\quad \bar{x}_1 = (\xi, \xi-\varepsilon). \tag{111}$$

Arguing similarly, we obtain that $\mathbf{x}_2 = \mathbf{x}_0$. □

Remark 5.2. If D–R is started at different points, as we recommend in Algorithm 3.12, then the iterates eventually settle into this cycling behaviour described in Example 5.1.

6. Numerical results

6.1. Experimental setup and stopping criteria.

6.1.1. *Generating the set of test problems.* We randomly generate 100 test problems, namely road splines in groups centered around length $L \in \{0.5, 1, 5, 10, 20\}$ (unit km). For each group, we select a design speed $V \in \{30, 50, 80, 100\}$ (with unit km/h) and maximum elevations $\xi_{\max} \in \{30, 60, 100, 120, 150\}$ (unit m). We then generate spline points $\{(t_1, v_1), \ldots, (t_n, v_n)\}$ such that $v \in [0, \xi_{\max}]^n$ and $(\forall i \in \{1, \ldots, n-1\})\ \|(t_i, v_i) - (t_{i+1}, v_{i+1})\| \geq 0.625V$, where the integer n lies in $[L/(3\min\{0.625V, 30\}), \ldots, 1 + L/(1.5\min\{0.625V, 30\})]$. The resulting splines correspond to rather challenging road profiles and are therefore ideal for testing.

6.1.2. *Stopping criteria.* The constraint sets C_i are as in Section 2 (and Section 5.1 in the nonconvex case). Let $(x_k)_{k\in\mathbb{N}}$ be a sequence to monitor generated by an algorithm under consideration and set $\varepsilon := 5 \cdot 10^{-3}$. The feasibility performance measure we employ is

$$d\colon X \to \mathbb{R}_+ \colon x \mapsto \sqrt{\frac{\sum_{i=1}^m d^2(x, C_i)}{\sum_{i=1}^m d^2(x_0, C_i)}}. \tag{112}$$

Note that d returns the value zero precisely at points drawn from $C = C_1 \cap \cdots \cap C_m$. If we are interested in feasibility only, then we terminate when

$$d(x_k) < \varepsilon; \tag{113}$$

in case of best approximation ($P_C v$, where $v \in X$ is given), we require additionally that $\|x_k - x_{k-1}\| < \varepsilon$. We cap the maximum number of iterations at $k_{\max} = 5000$.

6.2. Generation of plots and tables.

Let $\mathcal{P}$ be the set of test problems and $\mathcal{A}$ be the set of algorithms. Let $(x_k^{(a,p)})_{k\in\mathbb{N}}$ be the monitored sequence generated by algorithm $a \in \mathcal{A}$ applied to the problem $p \in \mathcal{P}$.

6.2.1. *Performance plots.* To compare the performance of the algorithms, we use *performance profiles*: for every $a \in \mathcal{A}$ and for every $p \in \mathcal{P}$, we set

$$r_{a,p} := \frac{k_{a,p}}{\min\left\{k_{a',p} \mid a' \in \mathcal{A}\right\}} \geq 1, \tag{114}$$

where $k_{a,p} \in \{1, 2, \ldots, k_{\max}\}$ is the number of iterations that a requires to solve p (see Section 6.1.2). If $r_{a,p} = 1$, then a uses the least number of iterations to solve problem p. If $r_{a,p} > 1$, then a requires $r_{a,p}$ times more iterations for p than the algorithm that uses the least iterations for p. For each algorithm $a \in \mathcal{A}$, we plot the function

$$\rho_a\colon \mathbb{R}_+ \to [0,1]\colon \kappa \mapsto \frac{\operatorname{card}\left\{p \in \mathcal{P} \mid \log_2(r_{a,p}) \leq \kappa\right\}}{\operatorname{card}\mathcal{P}}, \tag{115}$$

where "card" denotes the cardinality of a set. Thus, $\rho_a(\kappa)$ is the percentage of problems that algorithm a solves within factor 2^κ of the best algorithms. Therefore, an algorithm $a \in \mathcal{A}$ is "fast" if $\rho_a(\kappa)$ is large for κ small; and a is "robust" if $\rho_a(\kappa)$

is large for κ large. For further information on performance profiles we refer the reader to [**36**].

6.2.2. *Runtime plots.* $(d(x_k^{(a,p)}))_{k\in\mathbb{N}}$ measures the runtime progress of a on p with respect to the feasibility performance measure d (see (112)). To get a sense of the average (logarithmatized) progress of each algorithm $a \in \mathcal{A}$ at iteration k, we follow [**29**] and plot the values of *relative proximity function*, which is defined by

$$\beta_a \colon \mathbb{N} \to \mathbb{R} \colon k \mapsto 10\log_{10}\left(\frac{1}{\operatorname{card}\mathcal{P}}\sum_{p\in\mathcal{P}} d^2(x_k^{(a,p)})\right) \tag{116a}$$

$$= 10\log_{10}\left(\frac{1}{\operatorname{card}\mathcal{P}}\sum_{p\in\mathcal{P}} \frac{\sum_{i=1}^m \|P_{C_i}x_k^{(a,p)} - x_k^{(a,p)}\|^2}{\sum_{i=1}^m \|P_{C_i}x_0^{(a,p)} - x_0^{(a,p)}\|^2}\right). \tag{116b}$$

6.2.3. *Distance tables.* For each algorithm $a \in \mathcal{A}$ and problem $p \in \mathcal{P}$, assume that termination occurred at iteration $k(a,p)$. We compute the normalized distance to v by

$$\Delta_{(a,p)} := \begin{cases} \|v - x_{k(a,p)}^{(a,p)}\|/\|v\|, & \text{if } k_{(a,p)} < k_{\max};\\ \max\big\{\|v - x_k^{(a',p)}\| \mid a' \in \mathcal{A}\big\}/\|v\|, & \text{otherwise.}\end{cases} \tag{117}$$

This allows us to consider the collection of normalized distances $(\Delta_{(a,p)})_{p\in\mathcal{P}}$ for each algorithm $a \in \mathcal{A}$. Statistical values are recorded for each algorithm in a table allowing us to compare best approximation performance.

6.3. Results for feasibility algorithms.

6.3.1. *The convex case.* In this section, we record the results for the convex setting of Section 2 using the algorithms of Section 3.4.

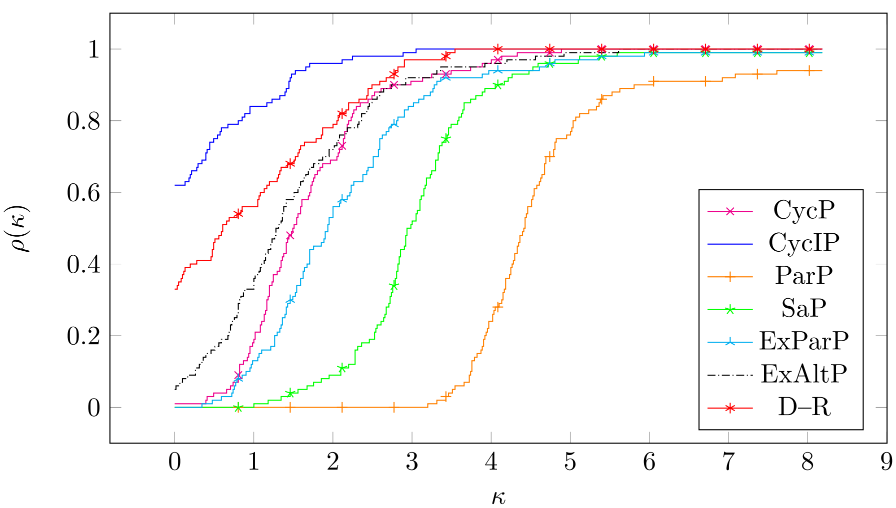

FIGURE 3. Performance profiles for feasibility algorithms in the convex case (see Section 6.2.1)

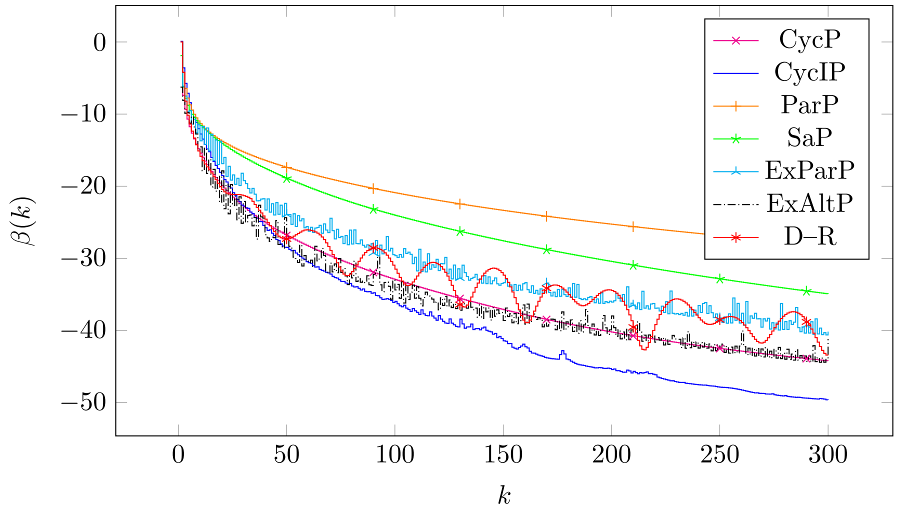

FIGURE 4. Runtime plots for feasibility algorithms in the convex case (see Section 6.2.2)

Algo.	Δ Min.	$\Delta 1^{\text{st}}$ Qrt.	Δ Median	$\Delta 3^{\text{rd}}$ Qrt.	Δ Max.	Δ Mean	Δ Std.dev.
CycP	0.0563	0.3039	0.3696	0.4806	0.9301	0.3993	0.1568
CycIP	0.0691	0.2840	0.3588	0.4419	0.7708	0.3767	0.1389
ParP	0.0517	0.2973	0.3687	0.4770	0.9191	0.3990	0.1612
SaP	0.0563	0.3033	0.3693	0.4795	0.9292	0.3993	0.1577
ExParP	0.0522	0.3012	0.3697	0.4767	0.9257	0.3981	0.1572
ExAltP	0.0522	0.3012	0.3697	0.4767	0.9257	0.3973	0.1558
D-R	0.0721	0.3833	0.4555	0.5486	1.1353	0.4807	0.1861

TABLE 1. Statistical data for feasibility algorithms in the convex case (see Section 6.2.3)

6.3.2. *The nonconvex case.* The plots and tables are analogous to those of Section 6.3.1 except that we work with the nonconvex slope constraint (see Section 5.1).

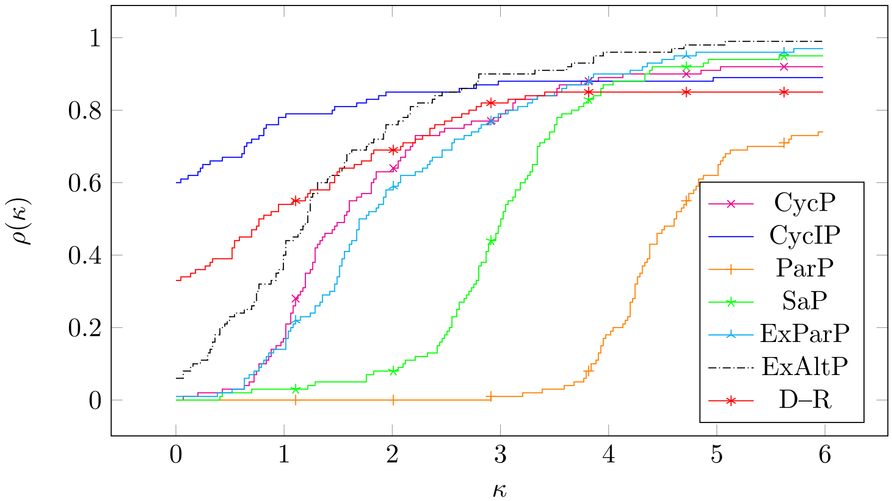

FIGURE 5. Performance profiles for feasibility algorithms in the nonconvex case (see Section 6.2.1)

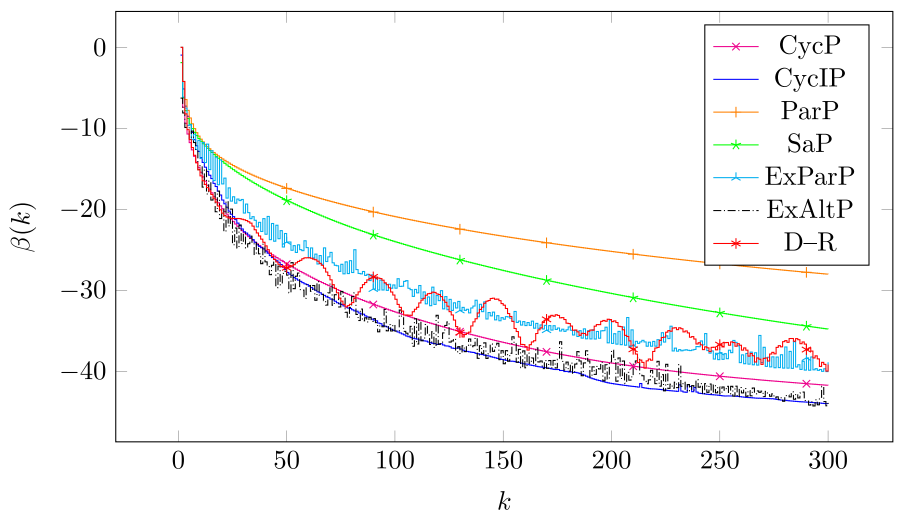

FIGURE 6. Runtime plots for feasibility algorithms in the nonconvex case (see Section 6.2.2)

Algo.	Δ Min.	$\Delta 1^{\text{st}}$ Qrt.	Δ Median	$\Delta 3^{\text{rd}}$ Qrt.	Δ Max.	Δ Mean	Δ Std.dev.
CycP	0.0566	0.3039	0.3871	0.4843	0.9301	0.4093	0.1628
CycIP	0.0691	0.2923	0.3674	0.4547	0.7707	0.3886	0.1426
ParP	0.0522	0.3004	0.3919	0.4853	0.9275	0.4137	0.1653
SaP	0.0566	0.3036	0.3820	0.4824	0.9292	0.4027	0.1574
ExParP	0.0527	0.3031	0.3796	0.4786	0.9306	0.4026	0.1603
ExAltP	0.0526	0.3022	0.3698	0.4771	0.9243	0.3981	0.1566
D-R	0.0818	0.3835	0.4567	0.5486	1.1044	0.4813	0.1840

TABLE 2. Statistical data for feasibility algorithms in the nonconvex case (see Section 6.2.3)

6.3.3. *Conclusions.* Overall, CycIP, ExAltP, and D–R emerge as good algorithms for feasibility. CycIP yields solutions closest to the initial spline and is the fastest algorithm, followed by D–R and ExAltP. ExAltP is the most robust in the nonconvex case.

6.4. Results for superiorized feasibility algorithms.

6.4.1. *The convex case.* In this section, we record the results for the convex setting of Section 2 using superiorized feasibility algorithms (see Section 3.5).

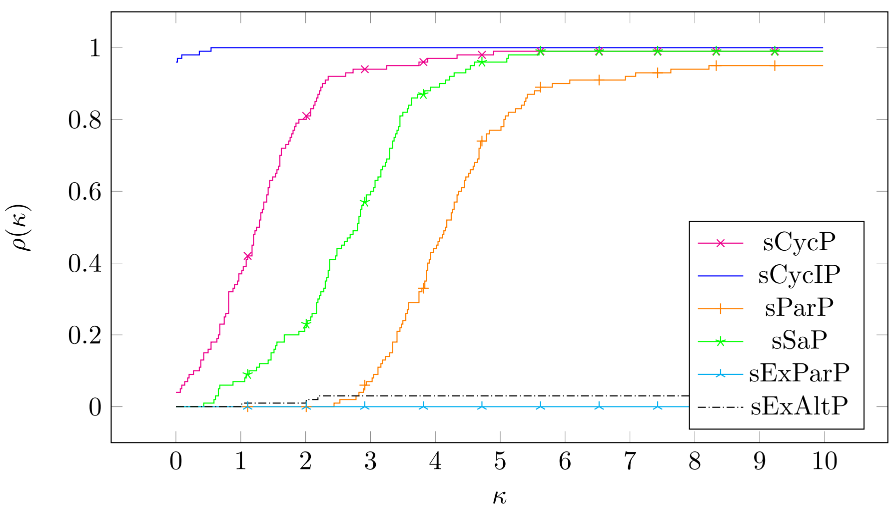

FIGURE 7. Performance profiles for superiorized feasibility algorithms in the convex case (see Section 6.2.1)

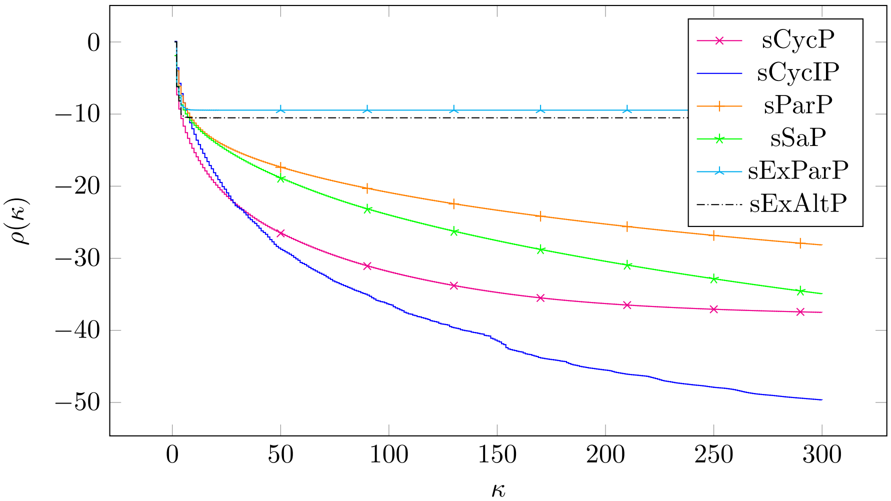

FIGURE 8. Runtime plots for superiorized feasibility algorithms in the convex case (see Section 6.2.2)

Algo.	Δ Min.	$\Delta 1^{\text{st}}$ Qrt.	Δ Median	$\Delta 3^{\text{rd}}$ Qrt.	Δ Max.	Δ Mean	Δ Std.dev.
sCycP	0.0519	0.3039	0.3696	0.4805	0.9301	0.3992	0.1570
sCycIP	0.0691	0.2840	0.3588	0.4419	0.7708	0.3767	0.1389
sParP	0.0517	0.2973	0.3659	0.4762	0.9191	0.3940	0.1559
sSaP	0.0519	0.3033	0.3693	0.4795	0.9292	0.3985	0.1567
sExParP	0.0691	0.3039	0.3710	0.4825	0.9301	0.4021	0.1551
sExAltP	0.0519	0.3039	0.3710	0.4825	0.9301	0.4016	0.1562

TABLE 3. Statistical data for superiorized feasibility algorithms in the convex case (see Section 6.2.3)

6.4.2. *The nonconvex case.* The plots and tables are analogous to those of Section 6.4.1 except that we work with the nonconvex slope constraint (see Section 5.1).

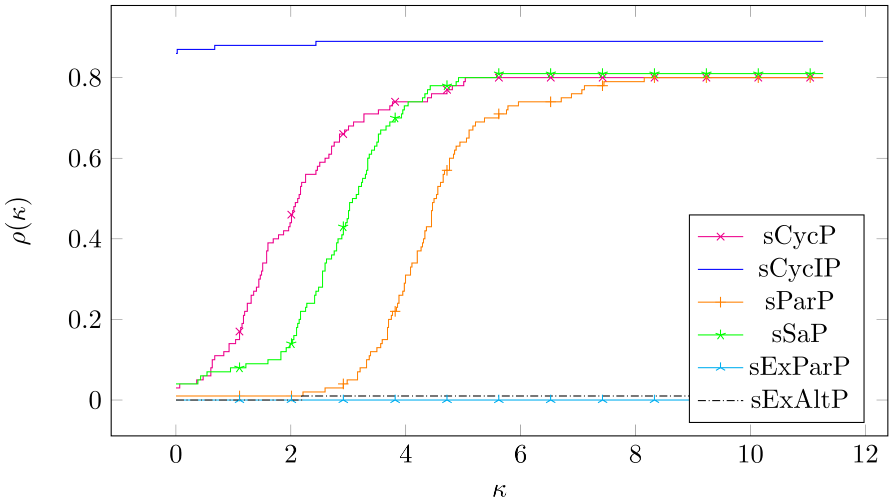

FIGURE 9. Performance profiles for superiorized feasibility algorithms in the nonconvex case (see Section 6.2.1)

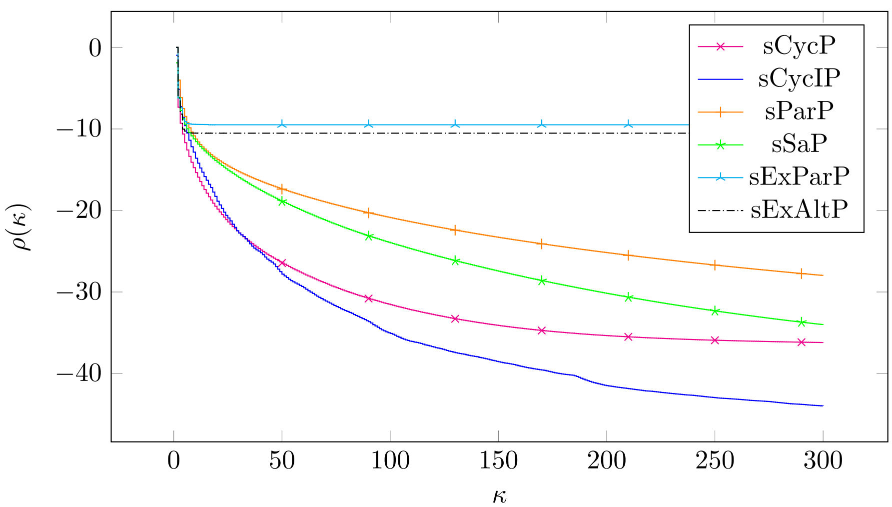

FIGURE 10. Runtime plots for superiorized feasibility algorithms in the nonconvex case (see Section 6.2.2)

Algo.	Δ Min.	Δ1^{st} Qrt.	Δ Median	Δ3^{rd} Qrt.	Δ Max.	Δ Mean	Δ Std.dev.
sCycP	0.0566	0.3039	0.3790	0.4759	0.9301	0.4004	0.1562
sCycIP	0.0691	0.2923	0.3662	0.4473	0.7707	0.3811	0.1382
sParP	0.0522	0.3004	0.3790	0.4744	0.9275	0.3975	0.1554
sSaP	0.0522	0.3036	0.3718	0.4734	0.9292	0.3980	0.1556
sExParP	0.0691	0.3042	0.3810	0.4762	0.9301	0.4030	0.1547
sExAltP	0.0523	0.3042	0.3810	0.4762	0.9301	0.4025	0.1558

TABLE 4. Statistical data for superiorized feasibility algorithms in the nonconvex case (see Section 6.2.3)

6.4.3. *Conclusions.* We clearly see that sCycIP is not only the fastest but also the most robust superiorized feasibility algorithm.

6.5. Results for best approximation algorithms.

6.5.1. *The convex case.* In this section, we record the results for the convex setting of Section 2 using the best approximation algorithms of Section 4.

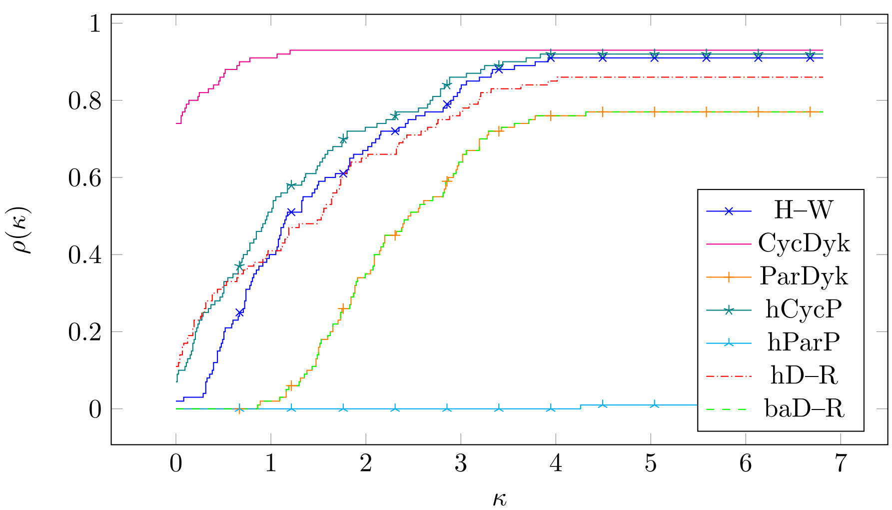

FIGURE 11. Performance profiles for best approximation algorithms in the convex case (see Section 6.2.1)

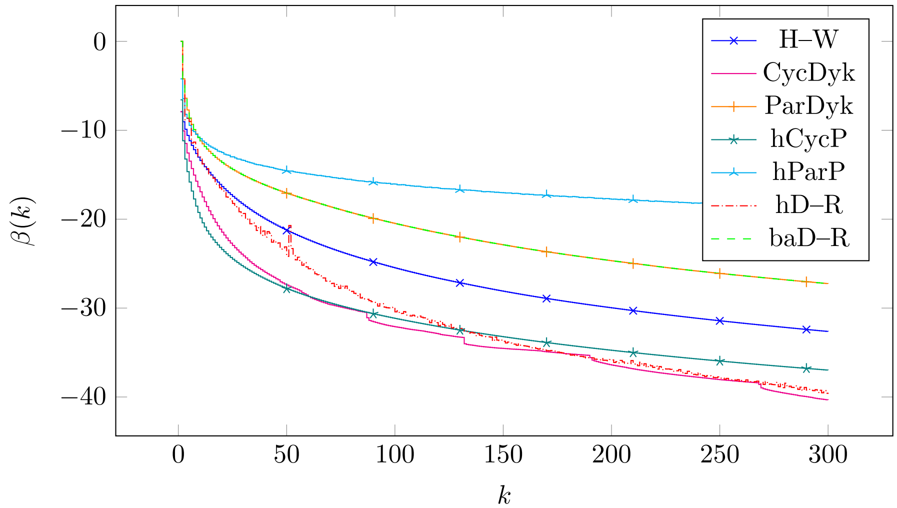

FIGURE 12. Runtime plots for best approximation algorithms in the convex case (see Section 6.2.2)

Algo.	Δ Min.	Δ1^{st} Qrt.	Δ Median	Δ3^{rd} Qrt.	Δ Max.	Δ Mean	Δ Std.dev.
H–W	0.0525	0.2978	0.3644	0.4800	0.9251	0.3959	0.1570
CycDyk	0.0519	0.3022	0.3695	0.4862	0.9379	0.4006	0.1584
ParDyk	0.0517	0.2962	0.3656	0.4799	0.9379	0.3972	0.1576
hCycP	0.0519	0.2986	0.3645	0.4801	0.9251	0.3962	0.1569
hParP	0.0517	0.3042	0.3720	0.4864	0.9379	0.4016	0.1585
hD–R	0.0519	0.3042	0.3720	0.4864	0.9337	0.4013	0.1579
D–Rba	0.0517	0.2962	0.3656	0.4799	0.9379	0.3972	0.1576

TABLE 5. Statistical data for best approximation algorithms in the convex case (see Section 6.2.3)

6.5.2. *The nonconvex case.* The plots and tables are analogous to those of Section 6.5.1 except that we work with the nonconvex slope constraint (see Section 5.1).

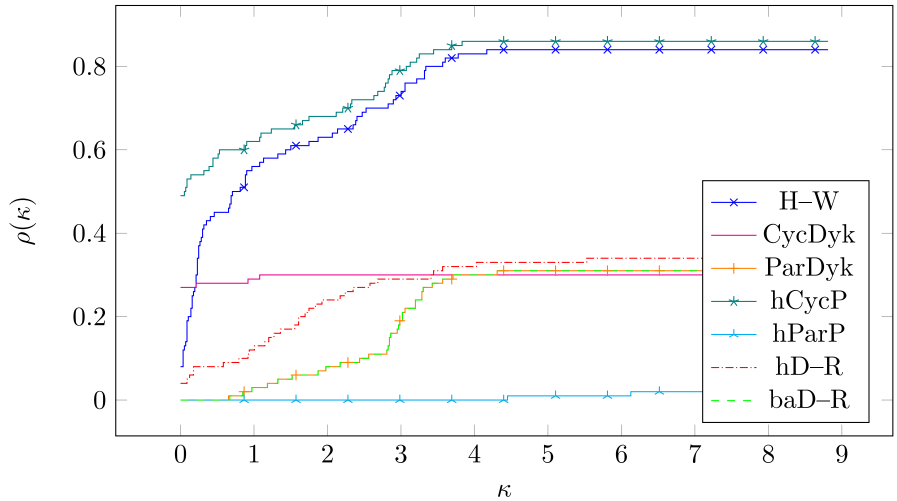

FIGURE 13. Performance profiles for best approximation algorithms in the nonconvex case (see Section 6.2.1)

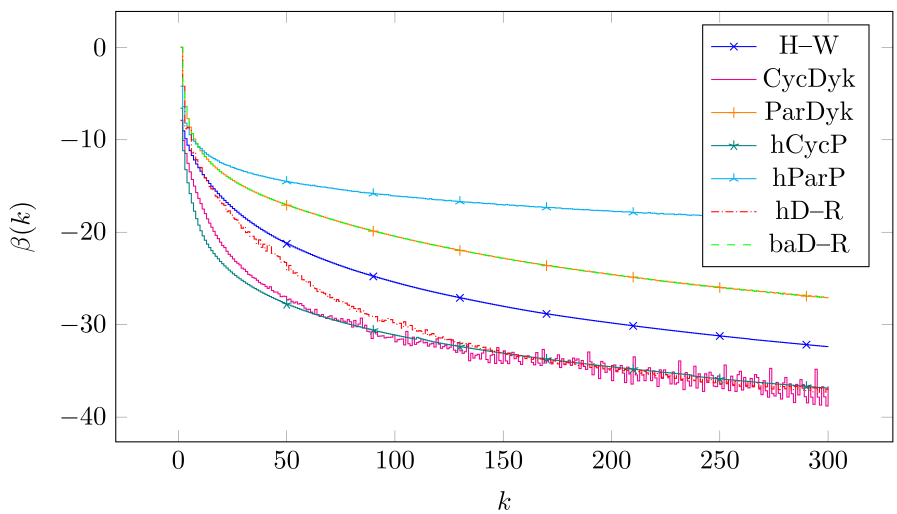

FIGURE 14. Runtime plots for best approximation algorithms in the nonconvex case (see Section 6.2.2)

Algo.	Δ Min.	Δ1st Qrt.	Δ Median	Δ3rd Qrt.	Δ Max.	Δ Mean	Δ Std.dev.
H–W	0.0525	0.3019	0.3794	0.4825	0.9252	0.3998	0.1570
CycDyk	0.0667	0.3053	0.3861	0.4863	0.9385	0.4079	0.1539
ParDyk	0.0667	0.3006	0.3846	0.4863	0.9385	0.4060	0.1529
hCycP	0.0522	0.3022	0.3782	0.4808	0.9253	0.3998	0.1568
hParP	0.0520	0.3057	0.3863	0.4864	0.9385	0.4069	0.1570
hD–R	0.0652	0.3057	0.3825	0.4864	0.9342	0.4074	0.1540
baD–R	0.0667	0.3006	0.3846	0.4863	0.9385	0.4060	0.1529

TABLE 6. Statistical data for best approximation algorithms in the nonconvex case (see Section 6.2.3)

6.5.3. *ParDyk vs baD–R.* The plots and tables of Sections 6.5.1 and 6.5.2 suggest that ParDyk and baD–R are the same algorithms. This is not true for these algorithms in their full generality as we pointed out in Remark 4.16. A possible explanation for this identical performance could be that the algorithms produce sequences that stay either outside or inside the halfspaces comprising the constraints. If that is indeed the case, then the projection onto the halfspace would be indistinguishable from the projection onto either a hyperplane or the entire space; consequently, the iterates would be identical by Remark 4.15.

6.5.4. *Conclusions.* hCycP is a good robust choice for both convex and nonconvex problems; CycDyk does well for convex problems, and H–W does well for nonconvex problems.

Examining the data for all three types of algorithms (feasibility, superiorization, best approximation), we can make a compelling case for CycIP, which emerges as the best overall algorithm with respect to speed, robustness, and best approximation properties.

6.6. Projection methods vs Linear Programming (LP) algorithms. In Section 3.2, we made the case for projection methods and commented on their competitiveness with other optimization methods. To illustrate this claim, we interpreted (33) as the constraints of a *Linear Programming (LP)* problem. We then solved our test problems with the *GNU Linear Programming Kit (GLPK)* [**42**]. Two objective functions were employed: $x \mapsto 0$, which corresponds to the feasibility problem (34), and $x \mapsto \|x - v\|_1$, which is similar to the best approximation problem[13] (35). We call these methods GLPK0 and GLPK1, respectively [**63**, page 257]. We shall compare these two methods against the overall best projection method, CycIP.

Name	Acronym
Cyclic Intrepid Projections	CycIP
0-objective function LP	GLPK0
ℓ_1-minimization LP	GLPK1

[13]Here, $\|x\|_1 = \sum_{i=1}^{n} |x_i|$ denotes the ℓ_1-norm of $x \in X$.

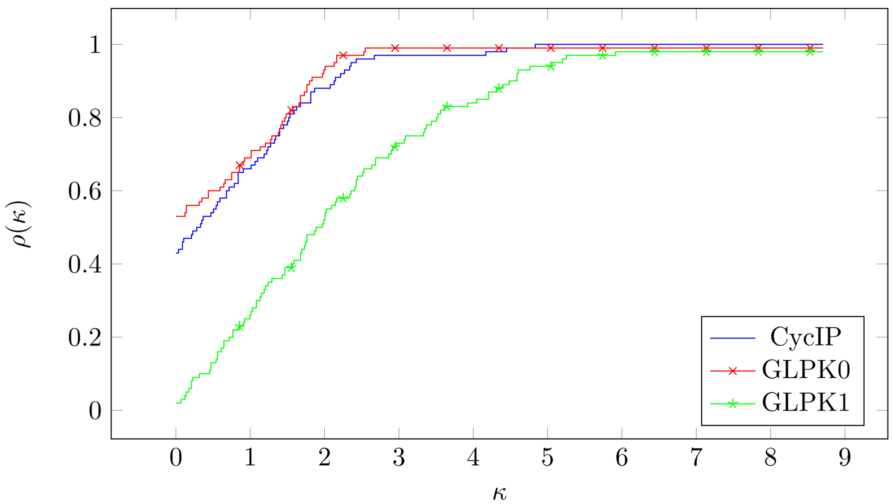

FIGURE 15. Performance profiles for convex feasibility problems comparing the number of iterations of CycIP, GLPK0 and GLPK1 until convergence (see Section 6.2.1)

Algo.	Δ Min.	$\Delta 1^{\text{st}}$ Qrt.	Δ Median	$\Delta 3^{\text{rd}}$ Qrt.	Δ Max.	Δ Mean	Δ Std.dev.
CycIP	0.0691	0.2840	0.3588	0.4419	0.7708	0.3767	0.1389
GLPK0	0.1938	0.6636	0.8030	0.9281	1.0039	0.7768	0.1811
GLPK1	0.0000	0.3062	0.3813	0.5001	0.9406	0.4076	0.1625

TABLE 7. Statistical data in the convex case (see Section 6.2.3)

The nonconvex slope constraints from Section 5.1 can be handled by using *binary variables*; this leads to a *Mixed Integer Linear Programming (MIP)* problem (see, e.g., [**53**, Chapter 12]), which GLPK [**42**] can handle as well.

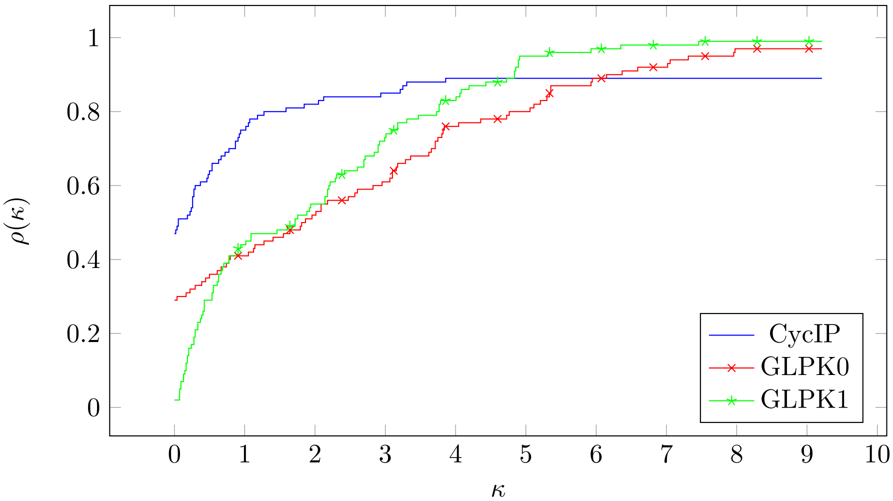

FIGURE 16. Performance profiles for nonconvex feasibility problems comparing the number of iterations of CycIP, GLPK0 and GLPK1 until convergence (see Section 6.2.1)

Algo.	Δ Min.	$\Delta 1^{\text{st}}$ Qrt.	Δ Median	$\Delta 3^{\text{rd}}$ Qrt.	Δ Max.	Δ Mean	Δ Std.dev.
CycIP	0.0691	0.2923	0.3674	0.5375	6.2070	0.5077	0.6928
GLPK0	0.1938	0.5371	0.6245	0.8240	6.2070	0.8226	0.7464
GLPK1	0.0840	0.3127	0.3847	0.5001	2.1468	0.4261	0.2325

TABLE 8. Statistical data in the nonconvex case (see Section 6.2.3)

7. Concluding remarks

Using the practical example of road design, we formulated (convex and nonconvex) feasibility and best approximation problems. We studied projection methods and implemented them to solve the feasibility problems. A clear winner emerged: CycIP, the method of cyclic intrepid projections. It compares well even to LP solvers.

In the future, we plan to study the influence of parameters on the performance of the algorithms presented. The design and testing of hybrid methods that aim to combine advantageous traits of various algorithms is also of considerable interest. Finally, rigorous convergence statements about CycIP in the nonconvex setting await to be discovered.

References

[1] American Association of State Highway and Transportation Officials, *A Policy on Geometric Design of Highways and Streets*, sixth edition, Washington, D.C., 2011.

[2] H.H. Bauschke and J.M. Borwein, Dykstra's alternating projection algorithm for two sets, *J. Approx. Theory* 79 (1994), 418–443.

[3] H.H. Bauschke and J.M. Borwein, On projection algorithms for solving convex feasibility problems, *SIAM Rev.* 38(3) (1996), 367–426.

[4] H.H. Bauschke, R.I. Boţ, W.L. Hare, and W.M. Moursi, Attouch-Théra duality revisited: paramonotonicity and operator splitting, *J. Approx. Theory* 164 (2012), 1065–1084.

[5] H.H. Bauschke and P.L. Combettes, A weak-to-strong convergence principle for Fejér-monotone methods in Hilbert spaces, *Math. Oper. Res.* 26(2) (2001), 248–264.

[6] H.H. Bauschke and P.L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, Springer, 2011.

[7] H.H. Bauschke, P.L. Combettes, and S.G. Kruk, Extrapolation algorithm for affine-convex feasibility problems, *Numer. Algorithms* 41(3) (2006), 239–274.

[8] H.H. Bauschke, P.L. Combettes, and D.R. Luke, Finding best approximation pairs relative to two closed convex sets in Hilbert spaces, *J. Approx. Theory* 127 (2004), 178–192.

[9] H.H. Bauschke, P.L. Combettes, and S. Reich, The asymptotic behavior of the composition of two resolvents, *Nonlinear Anal.* 60 (2005), 283–301.

[10] H.H. Bauschke and V.R. Koch, Projection Methods: Swiss Army Knives for Solving Feasibility and Best Approximation Problems with Halfspaces, arXiv preprint arXiv:1301.4506v1 (January 2013). Available at URL `http://arxiv.org/abs/1301.4506`

[11] H.H. Bauschke and S.G. Kruk, Reflection-projection method for convex feasibility problems with an obtuse cone, *J. Optim. Theory Appl.* 12(3) (2004), 503–531.

[12] H.H. Bauschke, D.R. Luke, H.M. Phan, and X. Wang, Restricted normal cones and the method of alternating projections: theory, *Set-Valued Anal.* (2012), in press. `DOI: 10.1007/s11228-013-0239-2`

[13] H.H. Bauschke, D.R. Luke, H.M. Phan, and X. Wang, Restricted normal cones and the method of alternating projections: applications, *Set-Valued Anal.* (2012), in press. `DOI: 10.1007/s11228-013-0238-3`

[14] H.H. Bauschke, E. Matoušková, and S. Reich, Projection and proximal point methods: convergence results and counterexamples, *Nonlinear Anal.* 56 (2004), 715–738.

[15] R. Bellman, On the approximation of curves by line segments using dynamic programming, *Comm. ACM* 4(6) (1961), 284.

[16] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, *Found. Trends Mach. Learn.* 3(1) (2010), 1–122.

[17] S. Boyd and J. Mattingley, CVXGEN: A Code Generator for Embedded Convex Optimization, *Optim. Eng.* 13(1) (2012), 1–27.

[18] J.P. Boyle and R.L. Dykstra, A method for finding projections onto the intersection of convex sets in Hilbert spaces, in *Advances in Order Restricted Statistical Interference*, Lecture Notes in Statistics 37, Springer, 1986.

[19] A. Cegielski, *Iterative Methods for Fixed Point Problems in Hilbert Spaces*, Springer, 2012.

[20] Y. Censor, W. Chen, P.L. Combettes, R. Davidi, and G.T. Herman, On the effectiveness of projection methods for convex feasibility problems with linear inequality constraints, *Comput. Optim. Appl.* 51(3) (2012), 1065–1088.

[21] Y. Censor, W. Chen, and H. Pajoohesh, Finite convergence of a subgradient projections method with expanding controls, *Appl. Math. Optim.* 64 (2011), 273–285.

[22] Y. Censor, R. Davidi, and G.T. Herman, Pertubation resilience and superiorization of iterative algorithms, *Inverse Problems* 26(6) (2010), 065008 (12 pages).

[23] Y. Censor and T. Elfving, New methods for linear inequalities, *Linear Algebra Appl.* 42 (1982), 199–211.

[24] Y. Censor, T. Elfving, and G.T. Herman, Averaging strings of sequential iterations for convex feasibility problems, in *Inherently Parallel Algorithms in Feasibility and Optimization and Their Applications*, D. Butnariu, Y. Censor, and S. Reich (editors), pp. 101–114, Elsevier, 2001.

[25] Y. Censor and E. Tom, Convergence of string averaging projection schemes for inconsistent feasibility problems, *Optim. Methods Softw.* 18 (2003), 543–554.

[26] Y. Censor and S.A Zenios, *Parallel Optimization*, Oxford University Press, 1997.

[27] W. Chen, D. Craft, T.M. Madden, K. Zhang, H.M. Kooy, and G.T. Herman, A fast optimization algorithm for multicriteria intensity modulated proton therapy planning, *Med. Phys.* 37(9) (2010), 4938–4945.

[28] W. Chen and G.T. Herman, Efficient controls for finitely convergent sequential algorithms, *ACM Trans. Math. Software* 37(2) (2010) 14 (23 pages).

[29] P.L. Combettes, Convex set theoretic image recovery by extrapolated iterations of parallel subgradient projections, *IEEE Trans. Image Process.* 6(4) (1997), 493–506.

[30] P.L. Combettes, Hilbertian convex feasibility problems: convergence of projection methods, *Appl. Math. Optim.* 35 (1997), 311–330.

[31] P.L. Combettes, Đinh Dũng, and B.C. Vũ, Dualization of signal recovery problems, *Set–Valued Anal.* 18 (2010), 373–404.

[32] P.L. Combettes and J.-C. Pesquet, Proximal splitting methods in signal processing, Chapter 10 in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, Springer, 2011.

[33] R. Davidi, R.W. Schulte, Y. Censor, and L. Xing, Fast superiorization using a dual perturbation scheme for proton computed tomography, *Trans. Amer. Nucl. Soc.* 106 (2012), 73–76.

[34] C. de Boor, *A Practical Guide to Splines*, revised edition, Springer, 2001.

[35] F. Deutsch, *Best Approximation in Inner Product Spaces*, Springer, 2001.

[36] E.D. Dolan and J.J. Moré, Benchmarking optimization software with performance profiles, *Math. Program. (Ser. A)* 91 (2002), 201–213.

[37] J. Eckstein and D.P. Bertsekas, *On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators*, *Math. Program. (Ser. A)* 55 (1992), 293–318.

[38] V. Elser, I. Rankenburg, and P. Thibault, Searching with iterated maps, *Proc. Natl. Acad. Sci. U.S.A.* 104(2) (2007), 418–423.

[39] D. Gabay and B. Mercier, A dual algorithm for the solution of nonlinear variational problems via finite elements approximations, *Computers Math. Appl.* 2 (1976), 17–40.

[40] E. Garduño, G.T. Herman, and R. Davidi, Reconstruction from a few projections by ℓ_1-minimization of the Haar transform, *Inverse Problems* 27 (2011) 055006 (13 pages).

[41] R. Glowinski and A. Marrocco, Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité, d'une classe de problèmes de Dirichlet non linéaires, *RAIRO Anal. Num.* 2 (1975), 41–76.

[42] GNU Linear Programming Kit, Version 4.45, 2010. Available at URL `http://www.gnu.org/software/glpk/glpk.html`

[43] K. Goebel and W.A. Kirk, *Topics in Metric Fixed Point Theory*, Cambridge University Press, 1990.

[44] K. Goebel and S. Reich, *Uniform Convexity, Hyperbolic Geometry, and Nonexpansive Mappings*, Marcel Dekker, 1984.

[45] N.I.M. Gould, How good are projection methods for convex feasibility problems? *Comput. Optim. Appl.* 40 (2008), 1–12.

[46] S. Gravel and V. Elser, Divide and concur: a general approach to constraint satisfaction, *Phys. Rev. E* 78 (2008) 036706 (5 pages).

[47] B. Halpern, Fixed points of nonexpanding maps, *Bull. Amer. Math. Soc.* 73 (1967), 957–961.

[48] Y. Haugazeau, *Sur les Inéquations Variationnelles et la Minimisation de Foctionnelles Convexes*, Ph.D. thesis, Université de Paris, 1968.

[49] G.T. Herman, A relaxation method for reconstructing objects from noisy X-rays, *Math. Program.* 8 (1975), 1–19.

[50] G.T. Herman, *Fundamentals of Computerized Tomography: Image Reconstruction from Projections*, second edition, Springer, 2009.

[51] G.T. Herman and W. Chen, A fast algorithm for solving a linear feasibility problem with application to intensity-modulated radiation therapy, *Linear Algebra Appl.* 428 (2008), 1207–1217.

[52] G.T. Herman, E. Garduño, R. Davidi, and Y. Censor, Superiorization: an optimization heuristic for medical physics, *Med. Phys.* 39(9) (2012), 5532–5546.

[53] F.S. Hillier and G.J. Lieberman, *Introduction to Operations Research*, seventh edition, McGraw Hill, 2001.

[54] W. Jin, Y. Censor, and M. Jiang, A heuristic superiorization-like approach to bioluminescence tomography, *International Federation for Medical and Biological Engineering Proceedings* 39 (2013), 1026–1029.

[55] V.R. Koch, Road Design Optimization, *US patent application no.* 13/626,451, filed on September 25, 2012.

[56] P.-L. Lions and B. Mercier, Splitting algorithms for the sum of two nonlinear operators, *SIAM J. Numer. Anal.* 16 (1979), 964–979.

[57] T. Nikazad, R. Davidi, and G.T. Herman, Accelerated perturbation-resilient block-iterative projection methods with application to image reconstruction, *Inverse Problems* 28 (2012) 035005 (19 pages).

[58] N. Parikh and S. Boyd, Proximal algorithms, *Found. Trends Optim.* (2013), to appear.

[59] S.N. Penfold, R.W. Schulte, Y. Censor, and A.B. Rosenfeld, Total variation superiorization schemes in proton computed tomography image reconstruction, *Med. Phys.* 37(11) (2010), 5887–5895.

[60] S. Reich, A limit theorem for projections, *Linear Multilinear Algebra* 13 (1983), 281–290.

[61] L.L. Schumaker, *Spline Functions: Basic Theory*, third edition, Cambridge University Press, 2007.

[62] T. Strohmer and R. Vershynin, A randomized Kaczmarz algorithm with exponential convergence, *J. Fourier Anal. Appl.* 15 (2009), 262–278.

[63] H.P. Williams, *Model Building in Mathematical Programming*, third edition, Wiley, 1990.

[64] R. Wittmann, Approximation of fixed points of nonexpansive mappings, *Arch. Math. (Basel)* 58 (1992), 486–491.

Mathematics, Irving K. Barber School, University of British Columbia, Kelowna, B.C. V1V 1V7, Canada.
E-mail address: heinz.bauschke@ubc.ca

Information Modeling & Platform Products Group (IPG), Autodesk, Inc.
E-mail address: valentin.koch@autodesk.com