The Keble Small Research Grant (KSRG) KSRG118 was used to remunerate Claudio Vestini, a third-year engineering science undergraduate at Keble, for his work over seven weeks during the long vacation. The majority of the technical work was carried out by Claudio, and weekly meetings were held throughout the project. The KSRG successfully enabled me to launch a new research stream, and the research outputs will be incorporated into a future publication with Claudio as an author, in which KSRG support will be gratefully acknowledged. Additionally, the program code developed during this project will be made publicly available upon publication.

Summary

Euclidean projections are ubiquitous in engineering, optimisation, and machine learning, playing a central role in many algorithms for solving problems constrained by complex sets. Mathematically, the projection of a point x° onto a set S is defined as the closest point $x^\star \in S$ to x°, as illustrated in Figure 1 for two dimensions. Graphically, this definition can be extended to the projection of a set of points onto another set to obtain the shadow of an object in 3D. In constrained optimisation problems, this definition is extended to N dimensions, modelling the projection of optimisation variables onto a constraint set. For example, the constraint set could represent delivery stops and available trucks in an optimisation problem that aims to minimise delivery times and fuel consumption. Such an optimisation problem is typically solved using a *solver* – an algorithm that iteratively approaches an optimum while projecting the variables onto a constraint set. However, the projection itself defines another optimisation problem that is difficult to solve in general. Most solvers therefore simplify the projection by reformulating the optimisation problem and adding auxiliary variables related to the constraints, usually at the expense of accuracy and computational efficiency.

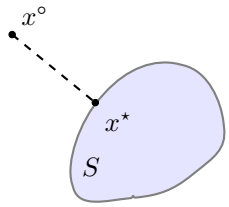


Figure 1: Illustration of the two-dimensional projection $x^\star$ of a point x° onto a set S.

An alternative to solving the complicated projection problem could be to use a specialised algorithm, *Dykstra's Alternating Projection*, an iterative procedure developed in the 1980s. However, although Dykstra's algorithm has proven to converge in theory, several practical issues remain. For example, it has been shown that the algorithm can *stall* in certain situations for an indefinite number of iterations, yielding inaccurate solutions or decreasing computational efficiency. In this project, we successfully addressed and solved the stalling problem associated with projections onto polyhedral sets, i.e. an intersection of multi-variable (in)equalities each of which can be interpreted as an N-dimensional half-space. Polyhedral sets can be used to approximate most constraint sets in practice and therefore play a particularly important role in engineering and optimisation.

To solve this problem, stalling situations referenced in the literature were replicated both graphically and mathematically. The mathematical properties of polyhedral sets were then used to simplify Dykstra's algorithm. Following an analysis of the simplified algorithm during stalling scenarios, a procedure for accurately predicting the *length* of the stalling period was identified. This procedure was then embedded in the algorithm to "fast-forward" the stalling period. These results are summarised in the attached technical report, which will serve as a draft for a future publication. This publication will also address additional research questions, such as a detailed characterisation of the convergence properties of the modified algorithm. The fast-forwarding approach could also be applied to similar algorithms that have been shown to have stalling issues.

Fast-Forwarding Stalling of Dykstra's Alternating Projection for Polyhedral Sets*

Claudio Vestini and Idris Kempf

Michaelmas Term 2024

1 Introduction

The projection of a point $x^\circ \in \mathbb{R}^n$ onto a closed convex set $\mathcal{H} \subset \mathbb{R}^p$ is defined as the point $x^\star \coloneqq \mathcal{P}_\mathcal{H}(x)$ that minimises the Euclidean distance between x° and any point in $\mathcal{H}$:

$$\mathcal{P}_\mathcal{H}(x) \coloneqq \underset{x \in \mathcal{H}}{\arg\min} \, \|x - x^\circ\|_2. \tag{1}$$

By the convexity of the set $\mathcal{H}$, problem (1) admits a unique solution [2, Ch. 3.2]. Here, we assume that $\mathcal{H}$ is a convex polyhedral set that can be represented as

$$\mathcal{H} \coloneqq \{x \in \mathbb{R}^p \mid Ax \leq c\}, \tag{2}$$

where $A \in \mathbb{R}^{n \times p}$ and $c \in \mathbb{R}^n$. Each row of $\mathcal{H}$ corresponds to a half-space $\mathcal{H}_i$,

$$\mathcal{H}_i \coloneqq \left\{x \in \mathbb{R}^p \;\middle|\; f_i^{\mathrm{T}} x \leq c_i\right\}, \tag{3}$$

where $i = 1, \ldots, n$ and f_i, $\|f_i\|_2 = 1$, is the normal vector of the plane $H_i \coloneqq \left\{x \in \mathbb{R}^p \;\middle|\; f_i^{\mathrm{T}} x = c_i\right\}$. The polyhedral set (2) can also be represented as the intersection of the half-spaces $\mathcal{H}_i$, i.e. $\mathcal{H} = \bigcap_{i=0}^{n-1} \mathcal{H}_i$. In many engineering applications, constraint sets can be approximated using (2).

While for some sets the solution of (2) can be obtained explicitly and in closed form, e.g., when $\mathcal{H}$ is an n-dimensional cube, no closed-form solution is known for arbitrary polyhedral sets. In these cases, the solution can be obtained using a solver for constrained quadratic programs. To solve (1), most solvers require introducing an additional variable $z \coloneqq Ax$ that is projected onto the set $\{z \in \mathbb{R}^n \mid z \leq c\}$, for which an explicit solution exists. However, for large n, this variable augmentation can reduce the computational efficiency, in particular when the projection is part of a larger iterative algorithm.

As an alternative to variable augmentation, problem (1) can be solved using *Dykstra's Alternating Projection Algorithm* [3]. Dykstra's algorithm, first published in 1983, extends von Neumann's *Method of Alternating Projections* (MAP) [8], which is designed to find a point lying in the intersection of n closed convex sets by cyclically projecting onto each individual set. While von Neumann's algorithm identifies *some* point in the intersection, Dykstra's algorithm determines the Euclidean projection (1). Both algorithms circumvent the potentially complex projection $\mathcal{P}_\mathcal{H}$ by iteratively applying the (known) projections $\mathcal{P}_{\mathcal{H}_0}, \ldots, \mathcal{P}_{\mathcal{H}_{n-1}}$.

Although Dykstra's algorithm is proven to eventually converge to the projection, it has shown to be prone to *stalling* [1] for certain sets, including polyhedral sets like (2). In such cases, the iterates of Dykstra's method remain unchanged for a number of iterations. The duration of the stalling period cannot be determined *a priori*, and depending on the starting point, can be arbitrary long. This phenomenon hinders the application of Dykstra's method in practice. Additionally, the algorithm *cannot* be run for a fixed number of iterations with a guarantee that the output will be closer to the projection than the initial point – a guarantee typically required when embedding Dykstra's method in iterative schemes, such as gradient methods [6].

*This research was supported by the Keble College Small Research Grant (KSRG118).

2 Dykstra's Alternating Projection

Given n convex sets $\mathcal{H}_0, \ldots, \mathcal{H}_{n-1}$, Dykstra's alternating projection algorithm [7] finds the orthogonal projection $x^\star$ of x° onto $\mathcal{H} \coloneqq \bigcap_{i=0}^{n-1} \mathcal{H}_i$ by generating a series of iterates $\{x_m\}$ using the scheme

$$x_{m+1} = \mathcal{P}_{\mathcal{H}_{[m]}}\left(x_m + e_{m-n}\right), \tag{4a}$$

$$e_m = e_{m-n} + x_m - x_{m+1}, \tag{4b}$$

where $[m] \coloneqq m \bmod n$, $x_0 = x$, $\mathcal{P}_{[m]} \coloneqq \mathcal{P}_{\mathcal{H}_{[m]}}$, and the auxiliary variables e_m are initialised as

$$e_{-n} = e_{-(n-1)} = \ldots = e_{-1} = 0. \tag{5}$$

Note that von Neumann's MAP can be obtained from (4) by setting $e_m \equiv 0 \quad \forall m$. The Boyle-Dykstra theorem [3] implies that $\lim_{m\to\infty} \|x_m - \mathcal{P}_{\mathcal{H}}(x)\| = 0$. For a finite number of iterations, there is no guarantee that $x_m \in \mathcal{H}$ nor that $x_m \neq x$.

2.1 The Polyhedral Case

For polyhedral sets (3), the projection step (4a) can be simplified to

$$x_{m+1} = \begin{cases} x_m + e_{m-n} & \text{if } x_m + e_{m-n} \in \mathcal{H}_{[m]} \\ x_m + e_{m-n} - \left((x_m + e_{m-n})^{\mathrm{T}} f_{[m]} - c_{[m]}\right) f_{[m]} & \text{if } x_m + e_{m-n} \notin \mathcal{H}_{[m]} \end{cases}, \tag{6}$$

and the update for the auxiliary vector to

$$e_m = \begin{cases} 0 & \text{if } x_m + e_{m-n} \in \mathcal{H}_{[m]}, \\ \left((x_m + e_{m-n})^{\mathrm{T}} f_{[m]} - c_{[m]}\right) f_{[m]} & \text{if } x_m + e_{m-n} \notin \mathcal{H}_{[m]}, \end{cases}. \tag{7}$$

The auxiliary vector e_m is either 0 or parallel to $f_{[m]}$, so that it can be represented as $e_m = k_m f_{[m]}$ with $k_m = \operatorname{dist}_{\mathcal{H}_{[m]}}(x_{m-1} + e_{m-n})$, further simplifying (11) to

$$k_m = k_{m-n} + x_m^{\mathrm{T}} f_{[m]} - c_{[m]}. \tag{8}$$

The convergence of Dykstra's iterates to the Eucledian projection has been analysed in [5, 4, 7] for polyhedral sets. The proof is based on partitioning the sets into inactive ($x^\star \notin H_i$) and active sets ($x^\star \in H_i$), i.e.

$$A = \{i \in \{0, \ldots, n-1\} \mid x_\infty \in H_i\}, \qquad B = \{0, \ldots, n-1\} \backslash A, \tag{9}$$

where $x_\infty = \lim_{m\to\infty} x_m$. It can be shown that there exists a number N_1 such that whenever

$$[m] \in B, \quad m \geq N_1 \quad \Rightarrow \quad x_m = x_{m-1}, \quad e_m = 0, \tag{10}$$

i.e. the half-spaces that become "inactive" remain inactive. Furthermore, there exists $N_2 \geq N_1$ such that whenever $n \geq N_2$, it holds that

$$\|x_{m+n} - x_\infty\|_2 \leq \alpha_{[m]} \|x_m - x_\infty\|_2, \tag{11}$$

where $0 \leq \alpha_{[m]} < 1$ are numbers related to angles between half-spaces. The number N_2 describes the iteration from which on the algorithm has determined the inactive half-spaces. Finally, it is shown that the iterates of the algorithm satisfy the following inequality:

Theorem 1 (Deutsch and Hundal [4])**.** *There exist constants $0 \leq c < 1$ and $\rho > 0$ such that*

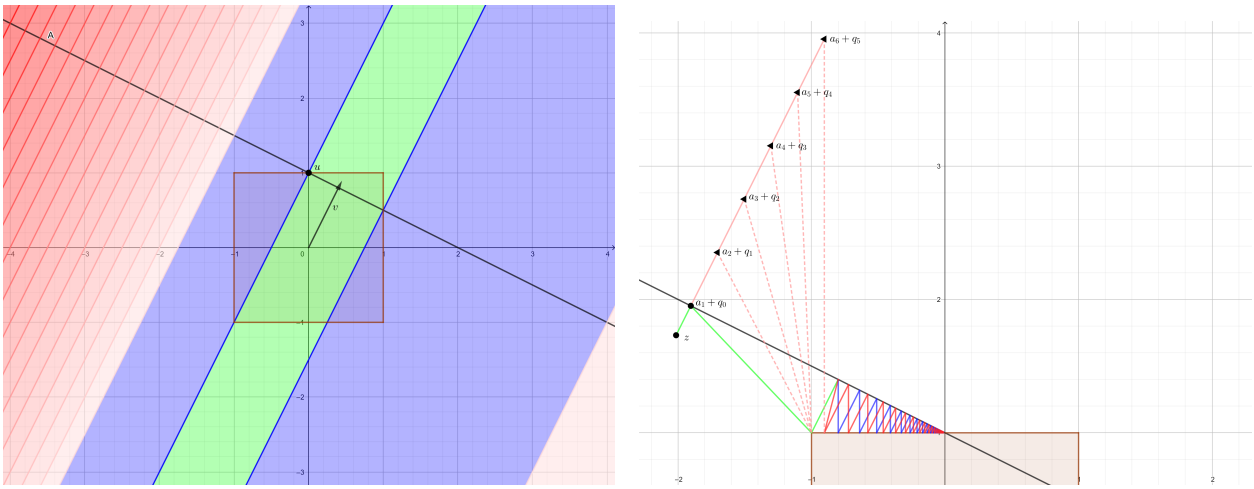
$$\|x_m - x_\infty\| \leq \rho c^m.$$

The factor c can be estimated from the smallest $\alpha_{[m]}$, which is characterized by the angle between certain subspaces (subspaces formed by the "active" halfspaces). The factor $\alpha_{[m]}$ can be upper-bounded by considering the "worst" angles in the polyhedron. The constant ρ, however, depends on an unknown iteration number $N_3 \geq N_2$ and on the starting point x°, and can therefore not be computed in advance [7, 9]. In the case of stalling, the variable ρ can become arbitrarily large, making the application of Dykstra's method difficult in practice. The authors of [7] proposed a combined Dysktra-conjugate-gradient method that allows for computing an upper bound on $\|x_m - x_\infty\|$. The authors of [9] proposed an alternative algorithm called *successive approximate algorithm*, which promises fast convergence, conditioned on knowing a point $x \in \mathcal{H}$ in advance.

2.2 Stalling

In [1], the behaviour of Dykstra's method is analysed for two sets. The authors give conditions on Dykstra's algorithm for (i) finite convergence, (ii) infinite convergence, and (iii) stalling followed by infinite convergence. A specific example is given for the case that the set is provided by the intersection of a line with a unit box in $\mathbb{R}^2$ ($\mathcal{H}$ is a polyhedron). It can be shown that cases (i)–(iii) depend on the starting point x_0, and one can determine the 3 regions shown in Figure 2a that yield different convergence behaviour. Convergence case (i) is obtained when starting in the green region, case (ii) when starting in the blue region, and case (iii) when starting in the red region.

To understand the stalling effect, consider Figure 2b, which shows the first iterations of Dykstra's algorithm with starting point in the red region. Note that the outcome of Dykstra's algorithm depends on the order of the sets $\mathcal{H}_i, \ldots, \mathcal{H}_n$. In Figure 2b, the algorithm starts by projecting onto the box and then onto the line. It can be seen that for the first 6 iterations[1], Dykstra's algorithm returns the top left corner of the box ("stalling"). The authors also determine the exact number of iterations required to break free from the red region, and show that if the starting point is arbitrarily far to the left, the algorithm will need an arbitrarily large iteration number to break free from the red region.



(a) Line-box example with different regions that yield different convergence properties.

(b) Stalling for the line-box example when x_0 is in the red region.

Figure 2: A demonstration of the stalling problem for a box and a line. Note how MAP applied to the same constraint sets would not result in any stalling: MAP follows the green line, and subsequently converges via the blue line path. Figure taken from [1].

[1] By one iteration we mean one cycle of n projections here.

3 Main Result: Fast Forwarding Stalling Period

By graphical examination of Figure 2 and formulae (6)-(8), the following observations can be made:

1. The stalling period continues until one of the active half-spaces becomes inactive. This is a necessary condition.
2. During stalling, it holds that $x_m \equiv x_{m-n}\ \forall m$, so that every iteration, a constant vector $\Delta_m := x_{m-1} - x_m$ is added to e_m.

These observations are formalised in the following theorem:

Theorem 2 (Length of Stalling Period)**.** *Suppose that at iteration m, the algorithm stalls, i.e. $x_{m+i} = x_{m-n+i}\ \forall i = 0, \dots, n-1$. Then, the length of the stalling period is given by*

$$N_{stall} := \arg\min_{i \in \mathcal{S}} \left\lfloor -k_{m-n+i} / \left(x_{m+i}^T f_{[m+i]} - c_{[m+i]} \right) \right\rfloor, \tag{12}$$

where $\mathcal{S} := \left\{ i \in \mathcal{A}_m \;\middle|\; x_{m-n+i}^T f_{[m+i]} - c_{[m+i]} < 0 \right\}$ and $\mathcal{A}_m := \left\{ i \in \{0, \dots, n-1\} \;\middle|\; x_{m+i} + e_{m-n} \notin \mathcal{H}_{[m+i]} \right\}$.

Proof. Note that the existence of a finite number N_{stall} is a consequence of the Boyle-Dykstra theorem. According to (6), the stalling period will terminate once one half-space becomes inactive, i.e. $x_{m+j} + e_{m-n+j} \notin \mathcal{H}_{[m-n+j]}$ for some $[m-n+j] \in \mathcal{A}_m$. According to (8), the only half-spaces that can become inactive are those for which $\delta k_i := x_{m+i}^{\mathrm{T}} f_{[m+i]} - c_{[m+i]} < 0$, which is, by definition, a constant during stalling. The length of the stalling period can therefore be obtained from choosing the smallest possible integer N for which

$$k_{m-n+i} + N \left(x_{m+i}^{\mathrm{T}} f_{[m+i]} - c_{[m+i]} \right) < 0, \qquad [m+i] \in \mathcal{A}_m,$$

which can be reformulated as in (12). □

By Theorem (2), the stalling period can be fast-forwarded by applying N_{stall} constant increments to each auxiliary variable k_m from (8). This is illustrated for the line-box example in Figure 3, where Figure 3a replicates the stalling situation from Figure 2 and the fast-forwarding is applied in Figure 3b. The first column illustrates the trajectories of the iterates and their distance to the Eucledian projection computed using an interior point method. The second column shows the half-space activity, where half-space 0 corresponds to the left side of the box, half-space 1 to the top side of the box, and half-space 2 to the line (down-facing normal). In Figure 3a, the algorithm stalls until iteration 16. From the second column, it can be seen that the algorithm stalls until half-space 0 becomes inactive.

In Figure 3b, stalling is detected at iteration 1, upon which the algorithm is fast-forwarded using $N_{stall} = 15$ iterations, where N_{stall} is computed using Theorem 2. At iteration 3, the iterates of the modified algorithm arrive where the iterates of the original algorithm arrive at iteration 16. The iterates of the modified algorithm then become identical to those of the original algorithm.

4 Additional Analysis: Control Perspective

Based on [7], Dykstra's method can be split into two phases:

(I) An initial phase during which half-spaces are successively discarded, which includes stalling phases.

(II) A second phase during which alternating projections eventually converge to the global projection.

Although the algorithm includes the non-linear projection operators onto each half-space, it can be simplified during both phases. Assume that at iteration m all half-spaces are active and remain active until iteration $m+n$. In this case, each projection is a linear operator and (4) (or (6)) becomes:

$$x_{m+1} = x_m + e_{m-n} - \left((x_m + e_{m-n})^{\mathrm{T}} f_{[m]} - c_{[m]} \right) f_{[m]}, \tag{13a}$$

$$e_m = \left((x_m + e_{m-n})^{\mathrm{T}} f_{[m]} - c_{[m]} \right) f_{[m]}, \tag{13b}$$

By considering that e_m is always parallel to $f_{[m]}$, i.e. $e_m = k_m f_{[m]}$ as in (8), we note that $((x_m + e_{m-n})^{\mathrm{T}} f_{[m]} - c_{[m]}) f_{[m]} = e_{m-n} + (x_m^{\mathrm{T}} f_{[m]} - c_{[m]}) f_{[m]}$. Moreover, it holds that $(x_m^{\mathrm{T}} f_{[m]}) f_{[m]} = f_{[m]} f_{[m]}^{\mathrm{T}} x_m$, and by defining

$$F_{[m]}^{\perp} \coloneqq I - f_{[m]} f_{[m]}^{\mathrm{T}}, \qquad \bar{b}_{[m]} \coloneqq c_{[m]} f_{[m]},$$

formulae (13) can be rewritten as

$$x_{m+1} = F_{[m]}^{\perp} x_m + \bar{b}_{[m]} =: \pi_{[m]}(x_m), \tag{14a}$$

$$e_m = e_{m-n} + (I - F_{[m]}^{\perp}) x_m - \bar{b}_{[m]} =: e_{m-n} + \tilde{\pi}_{[m]}(x_m), \tag{14b}$$

where $\tilde{\pi}_{[m]}(x_m) = x_m - \pi_{[m]}(x_m)$. Equations (14) hold as long as $x_m + e_{m-n} \notin \mathcal{H}_{[m]} \forall m$, or equivalently

$$e_{m-n}^{\mathrm{T}} f_{[m]} + x_m^{\mathrm{T}} f_{[m]} - c_{[m]} > 0, \quad \forall m. \tag{15}$$

For the following analysis, change the indexing in (13) for $[m] = 1, \ldots, n-1$ as

$$x_{m+1} \mapsto x_{k+1}^{[m]}, \qquad x_m \mapsto x_k^{[m-1]}, \qquad e_{m-n} \mapsto e_k^{[m]}, \qquad e_m \mapsto e_{k+1}^{[m]}, \tag{16}$$

and for $[m] = 0$ replace $x_m \mapsto x_k^{[m-1]}$ in (16) by $x_m \mapsto x_{k-1}^{[m-1]}$ so that (14a) becomes

$$x_{k+1}^{m} = \begin{cases} \pi_m(x_{k-1}^{n-1}) & \text{for } m = 0, \\ \pi_m(x_k^{m-1}) & \text{otherwise} \end{cases}, \qquad e_{k+1}^{m} = e_{m-n} + \begin{cases} \tilde{\pi}_m(x_{k-1}^{n-1}) & \text{for } [m] = 0, \\ \tilde{\pi}_m(x_k^{m-1}) & \text{otherwise,} \end{cases} \tag{17}$$

where now m is assumed to be restricted to the range $m = 0, \ldots, n-1$ and $k = 0, 1, 2, \ldots$. Using the notation $\pi_i \circ \pi_j(x) = \pi_i(\pi_j(x))$, we expand (17) as

$$\begin{aligned} x_{k+1}^{0} &= \pi_0(x_k^{n-1}), \\ x_{k+1}^{1} &= \pi_1 \circ \pi_0(x_k^{n-1}), \\ &\vdots \\ x_{k+1}^{n-1} &= \pi_{n-1} \circ \pi_{n-2} \circ \cdots \circ \pi_0(x_k^{n-1}), \end{aligned} \tag{18}$$

and (14b)

$$\begin{aligned} e_{k+1}^{0} &= e_k^{0} + \tilde{\pi}_0(x_k^{n-1}), \\ e_{k+1}^{1} &= e_k^{1} + \tilde{\pi}_1 \circ \pi_0(x_k^{n-1}), \\ &\vdots \\ e_{k+1}^{n-1} &= e_k^{n-1} + \pi_{n-1} \circ \pi_{n-2} \circ \cdots \circ \pi_0(x_k^{n-1}). \end{aligned} \tag{19}$$

Replacing x_k^{n-1} in (32) and (33) by $x_k^{n-1} = \pi_{n-1} \circ \cdots \circ \pi_{j-1}(x_k^j)$ yields

$$x_{k+1}^{i} = \mathop{\circ}_{p=0}^{n-1} \pi_{[i-p]}(x_k^i), \tag{20a}$$

$$e_{k+1}^{i} = e_k^i + \tilde{\pi}_i \circ \left(\mathop{\circ}_{p=1}^{n-1} \pi_{[i-p]}(x_k^i) \right) = e_k^i + \mathop{\circ}_{p=1}^{n-1} \pi_{[i-p]}(x_k^i) - \mathop{\circ}_{p=0}^{n-1} \pi_{[i-p]}(x_k^i), \tag{20b}$$

where $\mathop{\circ}_{p=0}^{n-1} \pi_{[i-p]}(x) = \pi_i \circ \cdots \circ \pi_0 \circ \pi_{n-1} \circ \cdots \circ \pi_{i+1}(x)$. Substituting matrix notation for $\pi_{[i-p]}$ yields

$$x_{k+1}^{i} = \left(\prod_{p=0}^{n-1} F_{[i-p]}^{\perp} \right) x_k^i + \sum_{p=0}^{n-1} \left(\prod_{l=0}^{n-2-p} F_{[i-l]}^{\perp} \right) \bar{b}_{[i-(n-1)+p]}, \tag{21a}$$

$$e_{k+1}^{i} = e_k^i + (I - F_i^{\perp}) \left(\prod_{p=1}^{n-1} F_{[i-p]}^{\perp} \right) x_k^i + (I - F_i^{\perp}) \sum_{p=1}^{n-1} \left(\prod_{l=0}^{n-2-p} F_{[i-l]}^{\perp} \right) \bar{b}_{[i-(n-1)+p]} - \bar{b}_i. \tag{21b}$$

and after gathering the matrices and constant vectors in $A_{x,i}$, $b_{x,i}$, $A_{e,i}$, and $b_{e,i}$:

$$x^i_{k+1} = A_{x,i}x^i_k + b_{x,i}, \qquad e^i_{k+1} = e^i_k + A_{e,i}x^i_k + b_{e,i}. \tag{22a}$$

This is a standard LTI system with constant inputs that can be rewritten in explicit form as:

$$x^i_k = A^k_{x,i}x^i_0 + \sum_{p=0}^{k-1} A^p_{x,i}b_{x,i}, \qquad e^i_k = e^i_0 + \sum_{p=0}^{k-1}(A_{e,i}x^i_p + b_{e,i}). \tag{23a}$$

We note that $A_{e,i} = \tilde{A}_i - A_{x,i}$ and $b_{e,i} = \tilde{b}_i - b_{x,i}$, where

$$\tilde{A}_i = \prod_{p=1}^{n-1} F^{\perp}_{[i-p]}, \qquad \tilde{b}_i = \sum_{p=1}^{n-1}\left(\prod_{l=0}^{n-2-p} F^{\perp}_{[i-l]}\right).$$

Although the pair (x^i_k, e^i_k) evolves independently of (x^j_k, e^j_k), $j \neq i$, the half spaces are coupled through (15), which in the new notation reads as

$$\begin{cases} (e^m_k)^{\mathrm{T}} f_m + (x^{n-1}_k)^{\mathrm{T}} f_m - c_m > 0 & \text{for} \quad m = 0, \\ (e^m_k)^{\mathrm{T}} f_m + (x^{m-1}_{k+1})^{\mathrm{T}} f_m - c_m > 0 & \text{otherwise.} \end{cases} \tag{24}$$

For example, for $i = n-1$, the matrices and vectors obtained as

$$
\begin{aligned}
A_{x,n-1} &= F^{\perp}_{n-1}F^{\perp}_{n-2}\dots F^{\perp}_{0},\\
b_{x,n-1} &= F^{\perp}_{n-1}F^{\perp}_{n-2}\dots F^{\perp}_{1}c_0f_0 + F^{\perp}_{n-1}F^{\perp}_{n-2}\dots F^{\perp}_{2}c_1f_1 + \dots + F^{\perp}_{n-1}c_{n-2}f_{n-2} + c_{n-1}f_{n-1},\\
A_{e,n-1} &= (I - F^{\perp}_{n-1})F^{\perp}_{n-2}\dots F^{\perp}_{0},\\
b_{e,n-1} &= (I - F^{\perp}_{n-1})F^{\perp}_{n-2}\dots F^{\perp}_{1}c_0f_0 + (I - F^{\perp}_{n-1})F^{\perp}_{n-2}\dots F^{\perp}_{2}c_1f_1 + \dots + (I - F^{\perp}_{n-1})c_{n-2}f_{n-2} - c_{n-1}f_{n-1}.
\end{aligned}
$$

For the remaining i, the matrices have similar structure but with products taken over cyclically permuted indices. Recalling that $F^{\perp}_{n-1} = I - f_{n-1}f^{\mathrm{T}}_{n-1}$, we note from setting $x^i_k = x^{i,\perp}_k + x^{i,\parallel}_k$ with $x^{i,\parallel}_k$ parallel to f_i:

$$
\begin{aligned}
x^{i,\perp}_{k+1} + x^{i,\parallel}_{k+1} &= A_{x,i}(x^{i,\perp}_k + x^{i,\parallel}_k) + \underbrace{b_{x,i}}_{b^{\perp}_{x,i}+b^{\parallel}_{x,i}},\\
&= A_{x,i}(x^{i,\perp}_k + x^{i,\parallel}_k) + b^{\perp}_{x,i} + b^{\parallel}_{x,i},
\end{aligned}
$$

where $b^{\parallel}_{x,i} = c_i f_i$. Considering that $A_{x,i}x \perp f_i$, we see that

$$
x^{i,\parallel}_k = b^{\parallel}_{x,i}, \qquad\qquad x^{i,\perp}_{k+1} = A_{x,i}x^{i,\perp}_k + b^{\perp}_{x,i} + A_{x,i}b^{\parallel}_{x,i}.
$$

In a possible steady state,

$$
(I - A_{x,i})x^{i,\perp}_{ss} = b^{\perp}_{x,i} + A_{x,i}b^{\parallel}_{x,i}.
$$

$$x^1_{k+1} = \pi_1(\pi_3(\pi_2(x^1_k))), \tag{25a}$$
$$x^2_{k+1} = \pi_2(\pi_1(\pi_3(x^2_k))), \tag{25b}$$
$$x^3_{k+1} = \pi_3(\pi_2(\pi_1(x^3_k))), \tag{25c}$$
$$e^1_{k+1} = \tilde{\pi}_1(\pi_3(\pi_2(x^1_k)) + e^1_k), \tag{25d}$$
$$e^2_{k+1} = \tilde{\pi}_2(\pi_1(\pi_3(x^2_k)) + e^2_k), \tag{25e}$$
$$e^3_{k+1} = \tilde{\pi}_3(\pi_2(\pi_1(x^3_k)) + e^3_k). \tag{25f}$$

Clearly, since π_i and $\tilde{\pi}_i$ are *linear* operators, (34) is of the form $\mathbf{x}_{k+1} = A\mathbf{x}_k$. Moreover, it can be seen that the pair (x^i_k, e^i_k) evolves independently of (x^j_k, e^j_k), $j \neq i$. However, the half spaces are coupled through (15), which in the new notation reads as

$$\begin{cases} (e^m_k)^{\mathrm{T}} f_m + (x^{m-1}_{k+1})^{\mathrm{T}} f_m - c_m > 0 & \text{for} \quad m = 1, \dots, n-1, \\ (e^m_k)^{\mathrm{T}} f_m + (x^{n-1}_k)^{\mathrm{T}} f_m - c_m > 0 & \text{for} \quad m = 0. \end{cases} \tag{26}$$

We compute $\pi_p(\pi_j(\pi_i(x)))$ as

$$\begin{aligned} \pi_p(\pi_j(\pi_i(x))) &= \pi_p(\pi_j(x - (\alpha_{x,i} - c_i) f_i))) \\ &= \pi_p(x - (\alpha_{x,i} - c_i) f_i - (\alpha_{x,j} - (\alpha_{x,i} - c_i)\alpha_{i,j} - c_j) f_j) \\ &= x - (\alpha_{x,i} - c_i) f_i - (\alpha_{x,j} - (\alpha_{x,i} - c_i)\alpha_{x,j} - c_j) f_j \\ &\quad - (\alpha_{x,p} - (\alpha_{x,i} - c_i)\alpha_{i,p} - (\alpha_{x,j} - (\alpha_{x,i} - c_i)\alpha_{i,j} - c_j)\alpha_{j,p} - c_p) f_p =: A_p x + b_p, \end{aligned} \tag{27}$$

where $\alpha_{i,j} \coloneqq f_i^{\mathrm{T}} f_j$ and

$$A_p \coloneqq I - f_i f_i^{\mathrm{T}} - f_j f_j^{\mathrm{T}} - f_p f_p^{\mathrm{T}} + \alpha_{i,j} f_j f_i^{\mathrm{T}} + (\alpha_{i,p} - \alpha_{i,j}\alpha_{j,p}) f_p f_i^{\mathrm{T}} + \alpha_{j,p} f_p f_j^{\mathrm{T}}, \tag{28a}$$
$$b_p \coloneqq c_i f_i + (c_j - c_i \alpha_{i,j}) f_j + (c_p - c_i(\alpha_{i,p} - \alpha_{i,j}\alpha_{j,p}) - c_j \alpha_{j,p}) f_p. \tag{28b}$$

Note that $A_p = A_{p \circ j \circ i}$, i.e. it depends on the order of j and i. Similarly, $\tilde{\pi}_p(\pi_j(\pi_i(x)) + e^p)$ is computed as

$$\begin{aligned} \tilde{\pi}_p(\pi_j(\pi_i(x)) + e^p) &= e^p + \tilde{\pi}_p(\pi_j(\pi_i(x))), \\ &= e^p + x - \pi_p(\pi_j(\pi_i(x))), \\ &= e^p + (I - A_p x - b_p. \end{aligned} \tag{29}$$

With these definitions, (34) can be rewritten as

$$x^p_{k+1} = A_p x^p_k + b_p, \tag{30a}$$
$$e^p_{k+1} = (I - A_p) x^p_k - b_p, \tag{30b}$$

where $p \in \{1, 2, 3\}$

4.1 OLD NOTATION

Algorithm (14) can be interpreted as a *linear, time-invariant dynamical system.* To see this, abbreviate (14a) and (14b) as $x_{m+1} \coloneqq \pi_{[m]}(x_m + e_{m-n})$ and $e_m \coloneqq \tilde{\pi}_{[m]}(x_m + e_{m-n})$, respectively, and change the indexing as:

$$x_{m+1} \mapsto x_{k+1}^{[m]}, \qquad x_m \mapsto x_k^{[m-1]}, \qquad e_{m-n} \mapsto e_k^{[m]}, \qquad e_m \mapsto e_{k+1}^{[m]}. \tag{31}$$

For simplicity, suppose that there are three half-spaces with iterates x_k^1, x_k^2, and x_k^3, and auxiliary variables e_k^1, e_k^2, and e_k^3. This allows to rewrite (14a) as

$$x_{k+1}^1 = \pi_1(x_k^3 + e_k^1), \tag{32a}$$
$$x_{k+1}^2 = \pi_2(\pi_1(x_k^3 + e_k^1) + e_k^2), \tag{32b}$$
$$x_{k+1}^3 = \pi_3(\pi_2(\pi_1(x_k^3 + e_k^1) + e_k^2) + e_k^3), \tag{32c}$$

and (14b) as

$$e_{k+1}^1 = \tilde{\pi}_1(x_k^3 + e_k^1), \tag{33a}$$
$$e_{k+1}^2 = \tilde{\pi}_2(\pi_1(x_k^3 + e_k^1) + e_k^2), \tag{33b}$$
$$e_{k+1}^3 = \tilde{\pi}_3(\pi_2(\pi_1(x_k^3 + e_k^1) + e_k^2) + e_k^3). \tag{33c}$$

Equations (32) and (33) can be further manipulated by substituting $x_k^3 = \pi_3(\pi_2(x_k^1 + e_k^2) + e_k^3)$ for half-space 1 and and $x_k^3 = \pi_3(x_k^2 + e_k^3)$ for half-space 2:

$$x_{k+1}^1 = \pi_1(\pi_3(\pi_2(x_k^1 + e_k^2) + e_k^3) + e_k^1), \tag{34a}$$
$$x_{k+1}^2 = \pi_2(\pi_1(\pi_3(x_k^2 + e_k^3) + e_k^1) + e_k^2), \tag{34b}$$
$$x_{k+1}^3 = \pi_3(\pi_2(\pi_1(x_k^3 + e_k^1) + e_k^2) + e_k^3), \tag{34c}$$
$$e_{k+1}^1 = \tilde{\pi}_1(\pi_3(\pi_2(x_k^1 + e_k^2) + e_k^3) + e_k^1), \tag{34d}$$
$$e_{k+1}^2 = \tilde{\pi}_2(\pi_1(\pi_3(x_k^2 + e_k^3) + e_k^1) + e_k^2), \tag{34e}$$
$$e_{k+1}^3 = \tilde{\pi}_3(\pi_2(\pi_1(x_k^3 + e_k^1) + e_k^2) + e_k^3). \tag{34f}$$

Clearly, since π_i and $\tilde{\pi}_i$ are *linear* operators, (34) is of the form $\mathbf{x}_{k+1} = A\mathbf{x}_k$, where

$$\mathbf{x}_{k+1} \coloneqq \begin{pmatrix} x_{k+1}^1 \\ x_{k+1}^2 \\ x_{k+1}^3 \\ e_{k+1}^1 \\ e_{k+1}^2 \\ e_{k+1}^3 \end{pmatrix}.$$

References

[1] H. H. Bauschke, R. S. Burachik, D. B. Herman, and C. Y. Kaya. On Dykstras algorithm: finite convergence, stalling, and the method of alternating projections. *ArXiv e-prints*, pages 1–14, January 2020.

[2] Heinz Bauschke and Patrick Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Space*. Springer, January 2011.

[3] 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 Inference*, pages 28–47. Springer New York, 1986.

[4] F. Deutsch and H. Hundal. The rate of convergence of Dykstra's cyclic projections algorithm: The polyhedral case. *Numerical Functional Analysis and Optimization*, 15(6):537–565, May 1994.

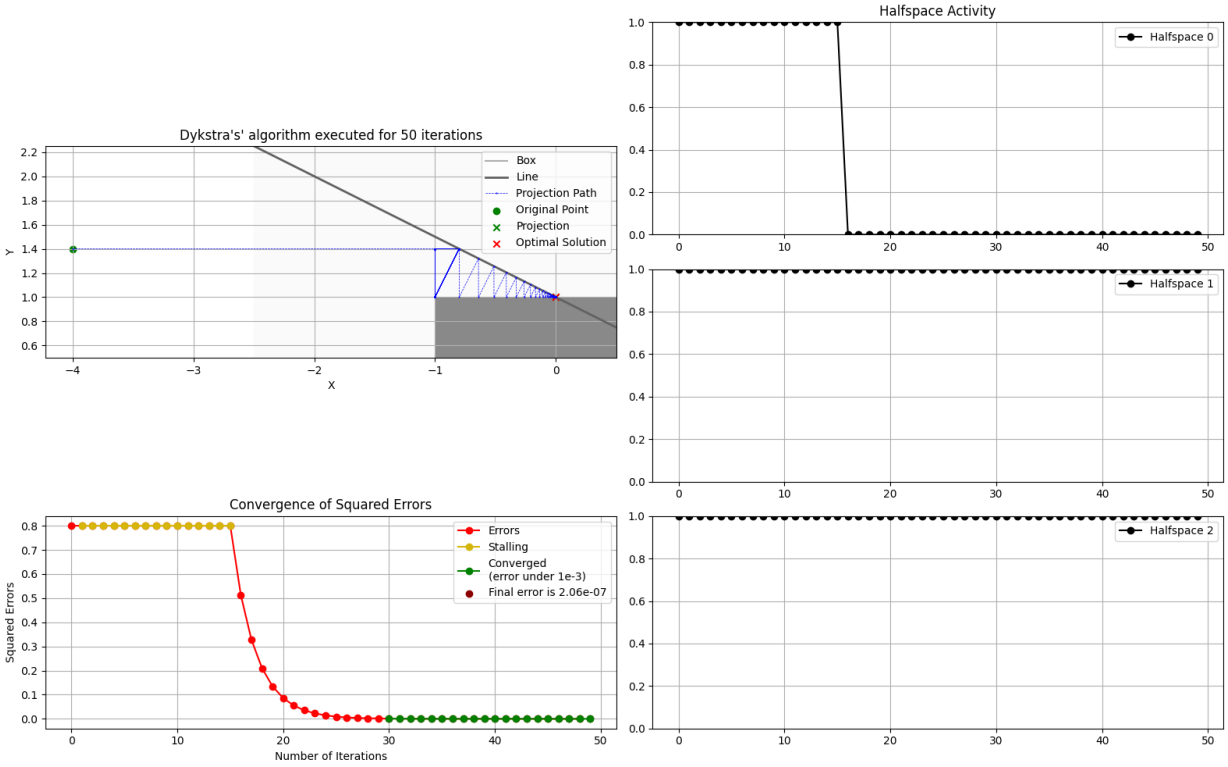
[5] A. N. Iusem and A. R. de Pierro. On the convergence properties of Hildreth's quadratic programming algorithm. *Math. Prog. Ser. A and B*, 47(1):37–51, May 1990.

[6] A. Patrascu and I. Necoara. On the convergence of inexact projection primal first-order methods for convex minimization. *IEEE Trans. on Automat. Cont.*, 63(10):3317–3329, Oct. 2018.

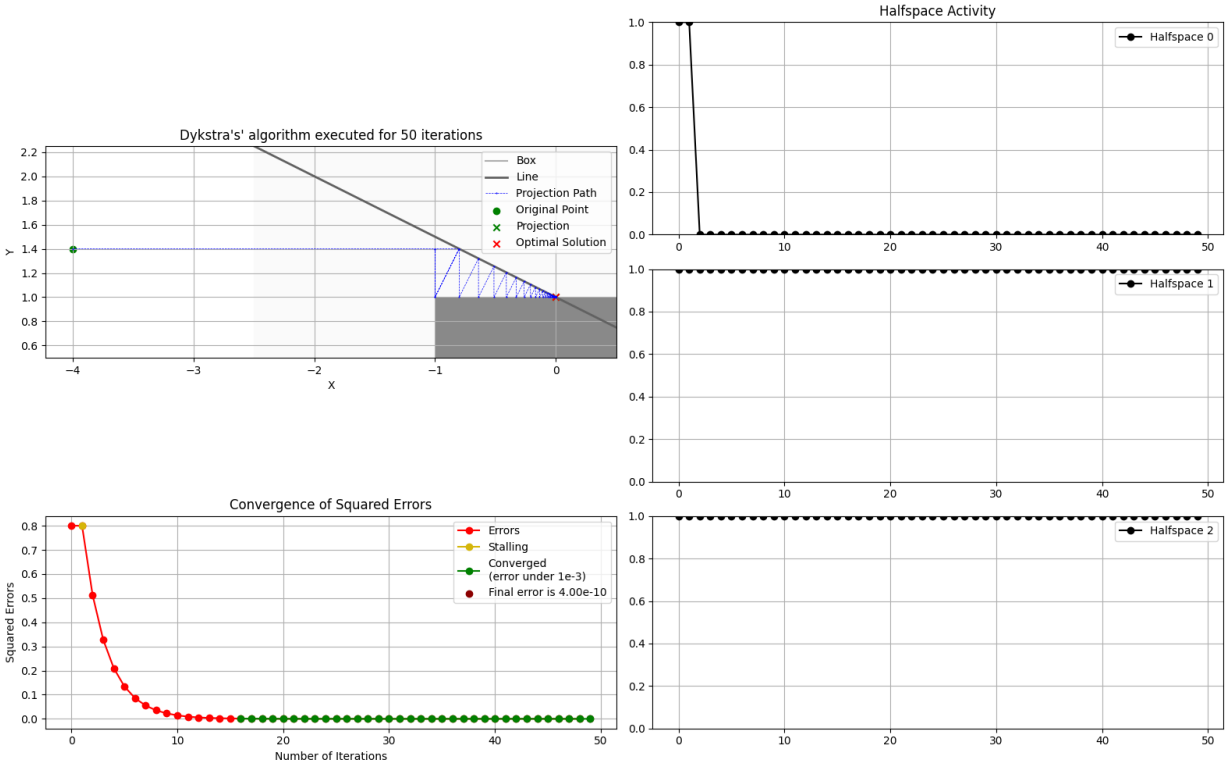
[7] C. Perkins. A convergence analysis of Dykstra's algorithm for polyhedral sets. *SIAM J. Numer. Anal.*, 40(2):732–804, July 2002.

[8] J. Von Neumann. *Functional Operators: The Geometry of Orthogonal Spaces*. Number Bd. 2 in Annals of Mathematics Studies. Princeton University Press, 1951.

[9] S. Xu. Successive approximate algorithm for best approximation from a polyhedron. *J. Approx. Theory*, 93(3):415–433, June 1998.



(a) Replication of the stalling situation from Figure 2.



(b) Modified algorithm with fast-forwarding.

Figure 3: Dykstra's algorithm applied to the line-box example from Figure 2. The first column illustrates the trajectories of the iterates and their distance to the Eucledian projection computed using an interior point method. The second column shows the half-space activity, where half-space 0 corresponds to the left side of the box, half-space 1 to the top side of the box, and half-space 0 to the line (down-facing normal).