Dykstra's Algorithm: Stalling Resolution, Algorithmic Enhancements, and Applications in Optimal Transport

Claudio Vestini, '26

Submitted to the
Department of Mechanical and Aerospace Engineering
Princeton University
in partial fulfillment of the requirements of
Undergraduate Independent Work.

Final Report

April 23, 2026

Ryne Beeson,
Bartolomeo Stellato (ORFE)
Luigi Martinelli
MAE 499
89 pages
File Copy

This thesis represents my own work in accordance with University regulations.

Abstract

Euclidean projections onto intersections of polyhedral sets are essential operations in constrained optimisation. Dykstra's algorithm provides an efficient iterative method for computing these projections; however, its practical utility is limited by a stalling phenomenon that can result in arbitrarily long execution cycles. The primary theoretical contribution of this thesis is the formalisation of the stalling condition and the derivation of its duration in closed form. Building on this mathematical resolution, a fast-forward modification is introduced that detects stalling and advances the internal memory variables beyond the stalling cycle in a single computational step. This modification eliminates the algorithm's unpredictable execution time while preserving its asymptotic convergence guarantees.

An accelerated version of the stall-averse solver is integrated within a large-scale, inexact projected gradient descent framework to address density estimation tasks via optimal transport. Specifically, we approximate Knothe-Rosenblatt triangular maps parameterised by a probabilist's Hermite polynomial basis. Empirical estimation of these maps necessitates enforcing monotonicity over a discrete sample ensemble, resulting in an ill-conditioned polyhedral feasible set. The modified Dykstra algorithm is consequently employed to compute the required Euclidean projections onto this geometric structure.

The combined architecture is evaluated across two nonlinear tracking domains. In astrodynamics, the framework is applied to non-Gaussian uncertainty propagation through orbital dynamics, reconstructing physical ground-truth shears of satellite state distributions where classical filters are inadequate. In geophysical data assimilation, the engine is used with the Lorenz 1963 system to approximate continuous mappings of chaotic probability densities. These results collectively demonstrate the robustness and scalability of the framework for mapping uncertainty distributions.

Acknowledgements

I would like to express my gratitude to my advisers, Professor Ryne Beeson and Professor Bartolomeo Stellato, for their guidance throughout this thesis. Their expertise was instrumental in bridging the theoretical development aspect of this thesis with its practical applications. I am thankful to the members of their respective research groups for their academic support and collaborative environment. I extend my appreciation to my reader, Professor Luigi Martinelli, for his time and review of this manuscript.

Ai miei genitori, Renato e Francesca, che mi hanno fortemente sorretto e sostenuto attraverso il mio percorso accademico.

Contents

Abstract iii
Acknowledgements iv
List of Tables viii
List of Figures ix
List of Symbols xi

1 Introduction **1**
1.1 The landscape of space situational awareness 1
1.2 The breakdown of Gaussian assumptions 2
1.3 Optimal transport and triangular maps 4
1.4 Computational challenges 4
1.5 Dykstra's algorithm and stalling 5
1.6 Thesis outline 6

2 Fast-forwarding stalling in Dykstra's algorithm **8**
2.1 Background on Euclidean projections 9
2.2 Dykstra's projection algorithm 10
2.3 The stalling phenomenon 14
2.4 A closed-form solution for the stalling period 17
2.5 The fast-forward modified algorithm 19
2.6 Numerical experiments on stalling resolution 21
2.6.1 The line-box example 21
2.6.2 Benchmarking on randomly generated constraints 23
2.7 Summary 26

3 Optimal transport **27**
3.1 The Knothe-Rosenblatt rearrangement 28
3.2 Hermite polynomial parameterisation 31
3.2.1 Total degree truncation 32

3.2.2 Approximating the map . . . 33
3.3 Constrained optimisation formulation . . . 34
3.4 Summary . . . 35
3.4.1 Connection with Dykstra's algorithm . . . 35

4 A highly scalable inexact projection framework 36
4.1 Outer loop projected gradient descent . . . 37
4.2 Stochastic gradient descent and minibatching . . . 39
4.3 Iterative hard thresholding . . . 40
4.4 Inner loop integration . . . 42
4.4.1 Dynamic iteration scheduling . . . 42
4.4.2 Inactive constraint removal . . . 44
4.4.3 Feasibility detection and early termination . . . 44
4.5 Summary . . . 46

5 Experiments and applications 47
5.1 Reconstructing synthetic distributions . . . 48
5.1.1 Numerical results . . . 49
5.2 Application to uncertainty propagation . . . 56
5.2.1 Parameterisation of orbital state . . . 56
5.2.2 Filtering . . . 58
5.2.3 Mapping Gauss von Mises distributions . . . 60
5.2.4 Numerical results . . . 61
5.3 Application to geophysics . . . 65
5.3.1 Lorenz 63 model . . . 65
5.3.2 Numerical results . . . 67
5.4 Performance scaling experiment . . . 69
5.4.1 Experimental setup . . . 70
5.4.2 Numerical results . . . 70

6 Conclusion and future work 73
6.1 Summary of research contributions . . . 73
6.2 Future work . . . 74

List of Tables

2.1 Summary of the polyhedral environments used for the benchmark. . . 24
2.2 Runtime and final error across the benchmark environments. 24

5.1 Optimisation parameters used across the synthetic shear experiments. 50
5.2 Runtime across the synthetic tests. 52
5.3 Distribution parameters for the orbital uncertainty experiments. . . . 61
5.4 Parameters for the orbital uncertainty experiments. 62
5.5 Computational runtime for the orbital uncertainty experiments. 64
5.6 Parameters for the L63 experiments. 67
5.7 Runtimes for the L63 experiments. 69

List of Figures

1.1 Evolution of equinoctial elements across propagation epochs [1]. . . . 3

2.1 The method of alternating projections yielding a sub-optimal solution. 11
2.2 Dykstra's algorithm circumventing premature termination. 12
2.3 Stalling regions for the line-box example as defined in [5]. 14
2.4 A demonstration of the stalling phenomenon for the line-box example. 15
2.5 Stalling break-through for the line-box example. 16
2.6 Dykstra's method and Algorithm 1 applied to the line-box example. . 22
2.7 Benchmark of Algorithm 1 versus Dykstra across varying geometries. 25

3.1 A schematic illustrating optimal transport via a deterministic map S. 28
3.2 The first five Hermite polynomials evaluated over the real line. 31

4.1 Inverse time decay schedule for the unconstrained learning rate. 38
4.2 Dynamic inner Dykstra budget over the outer learning iterations. . . . 43

5.1 Reconstruction of the three synthetic shear distributions. 51
5.2 Progress plot for the linear experiment (`SEED` = 2222). 53
5.3 Progress plot for the quadratic experiment (`SEED` = 5432). 54
5.4 Progress plot for the cubic experiment (`SEED` = 507). 55
5.5 Geometric representation of the classical Keplerian orbital elements. . 57
5.6 Results for the three GVM orbital uncertainty experiments. 63
5.7 Lorenz 1963 prior and posterior samples. Courtesy of Prof. Beeson [25]. 66
5.8 Optimal transport results for the L63 system. 68
5.9 Runtime scaling comparison: fast-forward Dykstra versus SciPy SLSQP. 70
5.10 Runtime scaling comparison: fast-forward Dykstra versus OSQP. . . . 71

List of Symbols

MAE	Mechanical and Aerospace Engineering	1
ORFE	Operations Research and Financial Engineering	1
PGD	Projected gradient descent	1
SGD	Stochastic gradient descent	1
KR	Knothe-Rosenblatt	1
IHT	Iterative hard thresholding	1
EKF	Extended Kalman filter	1
UKF	Unscented Kalman filter	1
GVM	Gauss Von Mises	1
$\mathbb{R}$	The set of real numbers	1
$\mathbb{N}$	The set of natural numbers	1
$\mathbb{N}_0$	The set of nonnegative integers	1
$\mathbb{S}^1$	The unit circle	1
d	Optimisation state dimension	1
n	Total number of half-spaces/particles	1
$\mathcal{H}$	Polyhedral feasible set formed by the intersection of half-spaces	1
A	Constraint matrix	1
a_i	Unit normal vector defining the boundary of the i-th half-space	1
b_i	Boundary offset parameter of the i-th half-space	1
w	Decision variable	1
w°	Initial unconstrained point prior to projection	1
$w^\star$	True Euclidean projection onto the feasible set	1
m	Inner Dykstra projection cycle index	1
t	Outer projected gradient descent learning loop index	1
$w^{(t)}$	Decision variable at outer iteration t	1
$w^{(m)}$	Decision variable at inner Dykstra iteration m	1
$w^{(t,m)}$	Decision variable at inner iteration m and outer iteration t	1
$e^{(m)}$	Auxiliary Dykstra variable at inner iteration m	1

$k^{(m)}$	Auxiliary Dykstra scalar at inner iteration m	1
$M^{(t)}$	Dynamic inner Dykstra iteration budget ceiling	1
M_{base}	Base inner Dykstra iteration budget	1
p	Base inner Dykstra iteration budget scaling exponent	1
ε_m	Monotonicity tolerance scalar	1
ϵ_m	Monotonicity tolerance vector	1
N_{stall}	Closed-form solution for stalling period duration	1
m_{stall}	Iteration index denoting the onset of the stalling phenomenon	1
N_{skip}	Fast-forward algorithmic skip duration	1
$\mathcal{B}^{(t)}$	Subset of indices for stochastic gradient descent	1
B	Stochastic gradient descent cardinality	1
$g^{(t)}$	Stochastic gradient	1
$\eta^{(t)}$	Projected gradient descent learning rate	1
η°	Projected gradient descent base learning rate	1
γ	Learning rate decay parameter	1
$\mathcal{M}^{(t)}$	Boolean active mask vector for iterative hard thresholding	1
ϵ_p	Tolerance scalar for iterative hard thresholding	1
ϵ_p	Tolerance vector for iterative hard thresholding	1
K	Application interval for iterative hard thresholding	1
T	Total outer projected gradient descent iterations	1
D	Physical state dimension of particles	1
P	Maximum coefficient of the map's polynomial	1
J_k	Maximum coefficient of the truncated polynomial, minus one	1
$\alpha^{(j)}$	Multi-index governing the j-th multivariate basis function	1
$\alpha_m^{(j)}$	The m-th spatial component of the multi-index $\alpha^{(j)}$	1
x	Unmapped particles	1
z	Mapped particles	1
$\tilde{z}$	True multivariate standard normal particles	1
S	Optimal transport map	1
S^k	The k-th scalar component of the optimal transport map	1
$\tilde{S}$	Artificial shear map	1
$\mathcal{S}_k$	Function hypothesis space for scalar map components	1
ℓ_2	Euclidean distance norm	1

Chapter 1

Introduction

1.1 The landscape of space situational awareness

Over the past decade, the near-Earth space environment has undergone significant densification. The proliferation of large satellite constellations and the increasing accumulation of orbital debris have transformed space situational awareness from a routine monitoring task into a critical, high-frequency computational challenge. Concurrently, geopolitical and scientific interests have expanded into the cislunar domain, a vast and complex region encompassing the Earth, the Moon, and the Lagrange points where their gravitational forces are balanced.

International initiatives such as the Artemis programme and the Lunar Gateway are positioning the cislunar environment as the next major frontier for human activity and autonomous satellite operations. These emerging regimes necessitate a fundamental paradigm shift in the tracking of resident space objects. In near-Earth orbit, operators must rigorously monitor the evolution of probability density functions over time to accurately assess collision probabilities. In the cislunar environment, continuous multi-body gravitational interactions result in highly nonlinear dynamics, with trajectories exhibiting extreme sensitivity to initial conditions.

As a result, deterministic tracking of satellites, which treats them as single points in space, is no longer sufficient. To maintain operational safety, prevent collisions, and ensure mission viability across all orbital regimes, the field is transitioning from tracking singular points to propagating entire uncertainty clouds.

1.2 The breakdown of Gaussian assumptions

The state of an object in orbit is most commonly parameterised using a set of coordinates known as the *equinoctial* elements, which are six-dimensional. This coordinate system resolves the mathematical singularities inherent in classical *Keplerian* elements. However, propagating probability distributions through the highly nonlinear dynamics of orbital mechanics poses significant analytical challenges. While the initial uncertainty in a satellite's state immediately after an observation might be well-approximated by a multivariate Gaussian distribution, the true physical distribution rapidly distorts as the state is propagated forward in time.

The primary driver of this distortion is the physical ground truth shear of the orbital environment. Differential gravitational effects, atmospheric drag, and solar radiation pressure collectively induce nonlinear shearing on the state space. As the uncertainty cloud is pushed through these dynamics, it stretches, folds, and develops heavy tails, frequently exhibiting distinct "boomerang" or highly coupled geometries [1].

Classical filtering techniques, including the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF), are fundamentally ill-equipped to handle these dynamics over extended time horizons. These methods assume that the underlying probability distribution remains approximately Gaussian. Constraining propagated uncertainty within a linearised Gaussian covariance ellipsoid mischaracterises the underlying nonlinear dynamical flow. Imposing this rigid geometry on a highly sheared physical distribution often results in classical filters placing their maximum likelihood estimate entirely outside the support of the true state 1.1. As a result, the filter may assign peak probability density to regions of the state space where the actual physical measure is zero.

This phenomenon is illustrated in Figure 1.1, where a particle cloud approximating a set of Low-Earth equinoctial orbital elements is forward-propagated over several orbital periods. Three probability distributions are fitted to these data: two Gaussians derived from the EKF and UKF, and a Gauss Von Mises (GVM) distribution, which more accurately captures the nonlinear shearing of the orbital elements (see Chapter 5 for further details). A two-dimensional slice along the semi-major axis and mean longitude plane is plotted with each probability contour. Figure 1.1 demonstrates that the GVM distribution significantly outperforms the EKF and UKF over long-term propagation. This divergence between mathematical assumptions and physical reality necessitates an alternative approach to uncertainty modelling that can accurately

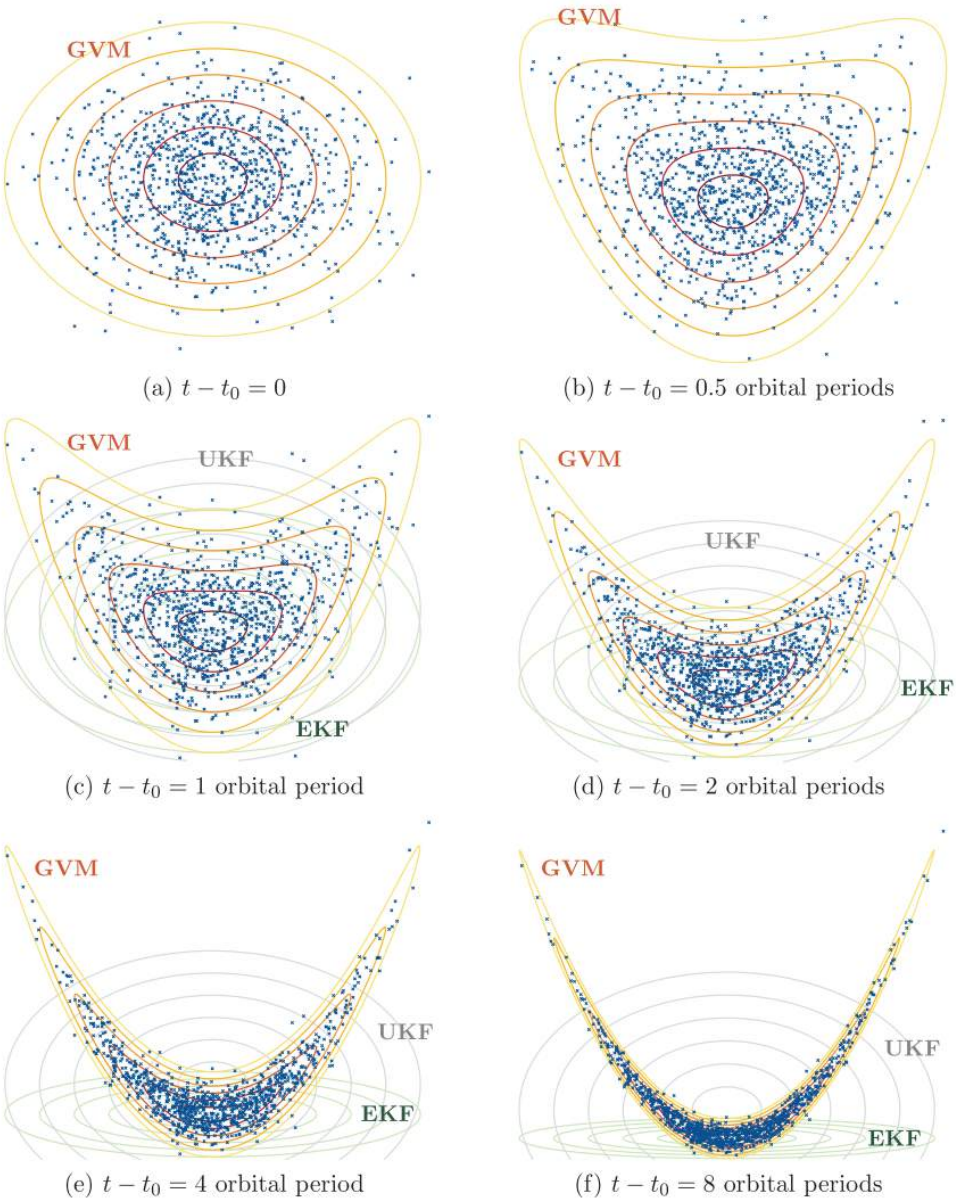


Figure 1.1: Evolution of equinoctial elements across propagation epochs [1].

represent arbitrary non-Gaussian distributions.

1.3 Optimal transport and triangular maps

To capture the complex, sheared distributions dictated by orbital mechanics, we turn to the mathematical theory of optimal transport. Originating from the problem of finding the most efficient way to move piles of earth to fill construction trenches [2], this discipline provides a rigorous framework for defining a continuous mathematical map that pushes a simple, easy-to-sample reference distribution onto a more complex target distribution (such as that discussed in Section 1.2).

Rather than propagating the physical probability measure directly through the nonlinear dynamical flow, optimal transport reframes uncertainty quantification as the problem of learning a continuous pushforward map between two distinct probability spaces. This deterministic transformation serves as a structural bridge, linking a tractable, well-characterised reference measure, such as an isotropic standard Gaussian, to the complex, sheared target distribution determined by the physical ground truth. By accurately reconstructing this map, it is possible to circumvent the local linearisation limitations of classical filters. Large ensembles of independent, identically distributed samples can be drawn from the reference space, mapped forward through the learned transformation, and used to recover the true, heavy-tailed non-Gaussian geometry of the evolved physical uncertainty cloud.

1.4 Computational challenges

Although triangular maps offer an elegant theoretical framework for probability transport, implementing this continuous abstraction as a practical computational tool presents substantial engineering challenges. For learning multi-dimensional maps, first-order optimisation methods such as projected gradient descent (PGD) provide notable computational advantages over more complex approaches, including quasi-Newton or interior-point methods, used in the literature [3].

Within the PGD framework proposed in this thesis, preserving the physical validity of the transported probability distribution requires that the learned map remain strictly monotonic. This requirement manifests as a large set of linear inequality constraints, defining a strict permissible region for the map's parameters. Geometrically, the space of valid parameters forms a sparse, multi-dimensional polyhedron.

When the PGD algorithm moves parameters outside this valid region, they must be mapped to the nearest point on the polyhedron's boundary through a process known as Euclidean projection.

1.5 Dykstra's algorithm and stalling

Because a Euclidean projection is required at every epoch of a high-frequency learning loop, the choice of projection algorithm is critical to the overall viability of the tracking framework. Standard interior-point optimisation software is often computationally prohibitive for large-scale, high-dimensional tasks. Dykstra's projection algorithm offers a computationally efficient and scalable method for computing these projections. Widely used in engineering disciplines such as restricted least squares in machine learning and model predictive control, Dykstra's algorithm decomposes the often intractable joint projection into a sequence of simple iterative steps, resulting in exceptionally low per-iteration computational overhead.

However, Dykstra's algorithm exhibits a significant mathematical limitation known as the stalling phenomenon. In certain geometric configurations, particularly near sharp and complex intersections of multiple constraint planes, the iterates can enter arbitrarily long cycles. During these stalling periods, the algorithm fails to progress toward the true optimal projection. In safety-critical or real-time applications, where a fixed iteration budget is imposed to meet computational constraints (as in the optimal transport framework), stalling can render Dykstra's algorithm infeasible. Additionally, the standard implementation lacks optimisation enhancements, a deficiency that must be addressed to enable its integration into real-time or safety-critical software.

This thesis addresses the intersection of fundamental algorithmic optimisation and high-dimensional physical modelling. As a cross-disciplinary collaboration between the Operations Research and Financial Engineering (ORFE) and Mechanical and Aerospace Engineering (MAE) departments, the core contributions are dual in nature, comprising two distinct yet interconnected main sections.

First, this work resolves the stalling phenomenon in Dykstra's algorithm. Through rigorous analysis of the geometry of polyhedral constraints, an exact, closed-form mathematical solution for the duration of the stalling period is derived. Building on this result, a fast-forward modification to the algorithm is introduced. Upon detection of a stall, a single, computationally inexpensive update advances the internal memory variables to the end of the stall. Together with additional optimisation enhancements,

these modifications eliminate unpredictable convergence behaviour while preserving strong asymptotic convergence guarantees, transforming the algorithm into a robust engine for real-time and safety-critical engineering applications.

Second, we architect a scalable software framework for optimal transport in orbital mechanics. Leveraging the fast-forward modified Dykstra solver, we construct a robust PGD framework to learn triangular maps. By successfully reconstructing nonlinear physical ground truth shears, we establish this combined engine as a viable, highly performant alternative to traditional filtering. Ultimately, we demonstrate that optimal transport, when coupled with rigorous mathematical optimisation, can fundamentally advance the state-of-the-art in space domain awareness. This applied framework enables dynamic tracking of complex, non-Gaussian uncertainty distributions across diverse orbital regimes, providing a critical computational tool for ensuring safe, reliable, and autonomous operations in the modern space environment.

1.6 Thesis outline

The remainder of this thesis is structured as follows.

Chapter 2 focuses on the theoretical advancements to Dykstra's projection algorithm. It introduces the geometry of Euclidean projections and polyhedral sets, formally defines the stalling phenomenon, and presents the mathematical proof for the closed-form duration of the stall. The chapter concludes by defining the fast-forward modified algorithm and demonstrating its superiority on pathological constraint geometries.

Chapter 3 pivots to the aerospace application, formally defining the theory of optimal transport and the structure of triangular maps. It details the parameterisation of these maps and formulates the monotonicity requirements into a constrained optimisation problem.

Chapter 4 details the architecture of the custom Python software framework. It outlines the integration of the outer learning loop, featuring stochastic gradient descent and several other dynamic learning schedules.

Chapter 5 presents the empirical results of applying the framework to orbital mechanics and geophysical applications. It successfully reconstructs physical ground-truth shears and provides performance-scaling metrics demonstrating the computational efficiency of the combined architecture.

Finally, Chapter 6 summarises the primary findings, details the broader implications for both the optimisation and astrodynamics communities, and proposes avenues

for future theoretical and applied work.

Chapter 2

Fast-forwarding stalling in Dykstra's algorithm

Deploying constrained learning algorithms requires a highly efficient method for computing Euclidean projections. Dykstra's method offers a computationally lightweight solution to find the projection of an initial point onto the intersection of multiple convex sets [4]. However, despite its strong theoretical guarantees, Dykstra's algorithm is mathematically encumbered by the stalling phenomenon [5]. Because the duration of these stalling periods is unknown *a priori*, the algorithm cannot be reliably restricted to a fixed iteration budget. This unpredictable convergence behaviour inhibits the deployment of Dykstra's method in practical, real-time computational software.

In this chapter, we address the stalling problem of Dykstra's method for polyhedral sets. Exploiting simplifications that arise in the polyhedral case, we derive the exact closed-form duration of the stalling period once stalling is detected. This mathematical derivation enables a modified version of Dykstra's algorithm that instantly fast-forwards through the stalling period, thereby eliminating unpredictable computational cycles. Crucially, this modification preserves all asymptotic convergence guarantees of the original algorithm. This theoretical development forms the fundamental optimisation pillar of this thesis, transforming Dykstra's method into a robust, predictable engine ready for safety-critical engineering applications.

Notation

We denote the transpose of a vector w as w^T, and its ℓ_2-norm as $\|w\|_2$. The interior of a set $\mathcal{H}$ is denoted as $\operatorname{int}\mathcal{H}$, and its boundary is denoted as $\partial\mathcal{H}$. Function composition is written as $f \circ g$, and the modulo and ceiling operators are denoted by $[\cdot]$ and $\lceil\cdot\rceil$, respectively. The absolute inner Dykstra iteration index is denoted as m and raised as a bracketed superscript (e.g., $w^{(m)}$), and the dimension of the decision variable is denoted as d. A standard normal distribution is denoted as $\mathcal{N}(\cdot,\cdot)$. The individual half-space index is denoted as i, while the total half-space count is denoted as n and remains fixed. We name an update with an increase by 1 in m ($m \leftarrow m+1$) an *iteration*, and refer to an advancement of n iterations ($m \leftarrow m+n$) as a *cycle*.

2.1 Background on Euclidean projections

The problem of projecting an arbitrary point in a high-dimensional space onto a restricted feasible region is ubiquitous in applied mathematics [6]. This fundamental operation underpins diverse disciplines, from computing restricted least squares in machine learning to the enforcement of strict safety bounds in model predictive control. Formally, we consider a Euclidean space equipped with the standard ℓ_2-norm, $\|w\|_2 \coloneqq (w_1^2+\cdots+w_d^2)^{1/2}$, where d represents the dimensions of the decision variable w. We define a finite family of n closed convex subsets $\{\mathcal{H}_i\}_{i=0}^{n-1}$ possessing a nonempty intersection $\mathcal{H} \coloneqq \bigcap_{i=0}^{n-1}\mathcal{H}_i$. Given a proposed unconstrained point $w^\circ \in \mathbb{R}^d$, its exact Euclidean projection $w^\star =: \mathcal{P}_{\mathcal{H}}(w^\circ)$ onto the feasible set $\mathcal{H}$ is obtained by solving the constrained quadratic program (CQP):

$$\begin{aligned} \underset{w\in\mathbb{R}^d}{\text{minimise}} \quad & \tfrac{1}{2}\|w-w^\circ\|_2^2 \\ \text{subject to} \quad & w \in \textstyle\bigcap_{i=0}^{n-1}\mathcal{H}_i. \end{aligned} \tag{2.1.1}$$

In this thesis, we restrict our analysis to the case where $\mathcal{H}$ is a polyhedral set. This polyhedron can be represented compactly in matrix form as $\mathcal{H} = \{w \in \mathbb{R}^d \mid Aw \le b\}$, with constraint matrix $A \in \mathbb{R}^{n\times d}$ and offset vector $b \in \mathbb{R}^n$. Correspondingly, each individual set $\mathcal{H}_i$ represents a single half-space constraint, written as:

$$\mathcal{H}_i \coloneqq \{w \in \mathbb{R}^d \mid a_i^T w \le b_i\}, \quad i = 0,\dots,n-1, \tag{2.1.2}$$

where vectors a_i are unit normals to bounding hyperplanes $\partial\mathcal{H}_i \coloneqq \{w \in \mathbb{R}^d \mid a_i^T w = b_i\}$.

While the projection onto a single half-space $\mathcal{H}_i$ is trivial to compute analytically, the joint projection operator $\mathcal{P}_{\mathcal{H}}$ onto the arbitrary intersection of several half-spaces is mathematically intractable, meaning no direct closed-form solution to (2.1.1) exists. In traditional settings, the solution to (2.1.1) is typically delegated to standard interior-point optimisation software. However, for large n, these solvers require introducing additional variables $z := Aw$ (e.g., [7] or [8]) that are projected onto the set $\{z \in \mathbb{R}^n \mid z \leq b\}$, for which an explicit solution exists [6]. However, for large n, this variable augmentation can reduce the computational efficiency, particularly in high-dimensional or time-sensitive applications when the projection is part of an overarching algorithm [9, 4].

2.2 Dykstra's projection algorithm

As a lightweight alternative to heavy interior-point solvers, Dykstra's projection algorithm [10] provides an iterative method to solve (2.1.1). It builds upon the method of alternating projections (MAP), which cyclically projects on the constraints without any additional variables added [11, 12]. Starting from an initial point $w^{(0)} = w^{\circ}$, MAP computes successive projections onto each individual half-space,

$$w^{(m+1)} = (P_{\mathcal{H}_{n-1}} \circ \cdots \circ P_{\mathcal{H}_0})(w^{(m)}). \tag{2.2.1}$$

This algorithm was first developed by John Von Neumann in 1951. While MAP is guaranteed to eventually find a feasible point within the intersection $\mathcal{H}$, it does not guarantee that this point will be the true Euclidean projection $w^{\star}$ [12]. As an example, consider the case outlined in Figure 2.1, where an original point $(-0.2, 1.2)$ (green dot) is projected onto the intersection of two "balls" (red, grey, approximated by half-spaces). After projecting onto each of the half-spaces once, the iterate lands at a point within the intersection of the two "balls" (green cross). On the next MAP cycle, each of the half-space inequalities (2.1.2) is already satisfied, and the algorithm terminates. From inspection, however, it is evident that the green cross does not correspond to the optimal solution, since there are points in the intersection of the two "balls" that are closer to the green dot (namely the top-left "corner" of the feasible region). Thus, the algorithm has terminated before finding the optimal solution.

Dykstra's algorithm resolves the premature termination by introducing auxiliary variables $e^{(m)} \in \mathbb{R}^d$ that store the negative error vectors associated with the projec-

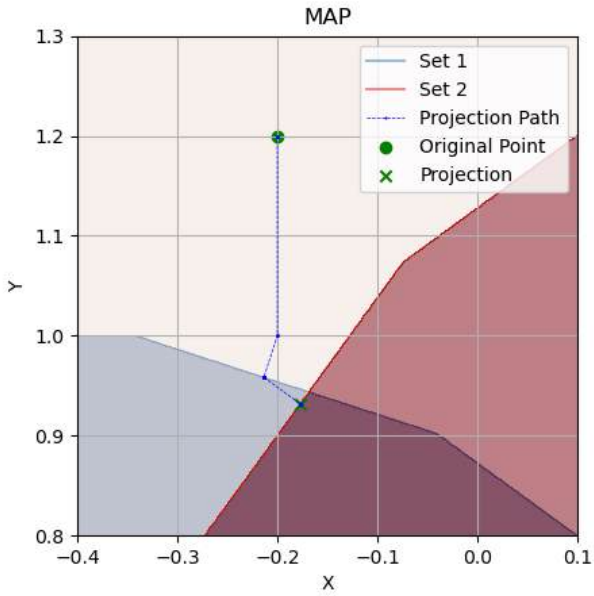


Figure 2.1: The method of alternating projections yielding a sub-optimal solution.

tion of a primary variable onto a given half-space [10]. The algorithm proceeds by generating a series of primary iterates $\{w^{(m)}\}$ and auxiliary iterates $\{e^{(m)}\}$ using the following scheme:

$$w^{(m+1)} = \mathcal{P}_{\mathcal{H}_{[m]}}\left(w^{(m)} + e^{(m-n)}\right), \tag{2.2.2a}$$

$$e^{(m)} = e^{(m-n)} + w^{(m)} - w^{(m+1)}, \tag{2.2.2b}$$

where $[m]$ represents the modulus operator $[m] \coloneqq m \bmod n$. The decision variable is initialised as $w^{(0)} = w^{\circ}$, and the auxiliary variables are initialised to zero as $e^{(-n)} = e^{(-(n-1))} = \cdots = e^{(-1)} = 0$.

By systematically storing the outward normal of the projection step, Dykstra's algorithm alters the subsequent augmented projections. As illustrated in Figure 2.2, after the first cycle completes, the addition of the auxiliary memory variable shifts the pre-projection point strictly outside the interior of the feasible region, forcing the algorithm to compute a further projection. This subsequent operation pulls the primary iterate (green cross) closer to the true Euclidean projection after the second cycle, effectively circumventing the sub-optimal termination observed in MAP.

In fact, by adding these auxiliary vectors back into the primary variables before subsequent projections, the Boyle-Dykstra theorem [10, 13] implies asymptotic

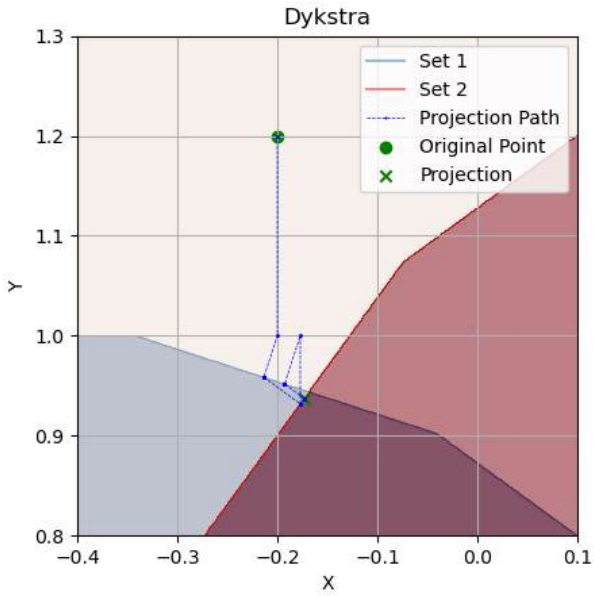


Figure 2.2: Dykstra's algorithm circumventing premature termination.

convergence to the true projection, guaranteeing that

$$\lim_{m\to\infty} \left\| w^{(m)} - \mathcal{P}_{\mathcal{H}}(w^{\circ}) \right\|_2 = 0.$$

For the specific case of polyhedral sets defined in (2.1.2), the projection step (2.2.2a) of Dykstra's method can be greatly simplified. If the augmented point $(w^{(m)} + e^{(m-n)})$ already satisfies the current constraint $\mathcal{H}_{[m]}$, no action is required and the point remains unchanged. If it violates the constraint, it is orthogonally projected onto the boundary hyperplane. This simplifies the projection update to:

$$w^{(m+1)} = \begin{cases} w^{(m)} + e^{(m-n)} & \text{if } w^{(m)} + e^{(m-n)} \in \mathcal{H}_{[m]} \\ w^{(m)} - \left((w^{(m)})^T a_{[m]} - b_{[m]} \right) a_{[m]} & \text{otherwise} \end{cases}, \tag{2.2.3}$$

and the corresponding update for the auxiliary vector (2.2.2b) to:

$$e^{(m)} = \begin{cases} 0 & \text{if } w^{(m)} + e^{(m-n)} \in \mathcal{H}_{[m]} \\ e^{(m-n)} + \left((w^{(m)})^T a_{[m]} - b_{[m]} \right) a_{[m]} & \text{otherwise} \end{cases}. \tag{2.2.4}$$

Because these updates rely purely on vector additions and scalar projections rather

than matrix operations, Dykstra's method presents a computationally inexpensive per-iteration overhead. The auxiliary vector $e^{(m)}$ is either zero or parallel to the half-space normal $a_{[m]}$, so it can be characterised via a scalar quantity $k^{(m)} \in \mathbb{R}$ as $e^{(m)} = k^{(m)} a_{[m]}$ with

$$k^{(m)} = \begin{cases} 0 & \text{if } w^{(m)} + k^{(m-n)} a_{[m]} \in \mathcal{H}_{[m]} \\ k^{(m-n)} + \left(w^{(m)}\right)^T a_{[m]} - b_{[m]} & \text{otherwise} \end{cases}. \tag{2.2.5}$$

The convergence of Dykstra's iterates to the Euclidean projection has been analysed in [13, 14, 15] for polyhedral sets. The analysis in [14] is based on partitioning the collection $\{\mathcal{H}_i\}_{i=0}^{n-1}$ into two subsets. The *inactive* subset contains half-spaces where the true projection $w^{(\infty)}$ lies strictly in their interior $\operatorname{int} \mathcal{H}_i$, and the *active* subset contains half-spaces where the projection lies exactly on their boundary $\partial \mathcal{H}_i$. The two subsets, $\mathcal{A}$ and $\bar{\mathcal{A}}$, respectively, are defined as:

$$\mathcal{A} := \left\{ i \in \{0, \dots, n-1\} \;\middle|\; w^{(\infty)} \in \partial \mathcal{H}_i \right\}, \tag{2.2.6}$$

$$\bar{\mathcal{A}} := \{0, \dots, n-1\} \setminus \mathcal{A} = \left\{ i \in \{0, \dots, n-1\} \;\middle|\; w^{(\infty)} \in \operatorname{int} \mathcal{H}_i \right\}, \tag{2.2.7}$$

where $w^{(\infty)} := \lim_{m \to \infty} w^{(m)}$.

It can be shown that there exists an integer N_1 such that when $[m] \in \bar{\mathcal{A}}$ at iteration $m \geq N_1$, it follows that $w^{(m)} = w^{(m-1)}$ and $e^{(m)} = 0$. This indicates that the half-spaces that become inactive remain so permanently [14, Lemma 3.1]. Furthermore, there exists an integer $N_2 \geq N_1$ such that whenever $m \geq N_2$, it holds that $\|w^{(m+n)} - w^{(\infty)}\|_2 \leq \alpha_{[m]} \|w^{(m)} - w^{(\infty)}\|_2$, where $0 \leq \alpha_{[m]} < 1$ are scalars related to the geometric angles between the half-spaces [14, Lemma 3.7]. The integer N_2 describes the iteration from which the algorithm has definitively identified the inactive half-spaces.

It then follows that there exist constants $0 \leq c < 1$ and $\rho > 0$ such that $\|w^{(m)} - w^{(\infty)}\|_2 \leq \rho\, c^m$ [14, Thm. 3.8]. The convergence rate constant c can be estimated from the smallest $\alpha_{[m]}$, which is fundamentally characterised by the angles between the subspaces formed by the active half-spaces. The scaling constant ρ, however, depends on an unknown iteration $N_3 \geq N_2$ and on the initial unconstrained point w°. Consequently, it cannot be computed in advance [13, 16].

2.3 The stalling phenomenon

Despite its strong theoretical convergence guarantees (Boyle-Dykstra Theorem) and low computational overhead, Dykstra's algorithm is known to suffer from a computational flaw known as the *stalling* phenomenon [5]. For specific geometric configurations, particularly near "corner" intersections of multiple constraint hyperplanes, the iterates can enter repetitive cycles. During these cycles, the primary iterate $w^{(m)}$ ceases to progress towards the optimal projection $w^\star$, remaining strictly unchanged for an unforeseen number of iterations.

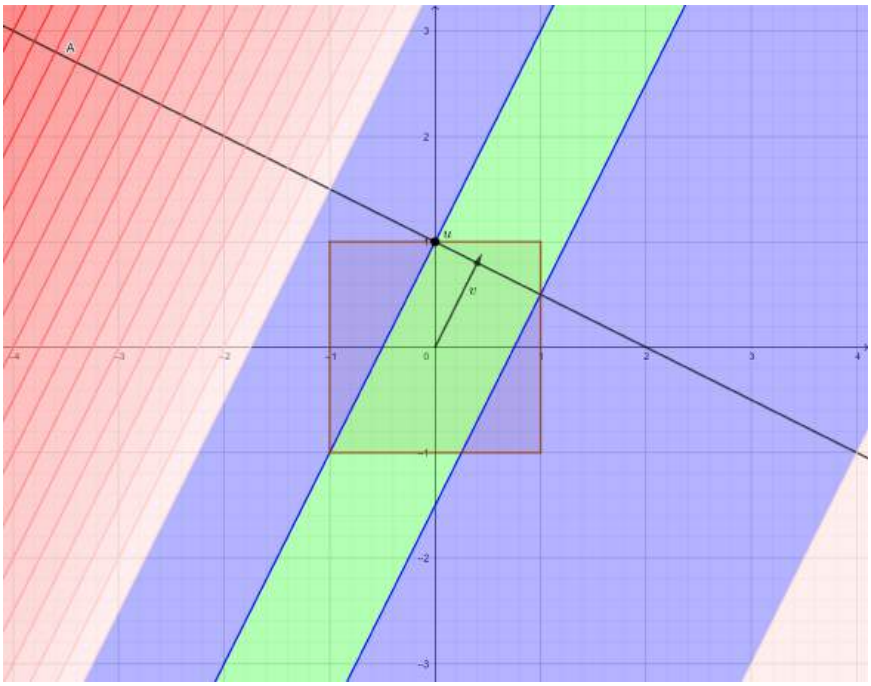


Figure 2.3: Stalling regions for the line-box example as defined in [5].

To understand the mechanics of this stalling effect, we examine a foundational geometric example provided by Bauschke et al. [5]. The authors analyse the behaviour of Dykstra's method for a simple polyhedral set in $\mathbb{R}^2$, defined by the intersection of a unit box and a line. They establish that the convergence behaviour of the algorithm is entirely dictated by the spatial location of the initial unconstrained point w°. By evaluating different starting conditions, the state space can be partitioned into three distinct regions, yielding fundamentally different convergence properties:

1. Finite convergence: The algorithm identifies the true projection $w^\star$ in a finite number of cycles.

2. Infinite convergence: The iterates approach $w^\star$ asymptotically without stalling.

3. Stalling followed by infinite convergence: The iterates become trapped in a static cycle for a finite duration before breaking free and converging asymptotically.

These three starting domains are visualised in Figure 2.3 as the green, blue, and red regions, respectively.

When the starting point w° originates within the red region, the stalling phenomenon is triggered. Assuming the algorithm projects onto the box constraints first, followed by the line, the primary iterate $w^{(m)}$ will repeatedly land exactly on the top-left corner of the box. For several consecutive cycles, Dykstra's algorithm returns this exact same sub-optimal coordinate. The authors demonstrate that the exact number of cycles required to break free from this stall is proportional to the distance of the starting point from the boundary of the red region. If w° is initialised arbitrarily far to the top-left, the algorithm will require an arbitrarily large number of iterations before the primary iterate moves away from the corner.

By graphically examining the line-box example and cross-referencing the algebraic update rules defined in (2.2.2), we can extract two fundamental mathematical observations regarding the behaviour of the algorithm during a stall. These observations form the analytical foundation for detecting and eventually bypassing the stalling period, which we later formalise in Section 2.4.

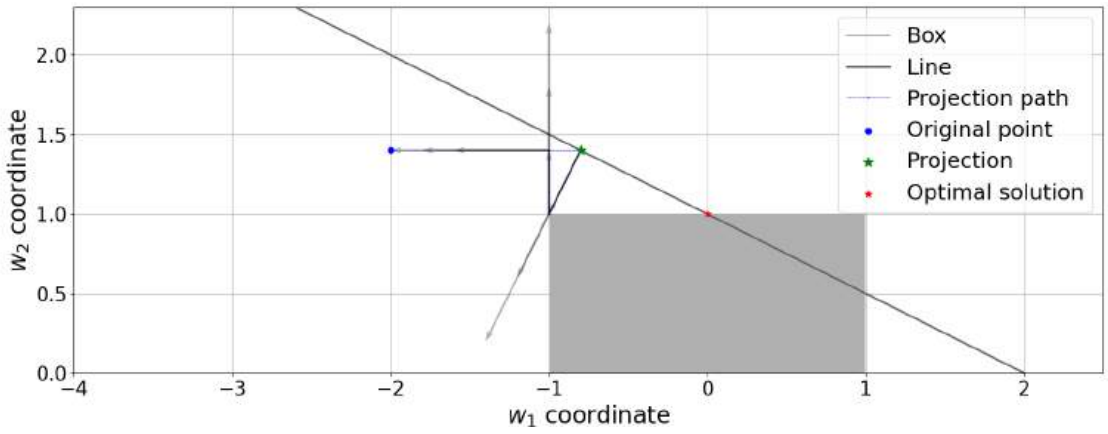


Figure 2.4: A demonstration of the stalling phenomenon for the line-box example.

First, the stalling period is strictly maintained as long as the active constraint set remains unchanged. The stall continues until at least one half-space that is active for the initial point w°, but is ultimately inactive for the true optimal projection $w^\star$, transitions from active to inactive. This condition is necessary for the algorithm to break the cycle and resume its descent towards the optimal solution.

Second, during the stalling phase, the primary iterates exhibit strict periodicity.

For every iteration m within the stall duration, it holds that $w^{(m)} = w^{(m-n)}$. Because the primary variables remain "spatially frozen" across cycles, the auxiliary update rule dictates that a constant vector is repeatedly added to the error variables $e^{(m)}$. Specifically, at each cycle, a constant corrective vector $w^{(m-1)} - w^{(m)}$ is summed to the respective auxiliary variable $e^{(m)}$. This predictable linear accumulation of the error variables is what ultimately forces the augmented pre-projection point to cross a hyperplane boundary, thereby breaking the stall.

These effects are visualised for the same line-box example in Figures 2.4–2.5, where we initialise the unconstrained iterate at $w^\circ = (-2.0, 1.4)$ (which lies within the red region of Figure 2.3) and run Dykstra's algorithm for a few cycles.

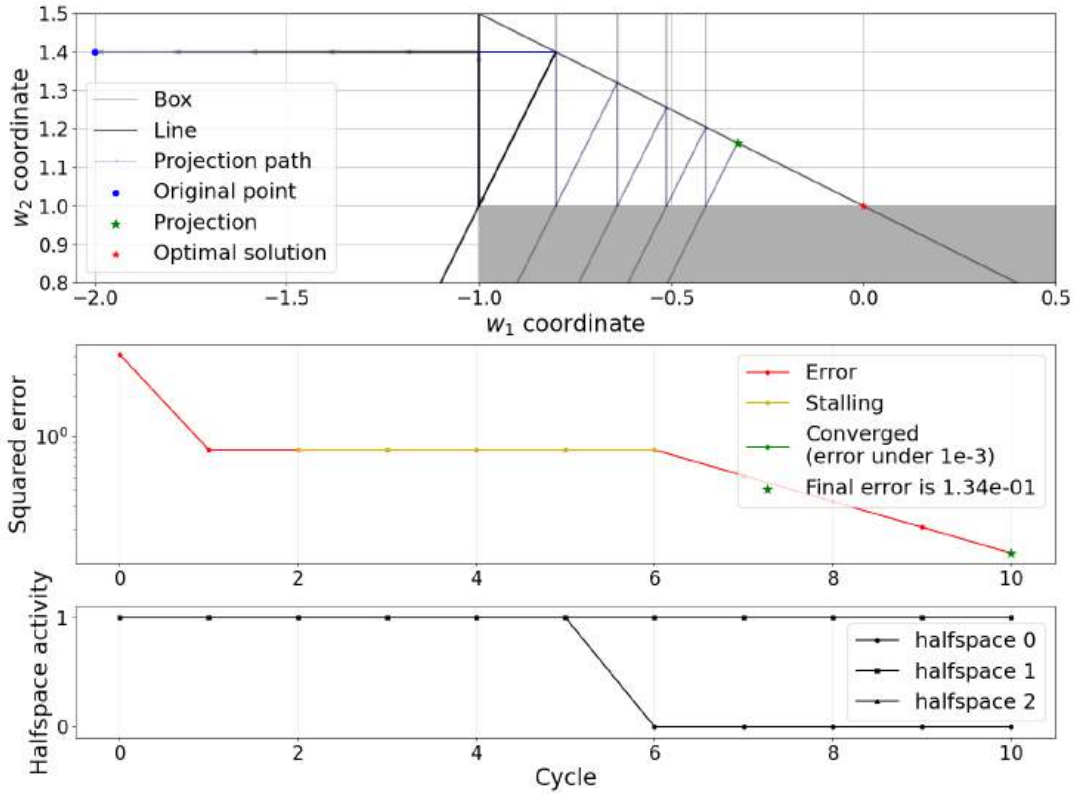


Figure 2.5: Stalling break-through for the line-box example.

Figure 2.4 plots the line and box as grey shaded regions, and the unconstrained iterate w° as a blue dot. Three cycles of the algorithm are run, and the error variables $e^{(m)}$ are plotted as grey quivers. At the beginning of the run, there are three active half-spaces, namely the left and top side of the box, and the top-right side of the line. Dykstra's algorithm proceeds by projecting on the left side of the box first, then the top side of the box, and ultimately the top-right side of the line. The error quivers are plotted with their tail at the projected iterate $w^{(m+1)}$ and their heads at

the unprojected iterate $w^{(m)}$. We can see that the error iterate associated with the left side of the box is shrinking, whereas the other two are growing, all by respective constant factors. From this figure, we can foresee that the active half-space that will become inactive to break stalling is the left side of the box, and stalling will end when the error variable associated with this half-space becomes negative.

Figure 2.5 plots the same scenario with a zoomed-in view (top plot), and, additionally, the squared errors at the end of each cycle (defined as the squared ℓ_2-norm of the iterates minus optimal solution $w^{(m)} - w^\star$, middle plot), and the activity of the half-spaces (1 for *active*, 0 for *inactive*, bottom plot), where the left side of the box, top side of the box and top-right side of the line correspond to halfspaces 0, 1, and 2, respectively. After 6 cycles, the left side of the box becomes inactive, and the squared error, which had remained constant through the stalling period, starts shrinking again. The algorithm has effectively escaped stalling, which for this example lasted for $N = 5$ cycles.

2.4 A closed-form solution for the stalling period

As established in Section 2.3, the stalling phenomenon is fundamentally characterised by a strict periodicity in the primary iterates and a corresponding linear accumulation within the auxiliary error variables. Based on the geometric observations discussed in [5], we can mathematically formalise the onset of this stalling phase.

Definition 2.4.1 (Stalling)**.** Given an iteration index $m \geq n - 1$, a stalling period is defined as a sequence of iterations $i \in \{0, \dots, m_{\text{stall}}\}$ with $m_{\text{stall}} \geq n$ for which the primary iterates exhibit strict cyclical repetition, such that $w^{(m+i)} = w^{(m+i-n)}$.

During a defined stalling period, the spatial coordinates of the primary variables $w^{(m+i)}$ remain constant across successive cycles. Consequently, only the auxiliary variables $e^{(m+i)}$ experience modification. As dictated by the update rule, these auxiliary variables are adjusted at each cycle by constant corrective increments $w^{(m)} - w^{(m+1)}$.

The stalling period will therefore strictly continue until the cumulative magnitude of the auxiliary variable $e^{(m+i)}$ becomes sufficiently large to satisfy

$$w^{(m+i)} + e^{(m-n+i)} \in \mathcal{H}_{[m+i]}$$

for a given constraint $[m]$. Geometrically, this indicates that the specific half-space

associated with the stall transitions from an active state to an inactive state. For polyhedral sets, because the auxiliary accumulation is strictly linear, the exact length of the stalling period can be deterministically pre-computed as soon as the algorithm detects the stalling condition. This computation relies exclusively on the active constraints at the onset of the stall and is formalised in Theorem 2.4.2.

Theorem 2.4.2 (Length of Stalling Period)**.** *Suppose that stalling commences at iteration m as outlined in Definition 2.4.1. Let the active half-spaces be denoted by the set $\mathcal{A} \coloneqq \{i \in \{0, \dots, n-1\} \mid w^{(m+i)} + e^{(m-n+i)} \notin \mathcal{H}_{[m+i]}\}$. The total length of the stalling period is given by:*

$$M_{\text{stall}} \coloneqq \min_{i \in \mathcal{S}} \left\lceil \frac{k^{(m-n+i)}}{b_{[m+i]} - (w^{(m+i)})^T a_{[m+i]}} \right\rceil, \tag{2.4.1}$$

where $\lceil \cdot \rceil$ represents the ceiling operator, $k^{(m-n+i)} = (e^{(m-n+i)})^T a_{[m+i]}$, and the candidate set $\mathcal{S}$ is defined as $\mathcal{S} \coloneqq \{i \in \mathcal{A} \mid (w^{(m-n+i)})^T a_{[m+i]} - b_{[m+i]} < 0\}$.

Proof. The existence of a finite integer M_{stall} is a direct consequence of the asymptotic convergence guaranteed by the Boyle-Dykstra theorem. Following the conditional logic in the simplified polyhedral projection step (2.2.3), the stalling period will strictly terminate once an active half-space becomes inactive. This transition is mathematically expressed as $w^{(m+i)} + e^{(m-n+i)} \in \mathcal{H}_{[m+i]}$ for some half-space index $i \geq n$ under Definition 2.4.1.

As a direct consequence of the scalar update rule detailed in (2.2.5), the only half-spaces capable of transitioning to an inactive state are those for which the spatial evaluation quantity $(w^{(m+i)})^T a_{[m+i]} - b_{[m+i]}$ is strictly negative. By Definition 2.4.1, because the primary iterates are "frozen" in space, this quantity remains constant throughout the stalling duration.

The requisite number of cycles to break the stall for any single half-space can therefore be obtained by determining the smallest possible integer M_i that satisfies the inequality:

$$k^{(m-n+i)} + M_i \left((w^{(m+i)})^T a_{[m+i]} - b_{[m+i]} \right) < 0, \qquad i \in \mathcal{A}.$$

Solving this inequality for M_i yields $M_i = \left\lceil k^{(m-n+i)} / \left(b_{[m+i]} - (w^{(m+i)})^T a_{[m+i]} \right) \right\rceil$. The global duration of the stalling period is subsequently the minimum across all candidate half-spaces, establishing $M_{\text{stall}} = \min_{i \in \mathcal{S}} M_i$. The specific half-space that triggers the termination is indexed by $i_{\text{stall}} = \operatorname{argmin}_{i \in \mathcal{S}} M_i$. If multiple indices $i_1 < \cdots < i_j$ yield identical values such that $M_{i_1} = \cdots = M_{i_j}$, the sequential update structure of Dykstra's algorithm defined in (2.2.2) ensures that the half-space evaluated earliest in the sequence, $i_{\text{stall}} \coloneqq i_1$, is discarded first. □

It is important to note that the quantity M_{stall} derived in Theorem 2.4.2 strictly refers to the number of full n-step cycles of (2.2.2) that must elapse. Following the conclusion of the stalling period, one specific constraint is discarded from the active half-space set, and the subset $\mathcal{A}$ must be redefined. Upon this redefinition and subsequent spatial movement, the algorithm may again navigate into a separate geometric corner that triggers a new stalling condition.

2.5 The fast-forward modified algorithm

By leveraging the geometric characteristics defined in Theorem 2.4.2, the unpredictable computational trap of the stalling period is resolved. Because the exact duration of the stall is known and the accumulation of the auxiliary variables is strictly linear, the entire stalling phase can be bypassed in a single computational step. By applying constant corrective increments directly to the auxiliary memory variables, we can instantly fast-forward the algorithm's state to the exact iteration where the active constraint breaks at the moment when stalling is detected. Theorem 2.4.2 and the baseline Dykstra update are successfully combined in Algorithm 1.

The proposed algorithm proceeds identically to the standard Dykstra update until it actively detects a stall. This detection is achieved by verifying the strict stationarity condition $w^{(m)} \approx w^{(m-n)}$ across a full cycle of half-spaces, governed by a small, positive numerical tolerance $\varepsilon_{\text{stall}}$.

Once a stall is detected at iteration m, the algorithm interrupts the standard sequential loop and executes its fast-forwarding phase. It first computes the exact number of required stalling cycles, M_{stall}, and identifies the specific half-space at which the stall terminates, i_{stall}, by invoking Theorem 2.4.2. For each of the n auxiliary variables in the current cycle, the algorithm computes the constant geometric increment $\Delta e^{(j)} = \left((w^{(\text{stall})})^T a_j - b_j \right) a_j$ and applies a single, aggregated update:

Algorithm 1 Dykstra's projection algorithm for polyhedral sets with fast-forwarding.

Inputs: $w^\circ, \{a_i, b_i\}_{i=0}^{n-1}, M_{\max}, \varepsilon_{\text{stall}}$
Output: Euclidean projection $w^{(m)}$

1: $w^{(0)} \leftarrow w^\circ$; $e^{(j)} \leftarrow 0 \; \forall j \in \{-n, \dots, -1\}$; $m \leftarrow 0$
2: **while** $m < M_{\max}$ **do**
3: $\quad w^{(m+1)} \leftarrow \mathcal{P}_{\mathcal{H}_{[m]}}(w^{(m)} + e^{(m-n)})$ ▷ Standard polyhedral projection (2.2.3)
4: $\quad e^{(m)} \leftarrow e^{(m-n)} + w^{(m)} - w^{(m+1)}$ ▷ Standard auxiliary update (2.2.4)
5: $\quad$ **if** $m \geq n-1$ **and** $\|w^{(m+1-i)} - w^{(m+1-i-n)}\|_2 < \varepsilon_{\text{stall}} \; \forall i \in \{0, \dots, n-1\}$ **then** ▷ Stall detected per Definition 2.4.1
6: $\quad\quad$ Obtain M_{stall} and i_{stall} from Theorem 2.4.2
7: $\quad\quad$ **for** $j \in \{0, \dots, n-1\}$ **do**
8: $\quad\quad\quad M_{\text{skip}} \leftarrow M_{\text{stall}}$ **if** $[m-j] \leq i_{\text{stall}}$ **else** $M_{\text{stall}} - 1$
9: $\quad\quad\quad e^{(m-j)} \leftarrow e^{(m-j)} + M_{\text{skip}}((w^{(m-j)})^T a_{[m-j]} - b_{[m-j]}) a_{[m-j]}$ ▷ Fast-forward memory variables
10: $\quad\quad m \leftarrow m - i_{\text{stall}}$ ▷ Adjust iteration counter
11: $\quad$ **else**
12: $\quad\quad m \leftarrow m + 1$
13: **return** $w^{(m-1)}$

$e^{(j)} \leftarrow e^{(j)} + M_{\text{skip}} \Delta e^{(j)}$. The multiplier M_{skip} is set to M_{stall} if $i_{\text{stall}} \geq j$ and $M_{\text{stall}} - 1$ otherwise, ensuring the internal sequencing aligns with the break condition.

This single operation artificially "jumps" the memory variables forward to their exact mathematical state following the stalling period, entirely bypassing several wasted cycles. Finally, the main iteration counter m is adjusted backwards to reflect the index within the cycle where the stall was broken. For practical deployment within an overarching learning framework, Algorithm 1 is bound by a maximum iteration budget $M_{\max}$. Certain technical implementation details, such as the dynamic updating of the active half-space set $\mathcal{A}$ throughout the algorithm, are omitted here for clarity but are strictly enforced in the software architecture.

Because the proposed fast-forward modification instantly advances the internal memory variables to the exact state they would have eventually reached natively under Dykstra's method, the mathematical integrity of the algorithm is fully preserved (lines 5 to 10). Consequently, the strong asymptotic convergence properties established by the Boyle-Dykstra theorem [10] are retained, ensuring that the primary iterates converge to the true Euclidean projection. This is formalised in Corollary 2.5.1.

Corollary 2.5.1 (Convergence of Algorithm 1 [10])**.** *The sequence of primary iterates* $\{w^{(m)}\}$ *generated by Algorithm 1 converges asymptotically to the true Euclidean optimal solution* $w^\star = \mathcal{P}_{\mathcal{H}}(w^\circ)$*, such that* $\lim_{m\to\infty} \|w^{(m)} - w^\star\|_2 = 0$.

2.6 Numerical experiments on stalling resolution

We demonstrate the fundamental efficacy of the fast-forward modification through two different example. The first, discussed in Section 2.6.1, comprises the foundational motivating experiment from [5]. The second acts as a numerical benchmark between the "vanilla" implementation and our implementation (Algorithm 1) on randomly generated constraints, and is discussed in section 2.6.2. The results of our experiments can be reproduced from the first of two repositories that accompany this thesis [17, 18], which can be found at:

ClouD-161803/Dykstra-Project.git.

2.6.1 The line-box example

The selected example evaluates the Euclidean projection of an initial point w° onto the intersection of a box and a single line in $\mathbb{R}^2$ ($d = 2$). The box is centred at the origin and is defined by the bounds $[-1, 1] \times [-1, 1]$, while the line passes through the spatial coordinates $(0, 1)$ and $(2, 0)$. This specific pathological configuration was first described in detail by Bauschke et al. [5], which motivates our research on stalling for Dykstra's algorithm.

The initial point was explicitly set to $w^\circ = (-4, 1.4)$ to artificially induce a prolonged stalling period. To serve as an absolute benchmark for error calculations, the ground-truth optimal solution $w^\star$ was independently computed using the heavy interior-point CQP solver `quadprog` [19, 20]. During the execution of both projection algorithms, individual half-space activity was monitored by verifying the condition defined in (2.1.2) for each set at every iteration. The overall convergence behaviour was tracked via the squared ℓ_2-norm of the absolute spatial distance to optimality, defined as $E(w^{(m)}) := \|w^{(m)} - w^\star\|_2^2$.

The comparative results for both the standard Dykstra method and Algorithm 1 are visualised in Figure 2.6. The top plot displays the spatial trajectories of the primary iterates $w^{(m)}$ for both algorithms, with the static stalling cycle of the baseline Dykstra method explicitly highlighted in red. The middle plot tracks the corresponding squared spatial errors $E(w^{(m)})$, where the identical stalling period is similarly highlighted. Finally, the bottom plot illustrates the binary activity of the vertical half-space corresponding to the left boundary of the box. In this tracking metric, a value of 1 indicates that the half-space is active and 0 indicates inactivity. It is important to note that the spatial trajectories of the iterates $w^{(m)}$ in generated by

the baseline Dykstra method and Algorithm 1 are mathematically identical (with the exception of stalling), rendering them visually indistinguishable in the top spatial plot.

The error profile in the middle plot clearly indicates that the standard Dykstra algorithm stalls, making zero progress towards the optimum up to cycle 16. The recorded half-space activity confirms that this stall is maintained until the half-space corresponding to the left side of the box transitions to an inactive state.

In contrast, the application of Algorithm 1 definitively resolves this computational gridlock. As illustrated in the middle and bottom plots, the modified algorithm successfully detects the onset of the stall at cycle 2. Upon detection, it instantly calculates the required offset duration $M_{\text{stall}} = 14$ using Theorem 2.4.2 and fast-forwards

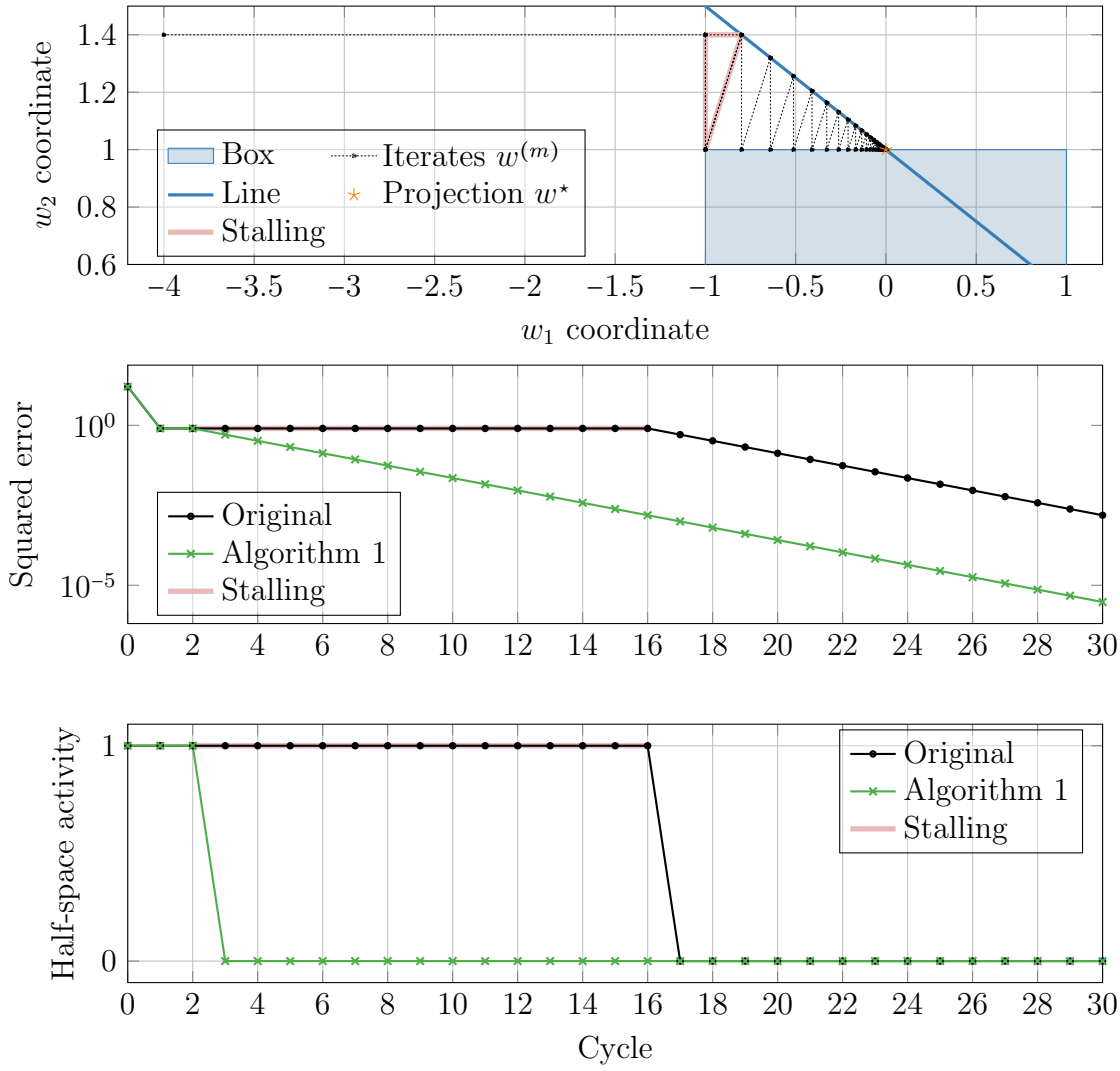


Figure 2.6: Dykstra's method and Algorithm 1 applied to the line-box example.

the memory variables. At cycle 3, the internal state of the modified algorithm mirrors the internal state the original algorithm achieves only at cycle 17. Consequently, the modified method exhibits superior convergence properties by eliminating 14 wasted computational cycles.

2.6.2 Benchmarking on randomly generated constraints

While the line-box scenario effectively isolates the mechanics of the stalling phenomenon in $d = 2$ dimensions with only $n = 3$ active half-spaces, the theoretical guarantees of Theorem 2.4.2 and Algorithm 1 extend strictly to any generic dimension d and any arbitrary number of polyhedral constraints n. To empirically validate the computational advantages of the fast-forward modification in high-dimensional environments relevant to modern engineering architectures, we evaluate the algorithm against dense, randomly generated polyhedral sets.

We construct a synthetic, highly constrained polyhedral environment designed to reliably induce the algorithmic stalling phenomenon. We define a strictly feasible reference point $w^\star \in \mathbb{R}^d$ drawn from a uniform random distribution. The polyhedron is formed by n randomly generated half-spaces. For each half-space, a normal vector a_i is sampled from a uniform random distribution and normalised. The boundary offsets b_i are calculated as $b_i = a_i^T w^\star + \varepsilon_{\text{margin}}$, where $\varepsilon_{\text{margin}} = 0.01$ serves as a small geometric margin. This construction guarantees that $w^\star$ is strictly interior of the polyhedron.

We then instantiate an unconstrained starting point $w^\circ = w^\star + 10\eta$, where the offset $\eta \sim \mathcal{N}(0, I_d)$ serves as a way to initialise the point outside of the polyhedron. The Euclidean distance between w° and the feasible boundary forces the projection solver to navigate sharp polyhedral vertices, thereby isolating the numerical cost of standard Dykstra stalling and explicitly highlighting the computational acceleration achieved by our fast-forward algorithmic modification. Both the standard Dykstra solver and the Stall-Detection solver (Algorithm 1) are initialised from the same starting point w° and executed against the same constraint matrix $A \in \mathbb{R}^{n \times d}$. Both algorithms are granted a fixed maximum budget of $M_{\max} = 100$ iterations.

We set up three reference examples with a random seed of `42` for reproducibility. The first example is 3-dimensional, and is designed to establish a baseline for convergence behaviour in a low-dimensional space with a moderate number of intersecting boundaries ($n = 6$). The second scenario restricts the state space to a plane ($d = 2$) but drastically increases the constraint density to $n = 20$, creating a geometric envi-

ronment designed to imitate overconstrained problems. The final configuration scales the problem to a high-dimensional regime ($d = 20$) bounded by $n = 50$ half-spaces, creating a complex, highly constrained geometry for a heavier computational load. The three sets of parameters for the benchmark are summarised in Table 2.1.

Table 2.1: Summary of the polyhedral environments used for the benchmark.

Experiment	Dimensions (d)	Half-spaces (n)	Seed	Iterations ($M_{\max}$)
(a)	3	6	42	100
(b)	2	20	42	100
(c)	20	50	42	100

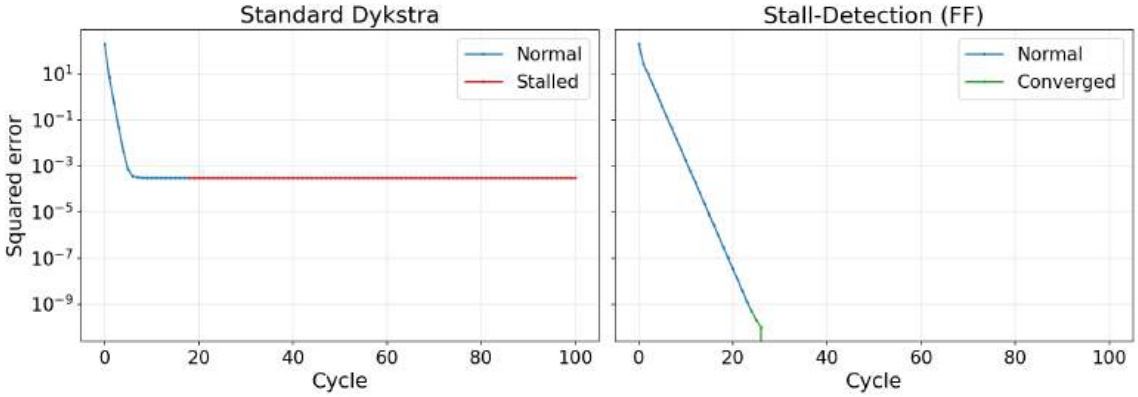
The convergence profiles for these three environments are visualised in Figure 2.7. During execution, the standard Dykstra algorithm runs into the stalling phenomenon for all three examples. In the low-dimensional configuration (Figure 2.7a), the baseline algorithm experiences a prolonged stalling plateau which does not terminate within the iteration budget, yielding a highly suboptimal final squared error. In the highly constrained planar case (Figure 2.7b), the behaviour degenerates into a severe "stair-step" convergence profile, where the primary iterate gets sequentially trapped in multiple polyhedral corners. Most critically, in the high-dimensional setting (Figure 2.7c), the standard method completely exhausts its iteration budget during an early stall, failing to make any meaningful spatial progress toward $w^\star$.

Conversely, as seen across all three subfigures, Algorithm 1 eliminates the flat plateaus caused by stalling. In both low-dimensional cases (Figures 2.7a–2.7b), it breaks the initial gridlock and drives the spatial error down to high numerical precision in a handful of iterations Because the closed-form calculation proposed in Theorem 2.4.2 relies exclusively on lightweight vector operations, the computational time overhead required to perform this detection and fast-forwarding is virtually negligible.

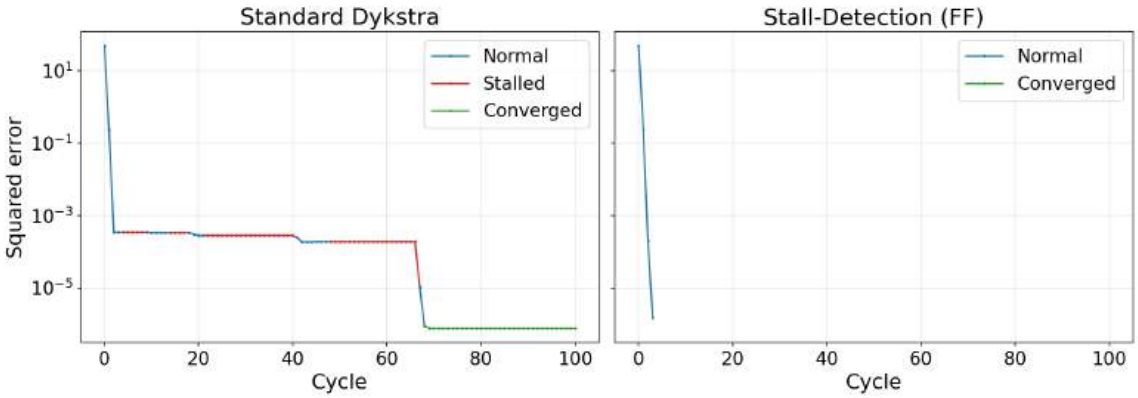
Table 2.2: Runtime and final error across the benchmark environments.

	Standard Dykstra		Algorithm 1 (FF)	
Experiment	Time (s)	Final Error	Time (s)	Final Error
(a)	0.0275	3.01×10^{-4}	0.0134	0.00
(b)	0.0963	7.56×10^{-7}	0.0176	1.50×10^{-7}
(c)	0.2811	2.15×10^{-3}	0.2261	6.00×10^{-10}

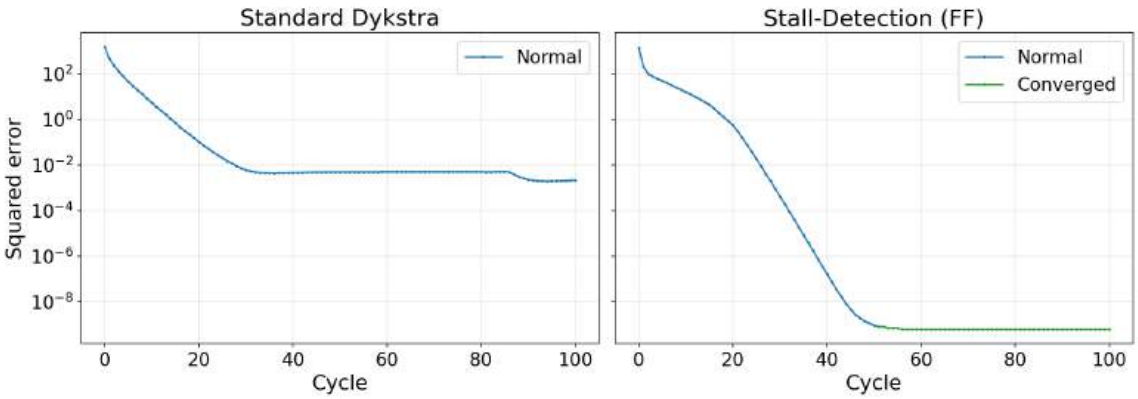
To quantify the direct computational advantages of bypassing stalling, the wall-clock execution time and the terminal squared spatial error $E(w^{(M_{\max})})$ were recorded



(a) Convergence behaviour in three dimensions ($d = 3$, $n = 6$)



(b) Convergence behaviour under a large number of polyhedral constraints ($d = 2$, $n = 20$)



(c) Convergence behaviour in high dimensions ($d = 20$, $n = 50$)

Figure 2.7: Benchmark of Algorithm 1 versus Dykstra across varying geometries.

for each scenario. As detailed in Table 2.2, the fast-forward modification yields significant performance improvements across all tested geometries. In the low-dimensional configuration (a), the execution time is more than halved while driving the spatial error strictly to zero (finite-iteration convergence). Under the dense planar constraints of experiment (b), the solver achieves an 81% reduction in runtime while maintaining comparable terminal precision. Most critically, in the high-dimensional regime (c), the modified algorithm not only reduces the overall computational runtime but also improves the final spatial accuracy by 7 orders of magnitude.

2.7 Summary

Dykstra's algorithm provides a computationally lightweight method for calculating Euclidean projections, yet the unpredictable stalling phenomenon has historically inhibited its deployment in time-critical and high-frequency engineering frameworks [5]. This chapter successfully resolved the stalling problem for polyhedral feasible regions. By formalising the cyclical mechanics of the primary and auxiliary variables during a stall (Section 2.3), we derived an exact, closed-form mathematical solution for its duration (Section 2.4). This derivation directly enabled the construction of a fast-forward modified solver (Section 2.5) that actively monitors the stationarity condition and instantly advances the internal memory variables past the stalling cycles.

As demonstrated by the empirical benchmarks (Section 2.6), this algorithmic modification eliminates computational gridlock across diverse pathological geometries. It systematically reduces wall-clock execution time and accelerates convergence while strictly preserving the asymptotic convergence guarantees of the baseline method (Corollary 2.5.1). The theoretical derivations and optimisation advancements detailed throughout this chapter have been formally published as a preprint as preprint in [21].

With this core mathematical foundation, the fast-forward projection solver is now transformed into a highly predictable, real-time computational engine for computing Euclidean projections. The thesis now pivots from pure optimisation to the applied aerospace domain. The subsequent chapters will leverage this fast-forward architecture to enforce structural constraints within the PGD learning framework, enabling the accurate tracking of highly non-linear satellite uncertainty in the cislunar environment via optimal transport.

Chapter 3

Optimal transport

In this chapter, we formulate a deterministic density estimation framework utilising the lower-triangular Knothe-Rosenblatt rearrangement for optimal transport. By parameterising the map components with a truncated probabilist's Hermite polynomial basis, we reformulate the infinite-dimensional map estimation task into a finite-dimensional, constrained maximum-likelihood problem. The necessity of probability mass conservation dictates a dense system of linear inequality constraints evaluated over an empirical particle ensemble. This monotonicity constraint forms the basis for the adoption of Algorithm 1 in a Projected Gradient Descent architecture.

Notation

We denote the continuous state space as $\mathcal{X} \subseteq \mathbb{R}^D$, where D is the physical spatial dimension. We further denote the set of natural numbers as $\mathbb{N}$, and the set of non-negative integers as $\mathbb{N}_0$. A discrete ensemble of particles drawn from the target measure is denoted by $\{x^{(i)}\}_{i=1}^n$, where n is the total particle count. The deterministic transport map is denoted as $S(x)$, and its individual scalar components are indicated by a superscript as $S^k(x)$. The maximum index of the retained multivariate Hermite polynomial basis for the k-th map component $w^{(k)}$ is denoted as J_k, resulting in J_k+1 scalar coefficients $w_j^{(k)}$. To bridge the notation of this chapter with that of Chapter 2, we explicitly distinguish the physical dimension D from the generic projection dimension d. Because the projection algorithm operates on the polynomial coefficient space rather than the physical state space, the projection dimension is instantiated as $d = J_k + 1$.

3.1 The Knothe-Rosenblatt rearrangement

The fundamental goal of optimal transport is to construct a deterministic coupling between two probability measures. We denote a continuous state space as $\mathcal{X} \subseteq \mathbb{R}^D$. We consider a target probability measure, which is often a complex and non-Gaussian distribution arising from nonlinear physical dynamics. In practice, this target distribution is analytically intractable to evaluate directly. Instead, we assume access to a discrete ensemble of n samples, or particles, denoted by $\{x^{(1)}, \dots, x^{(n)}\} \subset \mathcal{X}$, drawn from this target distribution.

Our overarching objective is to identify a deterministic transport map $S : \mathbb{R}^D \to \mathbb{R}^D$ that pushes the complex target distribution forward to a simple, well-characterised reference distribution. By transforming the particles into a domain where their distribution is known, subsequent analytical tasks in aerospace engineering, such as filtering or uncertainty propagation, become computationally tractable. We fix the reference measure as the standard multivariate normal distribution, $\mathcal{N}(0, I_D)$.

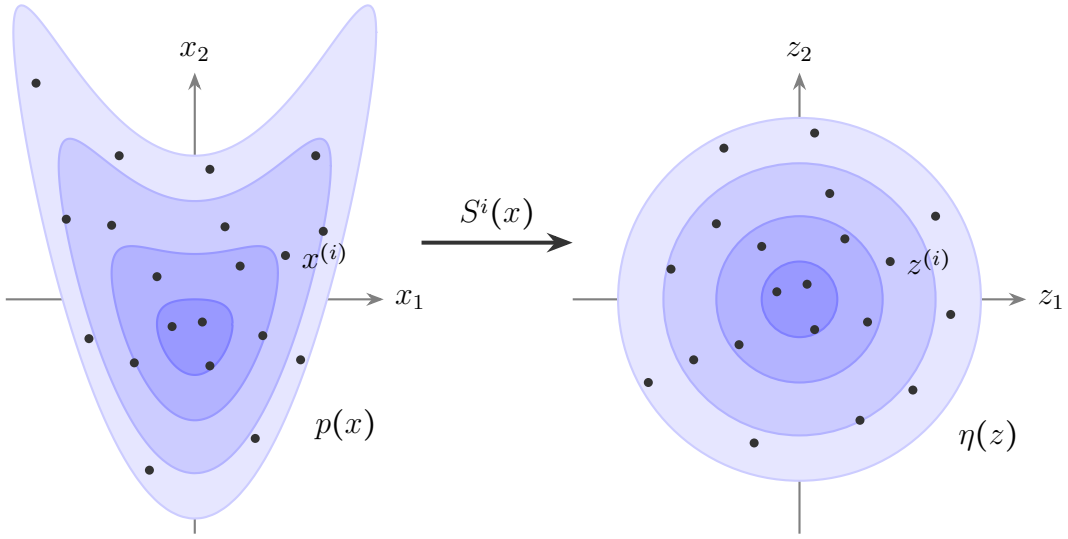


Figure 3.1: A schematic illustrating optimal transport via a deterministic map S.

The problem of finding a valid transport map S such that the pushforward of the target measure equals the reference measure is highly underdetermined; without further constraints, there exist infinitely many such couplings. To render the problem identifiable, we restrict our hypothesis space to the Knothe-Rosenblatt (KR) rearrangement. The KR map is strictly defined by two structural properties: it is lower-triangular, and it is strictly monotone [3]. We partition the map into its scalar

components as

$$S(x) = \begin{bmatrix} S^1(x_1) \\ S^2(x_1, x_2) \\ \vdots \\ S^D(x_1, \ldots, x_d) \end{bmatrix}.$$

To push one continuous measure forward to another while maintaining a valid probability density, the transport map must be a *global diffeomorphism*, a continuously differentiable, bijective mapping. Bijectivity guarantees that the transformation is invertible and that disparate physical states in the target space do not map to the identical coordinate in the reference space. Thus, to ensure the map S conserves probability mass and is invertible, the map must be *monotonic* [3].

The lower-triangular structure ensures that the Jacobian matrix of the map, $\nabla S(x)$, is inherently lower-triangular. Recall that the determinant of a triangular matrix is given by the product of its diagonal entries. The monotonicity constraint thus dictates that each individual component S^k must be a strictly increasing function with respect to its own k-th argument. Using ∂_k to denote the partial derivative with respect to the k-th variable, we require

$$\boxed{\partial_k S^k(x) > 0 \quad \forall k \in \{1, \ldots, d\}}. \tag{3.1.1}$$

To find the optimal map, we frame the task as a density estimation problem. We denote the unknown probability density function of the target distribution as $p(\cdot)$ and the probability density function of the standard normal reference distribution as $\eta(\cdot)$. By the fundamental change of variables formula for probability density functions, the target density evaluated at a spatial coordinate x is given by

$$p(x) = \eta(S(x)) \det \nabla S(x).$$

To construct a maximum likelihood estimator, we evaluate the logarithm of the target density. Taking the natural logarithm of both sides yields

$$\log p(x) = \log \eta(S(x)) + \log \det \nabla S(x).$$

Because the chosen reference distribution is a standard multivariate normal, its d spatial components are mutually independent. The density function is $\eta(z) \propto \exp\left(-\frac{1}{2}\|z\|_2^2\right)$, meaning the log-density is proportional to $-\frac{1}{2}\|S(x)\|_2^2$. Furthermore,

substituting the product of the diagonal elements for the determinant of the Jacobian simplifies the log-determinant term into a summation. Applying these substitutions yields

$$\log p(x) \propto -\frac{1}{2}\|S(x)\|_2^2 + \log\left(\prod_{k=1}^{d} \partial_k S^k(x)\right) = -\frac{1}{2}\|S(x)\|_2^2 + \sum_{k=1}^{d} \log \partial_k S^k(x).$$

Following a *maximum likelihood estimation* approach, we seek to find the map S that maximises the expected log-likelihood of the observed data, which is mathematically equivalent to minimising the negative log-likelihood evaluated over the n empirical particles [3]. Averaging across all particles $x^{(i)}$, and flipping signs, gives the empirical objective function:

$$\min_S \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{2}\|S(x^{(i)})\|_2^2 - \sum_{k=1}^{d} \log \partial_k S^k(x^{(i)})\right).$$

We write the squared ℓ_2-norm explicitly as the sum of its squared components, $\|S(x)\|_2^2 = \sum_{k=1}^{D}(S^k(x))^2$. Substituting and rearranging the order of summation yields:

$$\min_S \sum_{k=1}^{D}\left[\frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{2}(S^k(x^{(i)}))^2 - \log \partial_k S^k(x^{(i)})\right)\right].$$

It is now evident that the objective function completely decouples. The terms associated with any specific map component S^k do not depend on any other component S^j for $j \neq k$. This property allows us to separate the computationally demanding high-dimensional optimisation problem into D independent scalar estimation problems. For any given k-th dimension of the state space, the optimal map component is obtained by solving

$$\boxed{\min_{S^k \in \mathcal{S}_k} \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{2}(S^k(x^{(i)}))^2 - \log \partial_k S^k(x^{(i)})\right)}. \tag{3.1.2}$$

Here, $\mathcal{S}_k$ represents the infinite-dimensional hypothesis space of all functions that satisfy the monotonicity constraint defined in (3.1.1).

The objective function in (3.1.2) balances two simultaneous demands. The quadratic term encourages the mapped particles to cluster near the origin, adopting the unit variance of the standard normal reference. Simultaneously, the logarithmic term severely penalises maps that compress the probability space towards zero, thereby

preventing the formation of singular transformations. The formulation up to this point is continuous and infinite-dimensional; practically, to apply numerical optimisation techniques to learn the map, we must parameterise $\mathcal{S}_k$ using a finite-dimensional polynomial basis.

3.2 Hermite polynomial parameterisation

To solve the independent scalar estimation problems computationally, we must project the infinite-dimensional hypothesis space $\mathcal{S}_k$ into a finite-dimensional functional basis. The choice of basis is critical for the numerical stability of the optimisation routine. Utilising a standard monomial basis to expand the transport map yields severely ill-conditioned linear systems when evaluating expectations against a Gaussian measure, impairing the convergence of gradient-based solvers [3]. Because our objective function inherently assumes the mapped particles follow a standard normal reference distribution, we construct the basis using probabilist's Hermite polynomials. These polynomials are orthogonal with respect to the standard normal probability density function, which smooths the optimisation landscape.

We define the sequence of univariate probabilist's Hermite polynomials, $\mathrm{He}_j(y)$ for a scalar input y (dummy variable) and degree j, via the recurrence relation:

$$\begin{aligned} \mathrm{He}_0(y) &= 1, \\ \mathrm{He}_1(y) &= y, \\ \mathrm{He}_{j+1}(y) &= y\mathrm{He}_j(y) - j\mathrm{He}_{j-1}(y). \end{aligned}$$

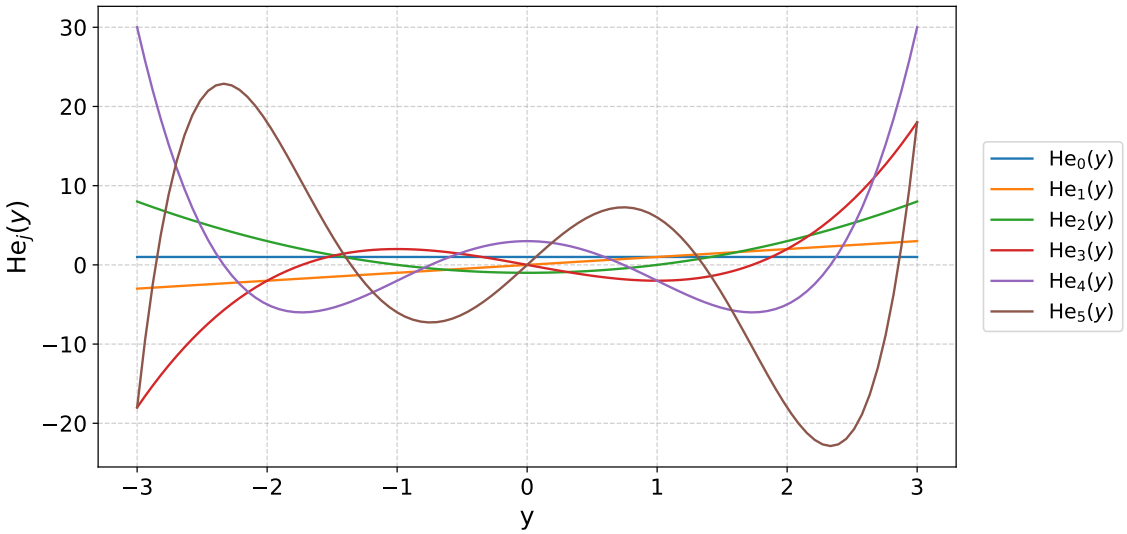


Figure 3.2: The first five Hermite polynomials evaluated over the real line.

The first five of these polynomials are illustrated in Figure 3.2. The polynomials exhibit structural zero-crossings and growth rates that are well-suited for capturing the variations of a probability density [3]. To parameterise the full multivariate transport map, we require multivariate basis functions. For the k-th component of the map, the function depends strictly on the first k spatial coordinates, $x_1, \ldots, x_k$, to enforce the lower-triangular structure of the KR rearrangement.

We construct multivariate basis functions $\Psi_j : \mathbb{R}^k \to \mathbb{R}$, by taking the *tensor product* of the univariate Hermite polynomials across these k active dimensions. We define a multi-index $\alpha^j = (\alpha_1^{(j)}, \ldots, \alpha_k^{(j)}) \in \mathbb{N}_0^k$, where $\mathbb{N}_0^k$ denotes the set of all k-dimensional tuples of non-negative integers, and each integer component $\alpha_m^{(j)}$ dictates the polynomial degree applied to the m-th spatial coordinate. The j-th multivariate basis function is then constructed as the product of the univariate evaluations:

$$\Psi_j(x) = \prod_{m=1}^{k} \mathrm{He}_{\alpha_m^{(j)}}(x_m).$$

3.2.1 Total degree truncation

A full tensor product basis up to a maximum polynomial degree P contains $(P+1)^k$ terms [3]. For high-dimensional state spaces, this exponential growth in decision variables can render the parameterisation computationally burdensome. To alleviate this issue, we enforce *a priori* mathematical sparsity through total degree truncation. We construct a filtered index set where a tensor product term is only retained if the sum of its univariate polynomial degrees is less than or equal to P:

$$\sum_{m=1}^{k} \alpha_m^{(j)} \leq P.$$

This truncation strategy limits the size of the basis while preserving the expressivity required to invert the physical nonlinear shears of the target distribution.

We denote the number of retained basis functions for the k-th map component as $J_k+1 \in \mathbb{N}$. By combinatorial design, following a classical "stars and bars" construction, the number of integer multi-indices satisfying this condition evaluates to the binomial coefficient. Consequently, the total number of retained basis functions for the k-th map component is given by

$$J_k + 1 = \binom{P+k}{k} = \frac{(P+k)!}{P!k!},$$

substantially reducing the dimensionality of the coefficient space compared to a full tensor product. As a practical example, when mapping the sixth spatial dimension ($k = 6$) corresponding to an equinoctial state space with a maximum total degree of $P = 4$, the total number of retained polynomial coefficients evaluates to $J_6 + 1 = \binom{4+6}{6} = \frac{10!}{4!6!} = 210$. Without total degree truncation, a full tensor product basis would have required $(P+1)^k = 5^6 = 15,625$ terms. In practice, the effects of physical shearing in most engineering distributions are predominantly captured by lower-order couplings. The complex, high-degree tensor products discarded by the truncation typically carry negligible physical significance and predominantly contribute to numerical noise and over-parameterisation [3]. Consequently, total degree truncation not only ensures computational tractability but also enforces a necessary, physically motivated sparsity upon the hypothesis space.

3.2.2 Approximating the map

Armed with the truncated multivariate Hermite basis functions Ψ_j, the scalar map component can be approximated as:

$$S^k(x) = \sum_{j=0}^{J_k} w_j^{(k)} \Psi_j(x), \tag{3.2.1}$$

where the vector $w^{(k)} \in \mathbb{R}^{J_k+1}$ contains the unknown scalar coefficients that we must optimise. A direct mathematical advantage of the Hermite parameterisation emerges when evaluating the monotonicity constraint (3.1.1). The derivative of a probabilist's Hermite polynomial obeys the linear relation:

$$\frac{d}{dy}\mathrm{He}_j(y) = j\mathrm{He}_{j-1}(y).$$

Because $\Psi_j(x)$ is a tensor product, the partial derivative ∂_k only operates on the univariate polynomial associated with the k-th spatial coordinate, x_k. Consequently, the partial derivative of the map component reduces to a linear combination of lower-degree Hermite polynomials, evaluated strictly with respect to the decision variables $w^{(k)}$:

$$\partial_k S^k(x) = \sum_{j=0}^{J_k} w_j^{(k)} \partial_k \Psi_j(x). \tag{3.2.2}$$

With this parametrisation, we are ready to reformulate the density estimation task as a finite-dimensional optimisation problem.

3.3 Constrained optimisation formulation

We can substitute the finite expansion (3.2.1) and its linear derivative (3.2.2) directly into the continuous negative log-likelihood (3.1.2). For a specific k-th dimension of the state space, the density estimation task becomes a finite-dimensional optimisation problem over the unknown coefficient vector $w^{(k)} \in \mathbb{R}^{J_k+1}$. We define the resulting empirical objective function as $f : \mathbb{R}^{J_k+1} \to \mathbb{R}$, mathematically given by

$$f(w^{(k)}) = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{1}{2} \left(\sum_{j=0}^{J_k} w_j^{(k)} \Psi_j(x^{(i)}) \right)^2 - \log \left(\sum_{j=0}^{J_k} w_j^{(k)} \partial_k \Psi_j(x^{(i)}) \right) \right). \tag{3.3.1}$$

Because the quadratic penalty is strictly convex and the negative logarithm is strictly convex over its positive domain, the composite objective function $f(\cdot)$ is strictly convex with respect to the decision variables $w^{(k)}$.

To ensure the map remains a global diffeomorphism, we impose the monotonicity constraint (3.1.1). Because we only possess empirical access to the target distribution via the n discrete samples, we enforce this constraint exactly at every observed particle. Furthermore, to ensure numerical stability and avoid boundary cases that could momentarily fold the probability mass during the iterative learning process, we introduce a small positive monotonicity tolerance parameter $\varepsilon_m > 0$. The monotonicity condition evaluated at an arbitrary particle $x^{(i)}$ therefore becomes

$$\sum_{j=0}^{J_k} w_j^{(k)} \partial_k \Psi_j(x^{(i)}) \geq \varepsilon_m. \tag{3.3.2}$$

By exploiting the linearity of the derivative operator across the polynomial basis, the condition in (3.3.2) constitutes a purely linear inequality constraint on the decision variables. By evaluating these basis derivatives at every particle $x^{(i)}$ in the ensemble and stacking the results, we construct a dense constraint matrix $A \in \mathbb{R}^{n \times (J_k+1)}$. We define a vector $\epsilon_m \in \mathbb{R}^n$ where every entry is equal to the small positive scalar ε_m. This transforms the n distinct physical constraints into a unified matrix inequality:

$$A w^{(k)} \geq \epsilon_m \quad \Longrightarrow \quad -A w^{(k)} \leq -\epsilon_m.$$

Geometrically, each row of the matrix $-A$ defines a distinct closed half-space for a corresponding particle $x^{(i)}$ upon which the monotonicity constraint (3.3.2) is evaluated. The truncated Hermite polynomial degree $J_k + 1$ is practically chosen high enough to model the effects of the physical distribution shift. For consistency with Chapter 2 and ease of notation, we will hereafter denote this degree as $d = J_k + 1$ (not to be confused with D, the physical dimension of particles $\{x^{(i)}\}_{i=1}^{n}$). The feasible region for the optimal map coefficients is thus defined by the intersection of n half-spaces, forming a dense polyhedral set, denoted as in Chapter 2 as:

$$\mathcal{H} = \left\{ w \in \mathbb{R}^d \mid -Aw \leq -\epsilon_m \right\}. \tag{3.3.3}$$

3.4 Summary

This chapter outlined the mathematical foundation of the optimal transport framework adopted in this thesis. Our overarching objective is to find the coefficient vector that minimises the objective $f(w^{(k)})$ defined in (3.3.1) subject to constraint $w^{(k)} \in \mathcal{H}$.

3.4.1 Connection with Dykstra's algorithm

To navigate this constrained landscape, we implement a *projected gradient descent* (PGD) framework. In this approach (which we will treat in more detail in Chapter 4), we compute the gradient of the objective and take an unconstrained descent step, resulting in an intermediate, unconstrained point which we denote w°.

Since the unconstrained point w° will frequently violate the monotonicity conditions (i.e., $w^{\circ} \notin \mathcal{H}$), we must project it back onto the feasible polyhedral intersection at each PGD iteration. This projection must alter the learned coefficients as minimally as possible to retain the maximum likelihood data. Therefore, the projection operation requires us to find the unique point $w^{\star} \in \mathcal{H}$ that minimises the Euclidean distance to the unconstrained point w°. Mathematically, this is equivalent to solving the following CQP:

$$\begin{aligned} \underset{w \in \mathbb{R}^d}{\text{minimise}} \quad & \tfrac{1}{2}\|w - w^{\circ}\|_2^2 \\ \text{subject to} \quad & -Aw \leq -\epsilon_m. \end{aligned} \tag{3.4.1}$$

This formulation is equivalent to 2.1.1. Solving this Euclidean projection problem at each PGD iteration bridges the optimal transport framework defined in this chapter and the fast-forward Dykstra algorithm (Algorithm 1) derived in Chapter 2.

Chapter 4

A highly scalable inexact projection framework

Finding optimal transport map components requires navigating a high-dimensional polynomial coefficient space to minimise the empirical log-likelihood subject to polyhedral monotonicity constraints. To solve this constrained problem efficiently across large particle sets, we introduce a highly scalable, inexact Projected Gradient Descent (PGD) architecture. The framework integrates stochastic gradient evaluation to navigate complex objective landscapes, iterative hard thresholding to enforce structural sparsity, and dynamic projection scheduling to circumvent the prohibitive computational costs of exact Euclidean projections.

Notation

We denote set cardinality as $|\cdot|$. The Hadamard product, denoted by the operator $\odot$, represents the element-wise multiplication of two vectors or matrices of identical dimensions. The outer learning PGD loop is indexed by $t \in \{0, 1, \ldots, T-1\}$, where $T-1$ is the outer loop iteration budget, while the inner Dykstra projection cycle is indexed by m and constrained by a dynamic iteration ceiling $M^{(t)}$, which scales according to a base budget M_{base} and an inexact growth exponent p. The stochastic gradient $g^{(t)}$ is evaluated over a random subset of particle indices $\mathcal{B}^{(t)} \subset \{1, 2, \ldots, n\}$ with cardinality B. The learning rate is denoted as $\eta^{(t)}$, and decays according to an initial rate η° and a decay parameter γ. For the iterative hard thresholding procedure, we define a boolean active mask vector $\mathcal{M}^{(t)}$.

4.1 Outer loop projected gradient descent

The density estimation framework formulated in Section 3.3 requires the minimisation of the convex empirical objective $f(\cdot)$ defined in (3.3.1), subject to the polyhedral feasible set $\mathcal{H}$ defined in (3.3.3). To navigate this constrained polynomial coefficient space and compute the optimal transport map, we deploy a PGD architecture. The PGD algorithm operates an outer learning loop, indexed by t, which iteratively updates the decision variables governing the map. Each outer iteration comprises two sequential mathematical operations: an unconstrained descent step evaluated over the target distribution, followed by an exact Euclidean projection to enforce physical monotonicity.

In the first operation, we compute the gradient of the objective function evaluated at the current feasible iterate, $\nabla f(w^{(t)})$. The algorithm takes an unconstrained step in the direction of the negative gradient, scaled by a *step size* (or *learning rate*) parameter $\eta^{(t)}$. This operation produces an intermediate, potentially infeasible coefficient vector w°:

$$w^{\circ} = w^{(t)} - \eta^{(t)} \nabla f(w^{(t)}). \tag{4.1.1}$$

The learning rate schedule follows an inverse time decay model $\eta^{(t)} = \eta^{\circ}/(1 + \gamma t)$, where $\eta^{\circ} \in \mathbb{R}$ is the initial learning rate and $\gamma \in \mathbb{R}$ is a decay coefficient. This decay mechanism helps the optimiser settle into narrow minima as the gradients decrease in magnitude during later iterations. This inverse model is visualised in Figure 4.1, evaluated with an initial learning rate of $\eta^{\circ} = 0.1$ across varying decay coefficients γ.

The selection of the decay coefficient γ strictly governs the exploration-exploitation trade-off during the unconstrained descent phase. A value of $\gamma = 0$ reduces the schedule to a constant learning rate. While a constant step size promotes continuous exploration of the objective landscape, it frequently induces numerical oscillations around the optimal parameters and prevents the algorithm from converging to a precise local minimum. Conversely, an aggressive decay rate (e.g., $\gamma \geq 0.1$) rapidly diminishes the step size. This forces the optimiser to exploit the local geometry and settle quickly, but risks premature convergence to shallow, sub-optimal minima if deployed before the iterates reach the primary basin of attraction. A moderate decay provides a structural balance, allowing the framework to traverse flat ravines during the early learning iterations while ensuring the step size becomes sufficiently small to satisfy strict polyhedral feasibility constraints as t approaches $T - 1$.

Because this unconstrained step is driven solely by the maximum likelihood objective, it does not guarantee adherence to the structural constraints. The intermediate

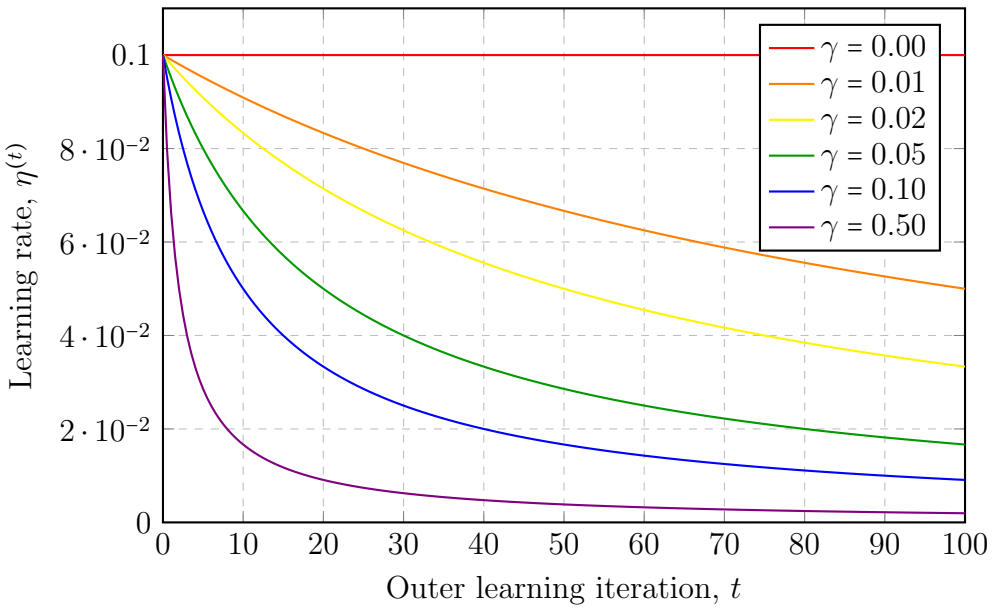


Figure 4.1: Inverse time decay schedule for the unconstrained learning rate.

vector w° will frequently exit the feasible polyhedral intersection $\mathcal{H}$, corresponding to a transport map that temporarily violates the monotonicity requirement.

To rectify this and ensure the map remains strictly monotonic at every learning iteration, the second operation requires projecting w° back onto $\mathcal{H}$. This requires finding the unique feasible point that minimises the Euclidean distance to w°, which is mathematically equivalent to solving the CQP defined in (3.4.1). We denote the exact Euclidean projection operator onto the set $\mathcal{H}$ as $\mathcal{P}_{\mathcal{H}}(\cdot)$. The projected point becomes the feasible update for the subsequent learning iteration:

$$w^{(t+1)} = \mathcal{P}_{\mathcal{H}}(w^\circ). \tag{4.1.2}$$

Within this framework, the evaluation of the projection operator $\mathcal{P}_{\mathcal{H}}(\cdot)$ acts as a direct function call to the accelerated Dykstra algorithm developed in Chapter 2. The integration of the inner projection solver with this outer learning loop, including the dynamic parameterisation of the projection iteration limits and other computational optimisation techniques adopted, is detailed further in Section 4.4.

The backbone of our PGD implementation therefore consists of applying the unconstrained gradient descent update (4.1.1) followed by the projection step (4.1.2). However, evaluating the deterministic gradient over the complete particle set at every iteration is computationally expensive and leaves the optimiser susceptible to "slowing down" in flat ravines within the objective landscape. To address these limitations and

scale the framework, we replace the full-batch gradient evaluation with a stochastic approximation.

4.2 Stochastic gradient descent and minibatching

In a traditional gradient descent framework, evaluating the exact gradient requires computing $\nabla f_i(w)$ for every index within the full particle set $\mathcal{I} = \{1, 2, \ldots, n\}$. When n is large, this full-batch computation introduces a severe computational bottleneck. To mitigate these computational and geometric challenges, we replace the full-batch gradient with a stochastic approximation. This technique is fundamentally rooted in the stochastic approximation methods developed by Robbins and Monro [22] and has been extensively adapted for large-scale machine learning applications [23]. By evaluating the gradient on a random subset of the data, Stochastic Gradient Descent (SGD) introduces inherent variance into the parameter updates. This stochastic noise disrupts deterministic "slowing-down", providing the iterates with the requisite momentum to escape near-flat ravines.

Mathematically, we define a mini-batch subset $\mathcal{B}^{(t)} \subset \mathcal{I}$, which is sampled uniformly at random without replacement at each outer learning iteration t. The cardinality of this subset is the mini-batch size $B = |\mathcal{B}^{(t)}|$, chosen such that $B \ll n$. By decomposing the primary objective function into individual particle contributions, $f_i(w)$, we compute the stochastic gradient $g^{(t)}$ as the sample mean over the mini-batch:

$$g^{(t)} = \frac{1}{B} \sum_{i \in \mathcal{B}^{(t)}} \nabla f_i(w^{(t)}).$$

The vector $g^{(t)}$ serves as an unbiased estimator of the true full-batch gradient and replaces the deterministic term $\nabla f(w^{(t)})$ during the unconstrained descent phase.

While stochastic evaluation accelerates the unconstrained descent phase, we isolate this mini-batching logic from the inner projection loop. If we were to apply mini-batching to the projection operation, Dykstra's algorithm would only evaluate the subset of half-space constraints corresponding to the sampled particles in $\mathcal{B}^{(t)}$. Consequently, the intermediate projection would enforce $-A_{\mathcal{B}^{(t)}} w \leq -\epsilon_{m,\mathcal{B}^{(t)}}$ rather than the complete polyhedral intersection $\mathcal{H}$. The unconstrained step explores the parameter space stochastically using B samples, while the Euclidean projection $\mathcal{P}_{\mathcal{H}}(\cdot)$ enforces the global geometry of all n half-spaces. This dual approach ensures computational scalability while guaranteeing the transport map remains a valid physical diffeomorphism at every iteration.

4.3 Iterative hard thresholding

As the spatial dimension of the physical system increases, the number of polynomial coefficients parameterising the Knothe-Rosenblatt map scales rapidly. Although the total degree truncation introduced in Chapter 3 provides *a priori* structural sparsity, the continuous optimisation process will nonetheless assign non-zero yet values to basis terms that possess no physical significance. Over thousands of learning iterations, these numerical artifacts accumulate, degrading computational efficiency and generalisation.

To enforce physical sparsity during the training process, we incorporate Iterative Hard Thresholding (IHT) into the outer learning loop. IHT functions as a periodic filtering mechanism that evaluates the magnitude of the coefficient iterates and forcibly zeroes out any terms falling below a designated scalar tolerance. Once a coefficient is pruned, it is permanently locked at zero for the remainder of the optimisation.

Mathematically, we introduce a pruning threshold $\epsilon_p > 0$ and a pruning interval $K_{\text{IHT}} \in \mathbb{N}$, ensuring the filtering operation only occurs every K_{IHT} iterations to allow the weights time to settle. We define a boolean active mask vector $\mathcal{M}^{(t)} \in \{0,1\}^d$, initialised as a vector of ones, $\mathcal{M}^{(0)} = \mathbf{1}$. At iteration t, if $t \pmod{K_{\text{IHT}}} \equiv 0$, the mask is updated via element-wise multiplication (Hadamard product) with an indicator function:

$$\mathcal{M}^{(t)} = \mathcal{M}^{(t-1)} \odot \mathbb{I}(|w^{(t)}| \geq \epsilon_p)$$

The mask $\mathcal{M}^{(t)}$ is immediately applied to the current weights $w^{(t)}$ via the Hadamard product to enforce the threshold. For our d-dimensional coefficient vector $w^{(t)}$ and boolean mask $\mathcal{M}^{(t)}$, this operation is defined component-wise as

$$[w^{(t)} \odot \mathcal{M}^{(t)}]_i = w_i^{(t)} \mathcal{M}_i^{(t)} \quad i = 1, 2, \ldots, d.$$

This linear algebraic operation ensures that if the i-th component of the mask evaluates to zero, the corresponding parameterised dimension in the coefficient space is collapsed to zero. To prevent the optimiser from attempting to descend along these pruned dimensions, the stochastic gradient $g^{(t)}$ is similarly masked before the unconstrained descent step. Furthermore, because the projection operator $\mathcal{P}_{\mathcal{H}}(\cdot)$ searches the full d-dimensional space to satisfy the constraints, the output of the Dykstra algorithm must be re-masked to prevent the projection from reactivating pruned coefficients. To prevent the optimiser from attempting to descend along pruned dimensions, the stochastic gradient $g^{(t)}$ is also masked prior to the unconstrained

descent step. Furthermore, because the projection operator $\mathcal{P}_{\mathcal{H}}(\cdot)$ searches the full d-dimensional space to satisfy the constraints, the output of the Dykstra algorithm must be re-masked to prevent the projection from reactivating pruned coefficients.

Because the target particle coordinates are fixed during the estimation of a specific map component, the projection variables A and ϵ_m are pre-computed before the initialisation of the PGD framework, permanently defining the static polyhedral set $\mathcal{H}$ onto which every unconstrained step w° must be projected.

Algorithm 2 unifies the learning rate decay, mini-batch gradient evaluation, and IHT masking into the complete architecture governing the PGD outer loop.

Algorithm 2 Inexact Projected Gradient Descent Framework

1: **Input:** Initial weights $w^{(0)} \in \mathbb{R}^d$, objective $f(w)$, constraint matrix $A \in \mathbb{R}^{n\times d}$, monotonicity tolerance $\epsilon_m \in \mathbb{R}^n$, initial learning rate η°, decay rate γ, max outer iteration budget T, batch size B, pruning threshold ϵ_p, prune interval K_{IHT}, base Dykstra budget M_{base}, Dykstra scaling exponent p

2: $\mathcal{M}^{(0)} \leftarrow \mathbf{1}_d$ ▷ Initialise active mask

3: **for** $t = 0, 1, \ldots, T-1$ **do**

4: $\quad \eta^{(t)} \leftarrow \eta^\circ/(1+\gamma t)$ ▷ Calculate decayed learning rate

5: $\quad$ Sample random mini-batch $\mathcal{B}^{(t)} \subset \{1, \ldots, n\}$ of size B

6: $\quad g^{(t)} \leftarrow \frac{1}{B}\sum_{i\in\mathcal{B}^{(t)}} \nabla f_i(w^{(t)})$ ▷ Evaluate stochastic gradient

7: $\quad$ **if** $t > 0$ **and** $t \pmod{K_{\text{IHT}}} \equiv 0$ **then**

8: $\quad\quad \mathcal{M}^{(t)} \leftarrow \mathcal{M}^{(t-1)} \odot \mathbb{I}(|w^{(t)}| \geq \epsilon_p)$ ▷ Update active mask

9: $\quad$ **else**

10: $\quad\quad \mathcal{M}^{(t)} \leftarrow \mathcal{M}^{(t-1)}$ ▷ Retain current mask

11: $\quad w^{(t)} \leftarrow w^{(t)} \odot \mathcal{M}^{(t)}$ ▷ Prune thresholded weights

12: $\quad g^{(t)} \leftarrow g^{(t)} \odot \mathcal{M}^{(t)}$ ▷ Mask gradient directions

13: $\quad w^\circ \leftarrow w^{(t)} - \eta^{(t)} g^{(t)}$ ▷ Take unconstrained step

14: $\quad w^{(t+1)} \leftarrow \text{Dykstra}(t, w^\circ, -A, -\epsilon_m, M_{\text{base}}, p)$ ▷ Project onto feasible set $\mathcal{H}$

15: $\quad w^{(t+1)} \leftarrow w^{(t+1)} \odot \mathcal{M}^{(t)}$ ▷ Enforce sparsity on projection

16: **Return** $w^{(T)}$

Section 4.4 will detail the implementation of line 14, the inner loop projection.

In addition to the core mechanisms outlined in Algorithm 2, the framework in-

cludes optional safeguards to maintain numerical stability during complex training scenarios. Highly nonlinear target distributions can occasionally produce anomalous gradient evaluations within a given mini-batch. To prevent these from destabilising the unconstrained descent phase, we implement element-wise gradient clipping. This operation bounds the magnitude of the stochastic gradient $g^{(t)}$ to a predefined scalar threshold before the descent step is taken, ensuring the parameter updates remain within a controlled numerical envelope. Furthermore, earlier versions of this architecture incorporated ℓ_1 regularisation directly into the objective function. By adding a penalty term proportional to the absolute magnitude of the coefficients, this approach sought to suppress disproportionately large weights and encourage sparsity. However, empirical testing demonstrated that the explicit boolean masking provided by iterative hard thresholding achieves the necessary sparsity with greater computational efficiency and structural control. Consequently, ℓ_1 regularisation is omitted from the primary algorithm and is reserved exclusively for degenerate cases where the coefficient landscape remains unstable despite gradient clipping.

4.4 Inner loop integration

The projected gradient descent architecture established in Algorithm 2 relies on the Euclidean projection operator $\mathcal{P}_{\mathcal{H}}(\cdot)$ to enforce the physical monotonicity of the KR map. Evaluating this projection requires computing an infinite sequence of Dykstra iterations. In a practical software implementation, the inner loop must be truncated. However, applying a static iteration ceiling to the Dykstra algorithm throughout the entire training process introduces computational inefficiencies. To scale the framework to higher-dimensional tensor bases and larger particle sets, we implement a highly optimised, inexact projection solver.

4.4.1 Dynamic iteration scheduling

During the early stages of the outer learning loop (when t is small), the unconstrained iterate w° is far from the optimal map parameterisation, and the gradient descent step size $\eta^{(t)}$ is large. Computing a high-precision projection at this stage wastes computational resources, as the subsequent unconstrained step will immediately displace the coefficients by a large margin. Conversely, as t approaches the maximum outer iteration budget $T-1$ and the optimiser settles into a narrow minimum, high-precision projections become necessary to guarantee the final map is a valid diffeomorphism.

To balance precision and computational cost, we dynamically scale the inner Dykstra iteration budget based on the outer learning iteration t. We define a base iteration limit $M_{\text{base}} \in \mathbb{N}$ and an inexact growth exponent $p \geq 1$. At outer iteration t, the maximum number of full Dykstra cycles (where one cycle consists of n individual half-space projections) is given by the schedule:

$$M^{(t)} = \lfloor M_{\text{base}}(t+1)^p \rfloor. \tag{4.4.1}$$

The dynamic iteration profile is visualised in Figure 4.2. This polynomial growth schedule ensures that the early gradient steps rely on cheap, approximate projections, while the later iterations enforce feasibility more aggressively.

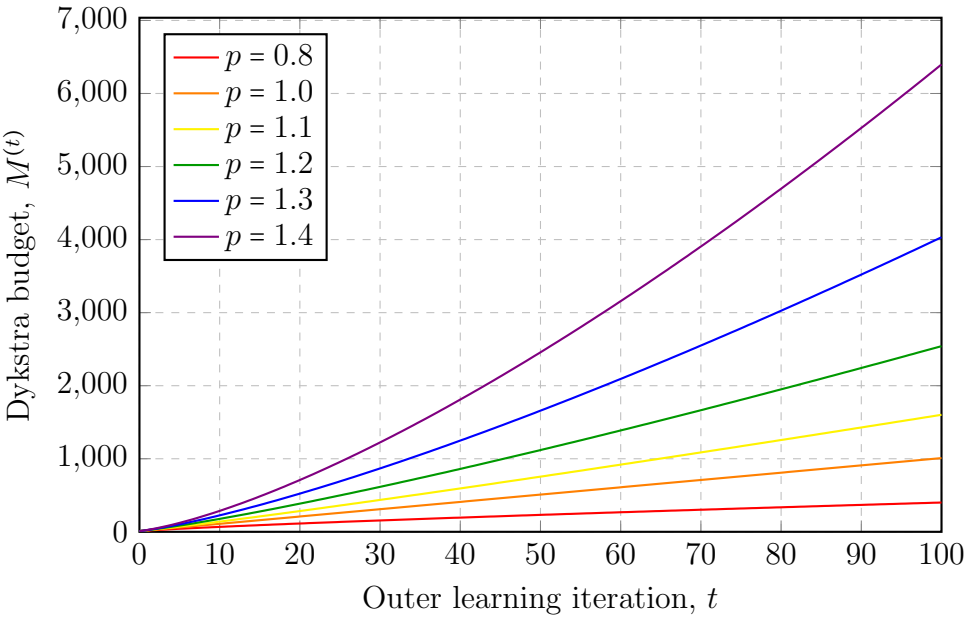


Figure 4.2: Dynamic inner Dykstra budget over the outer learning iterations.

The selection of the scaling exponent p defines distinct computational regimes that must correspond to the convergence rate of the underlying Dykstra projection. When the constraint geometries are poorly conditioned, characterised by sharp geometric angles between intersecting half-spaces, the Dykstra iterates converge slowly to the true projection. In these scenarios, a super-linear exponent ($p > 1$) is required to ensure the dynamic budget outpaces the slow convergence, thereby guaranteeing asymptotic feasibility. Conversely, if the half-space normals defined by $-A$ are nearly parallel, the inner projection step converges quickly in practice. Under such well-conditioned geometry, the rapid algorithmic convergence removes the need for large

iteration ceilings. Deploying a sub-linear exponent ($0 < p < 1$) in this regime yields a concave budget growth that provides a sufficient upper bound to reach the optimum, while strictly bounding the computational expenditure per outer iteration.

4.4.2 Inactive constraint removal

A standard Dykstra cycle iterates through all n half-spaces sequentially. As discussed in Chapter 2, the feasible polyhedral intersection $\mathcal{H}$ can be partitioned into an active subset $\mathcal{A}$ and an inactive subset $\bar{\mathcal{A}}$ (see (2.2.7)). From [14, Lemma 3.1], it is established that for indices belonging to $\bar{\mathcal{A}}$, there exists a finite integer iteration threshold beyond which the half-space is permanently inactive. For an inactive half-space $i \in \bar{\mathcal{A}}$, the Euclidean projection acts as the identity mapping, yielding $w^{(m)} = w^{(m-1)}$ and leaving the auxiliary error term strictly zero, $e_i^{(m)} = 0$. Continually evaluating the geometric boundary conditions for these inactive constraints contributes to significant computational overhead.

To eliminate this overhead, we implement *dynamic half-space removal*. We initialise an active tracking set $\mathcal{I} = \{1, 2, \ldots, n\}$. During each projection cycle, if the unconstrained intermediate point satisfies the constraint strictly (i.e., it lies in the interior $\operatorname{int} \mathcal{H}_i$) and the corresponding auxiliary error e_i is zero, we deduce the constraint has moved into the inactive set. The index i is subsequently popped from $\mathcal{I}$. For all remaining cycles in the current outer iteration, the Dykstra solver bypasses the removed indices, effectively shrinking the length of a full cycle from n to $|\mathcal{A}|$.

4.4.3 Feasibility detection and early termination

While the schedule in (4.4.1) provides a ceiling for the iteration budget, the sequence of projections will sometimes find a physically valid coefficient vector before hitting the full budget $M^{(t)}$. Forcing the algorithm to exhaust the budget to find the exact Euclidean limit point is computationally wasteful, particularly because the subsequent outer unconstrained step will immediately displace the coefficients. To prevent redundant cycles, we implement a strict feasibility-based early termination criterion. Rather than monitoring the spatial displacement between consecutive iterates, the solver evaluates whether the current coefficient vector strictly satisfies the complete polyhedral intersection $\mathcal{H}$.

This evaluation is applied at two distinct stages. First, before the projection loop begins, the solver checks the initial unconstrained point w°. If the gradient descent

step happens to land inside the feasible set (satisfying $-Aw^{\circ} \leq -\epsilon_m$), the projection mechanics are bypassed entirely, requiring zero inner iterations. Second, at the conclusion of each full Dykstra cycle, the solver re-evaluates the updated coefficient vector w. The moment the coefficients become strictly monotonic across all spatial boundaries, the inner loop terminates and returns the feasible iterate.

Algorithm 3 summarises the integration of the dynamic budget, constraint pruning, early termination, and the stall fast-forwarding mechanisms derived in Chapter 2.

Algorithm 3 Accelerated Inexact Dykstra Projection

Input: Outer iteration t, unconstrained weights $w^{\circ} \in \mathbb{R}^d$, constraints $-A \in \mathbb{R}^{n \times d}$, tolerances $-\epsilon_m \in \mathbb{R}^n$, base budget M_{base}, scaling exponent p
$M^{(t)} \leftarrow \lfloor M_{\text{base}}(t+1)^p \rfloor$ ▷ Calculate dynamic iteration budget
$w \leftarrow w^{\circ}$
$e_i^{(0)} \leftarrow \mathbf{0}_d \quad \forall i \in \{1, \ldots, n\}$ ▷ Initialise auxiliary variables
$\mathcal{I} \leftarrow \{1, 2, \ldots, n\}$ ▷ Initialise active index set
for $m = 0, 1, \ldots, M^{(t)} - 1$ **do**
    for $i \in \mathcal{I}$ **do**
        $w \leftarrow \mathrm{FF}(w, -A, -\epsilon_m)$ ▷ Stalling fast-forwarding as per Algorithm 1
        $w^{(m)} \leftarrow w$
        $w \leftarrow \mathcal{P}_{\mathcal{H}_i}(w^{(m)} + e_i^{(m)})$ ▷ Project onto individual half-space
        $e_i^{(m+1)} \leftarrow e_i^{(m)} + w^{(m)} - w$ ▷ Update auxiliary error
        if $w = w^{(m)}$ **and** $e_i^{(m+1)} = \mathbf{0}_d$ **then**
            $\mathcal{I} \leftarrow \mathcal{I} \setminus \{i\}$ ▷ Prune inactive half-space
    if $-Aw \leq -\epsilon_m$ **then**
        break ▷ Terminate early on feasibility
Return w

This algorithm acts as the Dykstra$(\cdot)$ function called by the master framework in line 14 of Algorithm 2.

4.5 Summary

This chapter has developed a scalable, inexact projection framework to compute the optimal transport map parameterisation, which is summarised in Algorithms 2–3 and complemented by Algorithm 1 developed in Chapter 2. By integrating stochastic gradient evaluation and iterative hard thresholding into an outer projected gradient descent loop, the architecture navigates flat objective landscapes and enforces structural sparsity on the physical parameters. Furthermore, by dynamically scheduling the inner Dykstra iteration budget and implementing feasibility-based early termination criteria, the framework circumvents the computational overhead associated with continuous exact Euclidean projections.

Chapter 5

Experiments and applications

In this thesis, we have developed a unified mathematical and computational framework for evaluating high-dimensional optimal transport under structural constraints. Chapter 2 presents a theoretical resolution to the stalling phenomenon in Dykstra's algorithm, introducing the fast-forward algorithmic modification. Concurrently, Chapter 3 introduces a density tracking methodology that learns KR triangular maps using a probabilist's Hermite polynomial basis, which is then integrated into the scalable inexact projection framework detailed in Chapter 4. In this chapter, we demonstrate the application of the theoretical framework to several examples to illustrate the effectiveness of our approach.

The numerical experiments are designed to validate the representational capacity of the transport maps and to assess the computational scalability of the custom optimisation solver. The initial evaluations examine the framework's geometric reconstruction capabilities using non-linear synthetic distributions. Subsequently, the analysis focuses on the primary aerospace application, specifically the transformation of cislunar satellite uncertainty parameterised by equinoctial elements. To further assess the methodology in the context of chaotic dynamical systems relevant to geophysics, the framework is evaluated using a dataset emanating from the Lorenz-63 model. Finally, the analysis returns to the core optimisation architecture to quantify the runtime efficiency of the inexact projected gradient descent solver in comparison with several existing solvers. The results of these experiments are reproducible using the second of two repositories accompanying this thesis [17, 18], available at:

ClouD-161803/Optimal-Transport-Dykstra.git.

Numerical setup

For each of the experiments described in this chapter, the numerical setup implements the architecture outlined in Chapters 2–3–4. To initialise the experiment, the state dimension D, number of particles n, and random seed are selected. The weights $w^{(0)}$ are initialised to yield the identity map $S^k(x^{(i)}) = x^{(i)}$. A maximum Hermite polynomial degree $J_k + 1$ is selected. Matrix A and monotonicity tolerance vector ϵ_m are initialised according to (3.3.2). The outer projected gradient descent loop executes for T total iterations to update the decision variables $w^{(t)}$. At each outer loop iteration, the objective is computed according to (3.3.1). To manage the computational demand of evaluating the full set of n particles, stochastic mini-batching is utilised with a subset cardinality B. The learning rate dictates the magnitude of the unconstrained step to w° and is updated at each iteration, starting from an initial value η° and decaying according to the parameter γ. Iterative hard thresholding is applied periodically at every K_{IHT}-th iteration, where non-physical weights are pruned via the boolean active mask vector $\mathcal{M}^{(t)}$ according to the threshold ϵ_p. Within the inner solver cycle, the Dykstra projection budget ceiling $M^{(t)}$ dynamically scales based on the base budget M_{base} and the scaling exponent p. Given the full set of initialisations, we run the combined routine Algorithm 1–Algorithm 2–Algorithm 3.

5.1 Reconstructing synthetic distributions

Before applying the optimal transport methodology to orbital mechanics, it is necessary to isolate and evaluate the representational capacity of the KR map. Testing the framework on synthetic distributions provides a simple and immediate empirical validation, independent of the confounding variables introduced by physical propagation models. By evaluating the methodology on artificial, non-linear geometries, the capacity to capture non-trivial structural deformations can be assessed without relying on external dynamics. Thus, this phase focuses purely on the geometric limits of the mapping procedure.

To generate the synthetic test cases, a known structural deformation is applied to a reference probability measure. The target distribution is represented by a finite set of n unmapped particles, denoted as $\{x^{(i)}\}_{i=1}^n \subset \mathbb{R}^D$, where D denotes the physical state dimension. These particles are generated by applying a non-linear artificial shear mapping, $\tilde{S} : \mathbb{R}^D \to \mathbb{R}^D$, to full state samples drawn from a multivariate standard normal distribution, $\{\tilde{z}^{(i)}\}_{i=1}^n$, where $\tilde{z}^{(i)} \sim \mathcal{N}(0, I_D)$. The shear mapping takes the

form

$$x^{(i)} = \tilde{S}\left(\tilde{z}^{(i)}\right), \quad i = 1, \ldots, n.$$

The reconstruction objective is to approximate the inverse transformation, mapping $S : \mathbb{R}^D \to \mathbb{R}^D$, that transports the sheared particles $\{x^{(i)}\}_{i=1}^n$ back to standard normal distribution. This produces a new set of mapped particles $\{z^{(i)}\}_{i=1}^n$. The success of the reconstruction is evaluated by comparing the empirical distribution of $\{z^{(i)}\}_{i=1}^n$ against the original unsheared particles $\{\tilde{z}^{(i)}\}$, thereby quantifying the ability of the algorithm to invert the artificial shear $\tilde{S}$.

5.1.1 Numerical results

The framework is tested against three distinct synthetic shear models. For these experiments, the state dimension is restricted to $D = 2$, which is chosen for ease of visualisation. We denote the spatial components of the true normal samples as $\tilde{z}^{(i)} = (\tilde{z}_1^{(i)}, \tilde{z}_2^{(i)})^T$ and the resulting sheared target particles as $x^{(i)} = (x_1^{(i)}, x_2^{(i)})^T$.

The first test case introduces a linear shear that pushes the particles on a straight line, applying an off-diagonal structural deformation to the state space. The transformation takes the form

$$\begin{aligned} x_1^{(i)} &= \tilde{z}_1^{(i)} - 0.85\tilde{z}_2^{(i)}, \\ x_2^{(i)} &= \tilde{z}_2^{(i)}. \end{aligned}$$

This shear provides a baseline for testing the framework's capability to invert highly correlated, off-diagonal linear deformations.

The second test case involves a quadratic shear. The mapping is defined as

$$\begin{aligned} x_1^{(i)} &= \tilde{z}_1^{(i)}, \\ x_2^{(i)} &= \tilde{z}_2^{(i)} + \left(\tilde{z}_1^{(i)}\right)^2. \end{aligned}$$

This quadratic warping is a prevalent characteristic of orbital uncertainty propagation over short to medium time horizons, making this synthetic case a rough approximation to the physical aerospace application. This acts as a first-order test to ensure feasibility on the full orbital uncertainty experiment, which we will later discuss in Section 5.2.

The final synthetic experiment employs a cubic shear. The transformation is given by

$$
\begin{aligned}
x_1^{(i)} &= \tilde{z}_1^{(i)}, \\
x_2^{(i)} &= \tilde{z}_2^{(i)} + 0.4\left(\left(\tilde{z}_1^{(i)}\right)^3 - \tilde{z}_1^{(i)}\right).
\end{aligned}
$$

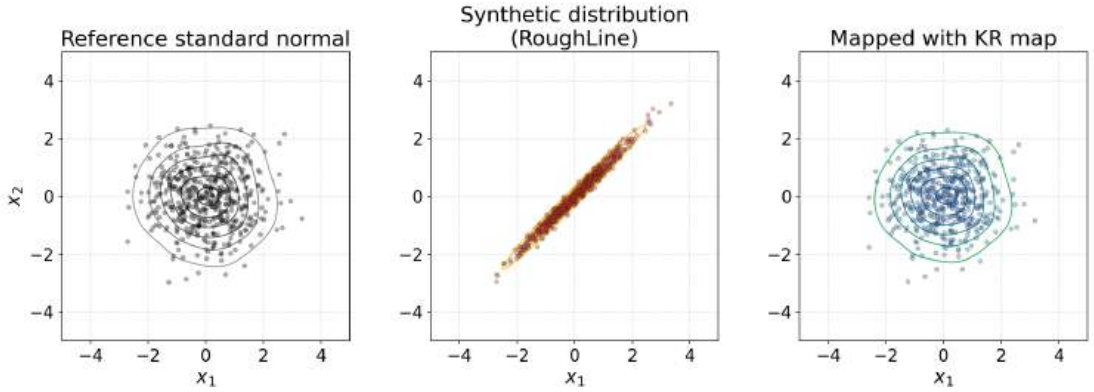
The cubic shear subjects the baseline probability measure to higher-order polynomial deformations, requiring a higher-order polynomial expansion to be modelled, and therefore a higher number of map components to be optimised. This benchmark is therefore aimed at preliminarily testing the scaling capabilities of the optimal transport inexact projection framework.

The specific parameter configurations for each synthetic shear experiment are summarised in Table 5.1. We remark that these parameters were chosen primarily through experience and guidance from literature works, the list of which would be hard to formulate and too long to recount. In addition, some parameter choices were refined through trial and error for each individual experience. We also note that several practical implementation details, such as additional numerical tolerances and specific architectural choices, are omitted from this discussion. This choice is aimed at preventing a verbose explanation of the numerics and keeping the reader focused on the more relevant implementation details.

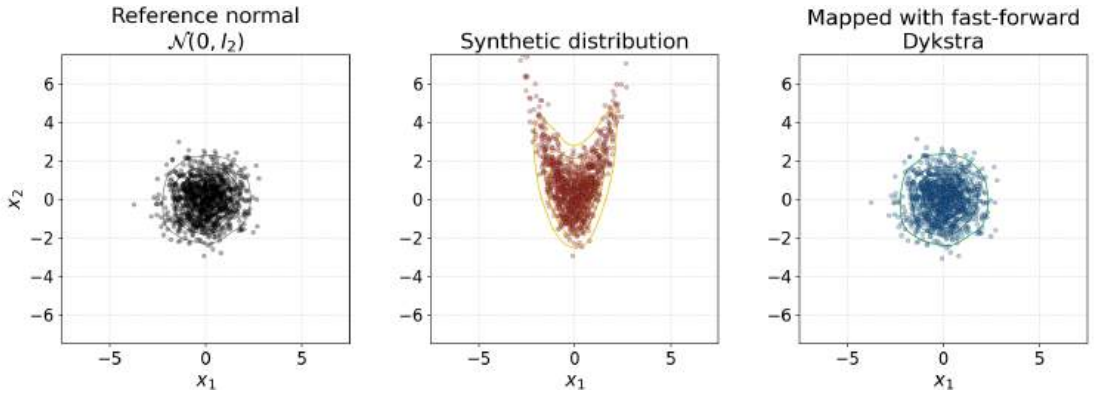
Table 5.1: Optimisation parameters used across the synthetic shear experiments.

Parameter	**Symbol**	**Linear**	**Quadratic**	**Cubic**
Random seed	–	2222	5432	507
State dimension	D	2	2	2
Hermite degree	$J_k + 1$	2	3	5
Particle count	n	500	1000	500
Mini-batch cardinality	B	100	200	100
Outer PGD iterations	T	10000	1000	5000
Dykstra budget	M_{base}	10	10	10
Budget scaling exponent	p	0.5	0.67	0.54
Initial learning rate	η°	0.2	0.1	0.01
Learning rate decay	γ	0.01	0.01	0.01
IHT interval	K_{IHT}	100	100	100
IHT tolerance	ε_p	0.01	0.01	0.01
Monotonicity tolerance	ε_m	1e-12	1e-12	1e-12

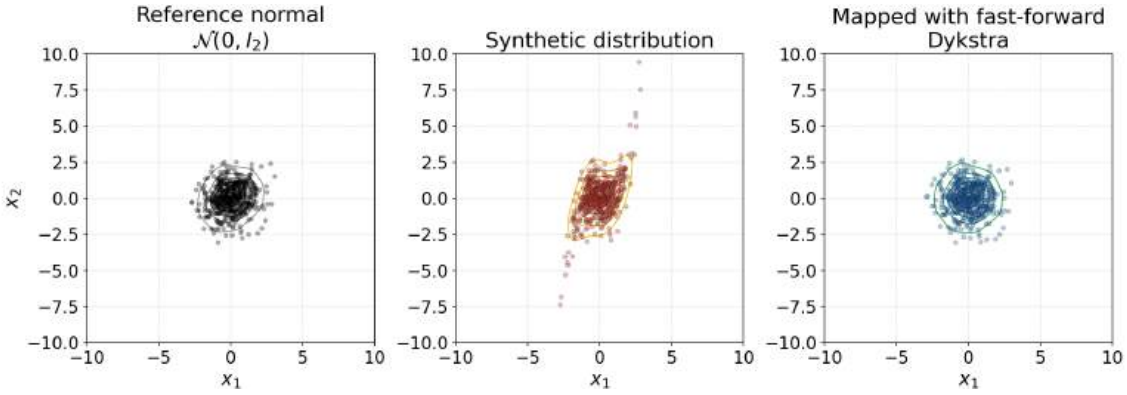
The sheared distributions, as well as final mapped results, are shown in Figure 5.1.



(a) Results for the linear synthetic shear (`SEED` = 2222).



(b) Results for the quadratic synthetic shear (`SEED` = 5432).



(c) Results for the cubic synthetic shear (`SEED` = 507).

Figure 5.1: Reconstruction of the three synthetic shear distributions.

The figure shows the linear, quadratic, and cubic models in Figures 5.1a, 5.1b, 5.1c, respectively. The left-most panels show the unsheared particles $\{\tilde{z}^{(i)}\}$, the center panels the sheared particles $\{x^{(i)}\}$, and the right-most panels the mapped particles $\{z^{(i)}\}$. Distribution contours are also plotted for each panel[1].

Visual evaluation of the reconstructed distribution indicates that the mapped particles accurately reconstruct the reference state. To quantitatively substantiate this geometric recovery, the empirical Wasserstein-2 distance between the mapped empirical distribution and the theoretical standard normal measure was evaluated to below 10^{-2} across all three configurations.

To visualise the temporal evolution of the mapping procedure, the intermediate mapped states are evaluated at distinct intervals during the learning process. Progress plots, displaying the reconstructed distributions across varying outer projected gradient descent iterations, are presented in Figure 5.2, Figure 5.3, and Figure 5.4 for the linear, quadratic, and cubic models, respectively.

Table 5.2 details the empirical execution time required to optimise the individual map components, S^1 and S^2, alongside the cumulative algorithmic runtime, for each synthetic distribution.

Table 5.2: Runtime across the synthetic tests.

		Runtime (s)		
Model	Seed	S^1	S^2	Total
Linear	2222	52.84	55.21	108.05
Quadratic	5432	10.38	26.30	36.68
Cubic	507	44.09	51.19	95.28

The quadratic deformation required the lowest total execution time. In contrast, the linear and cubic models demanded higher computational effort, yielding cumulative runtimes of 108.05 seconds and 95.28 seconds, respectively. Across all configurations, the optimisation of the two-dimensional component S^2 required greater computational duration than the one-dimensional component S^1, as expected.

[1]To visualise the geometric structure of the multidimensional states, iso-density contour lines are generated directly from the empirical particle sets rather than analytical probability density functions. The continuous density field is first approximated over a uniformly discretised spatial grid using either a Gaussian kernel density estimator or a smoothed two-dimensional spatial histogram. The resulting field is then discretised into a finite set of contour levels defined relative to the maximum evaluated density. By linearly spacing these isolines between 10% and 95% of the maximum density, the primary mass of the distribution is consistently captured while filtering low-density boundary artefacts.

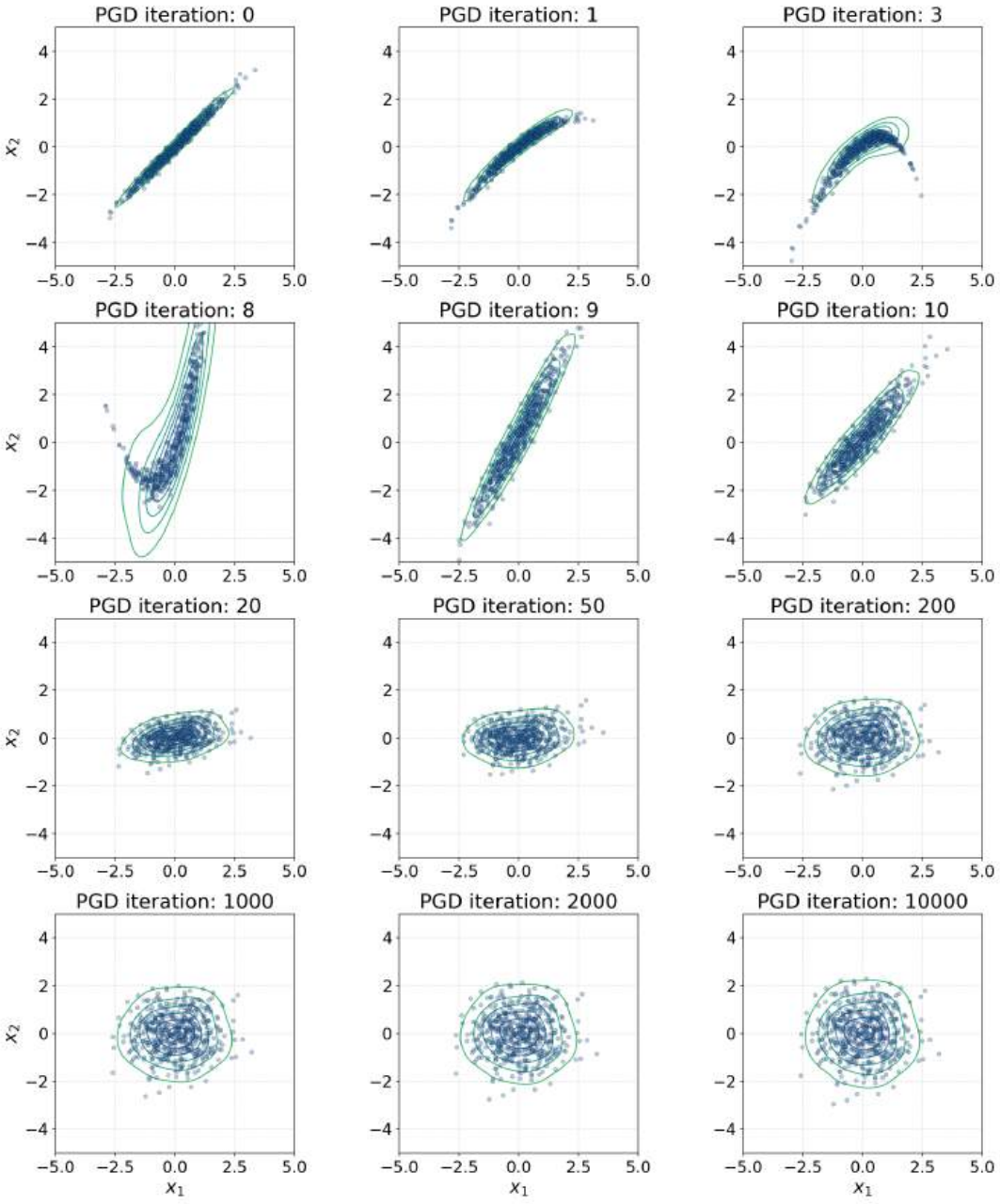


Figure 5.2: Progress plot for the linear experiment (`SEED` = 2222).

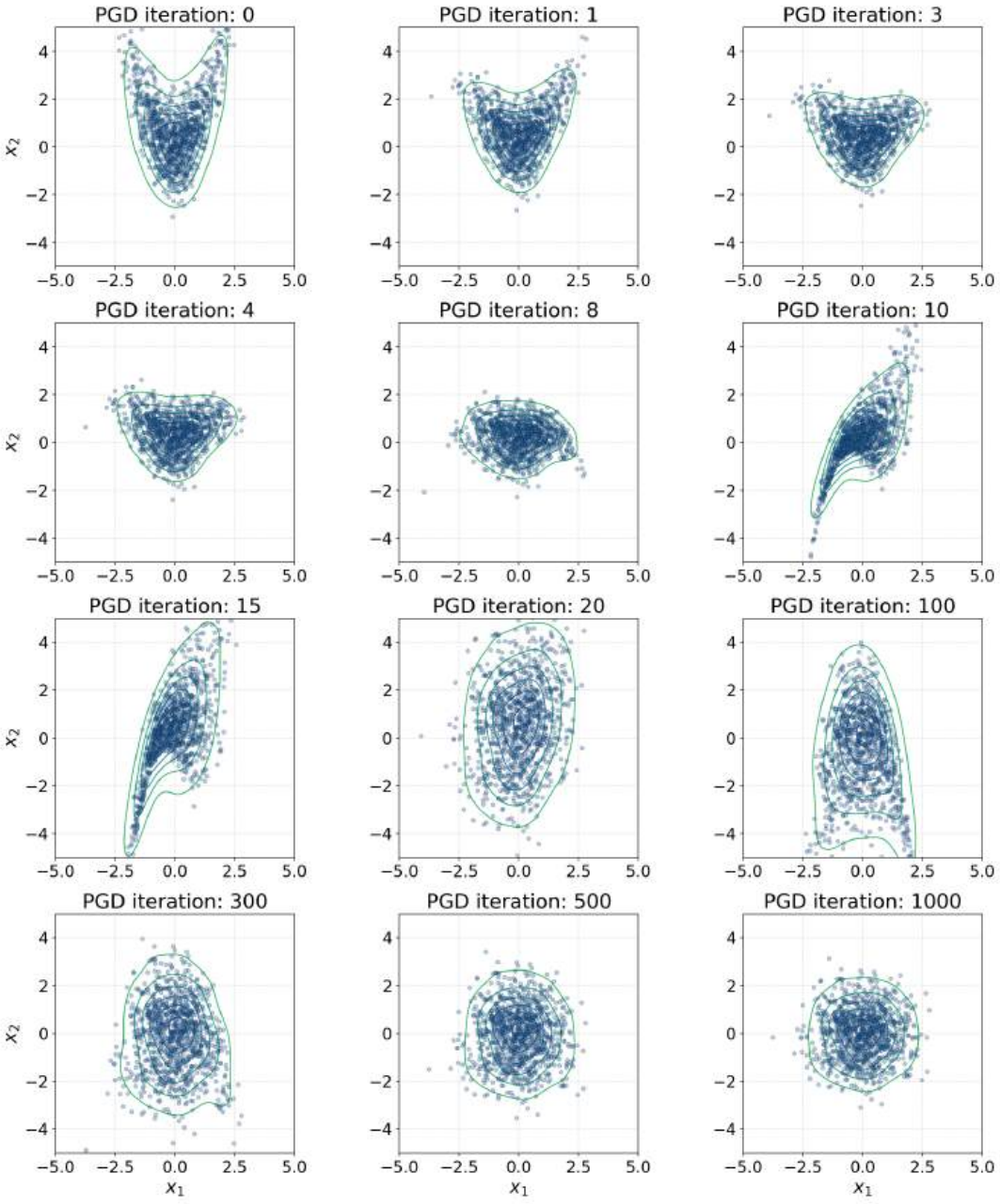


Figure 5.3: Progress plot for the quadratic experiment (`SEED` = 5432).

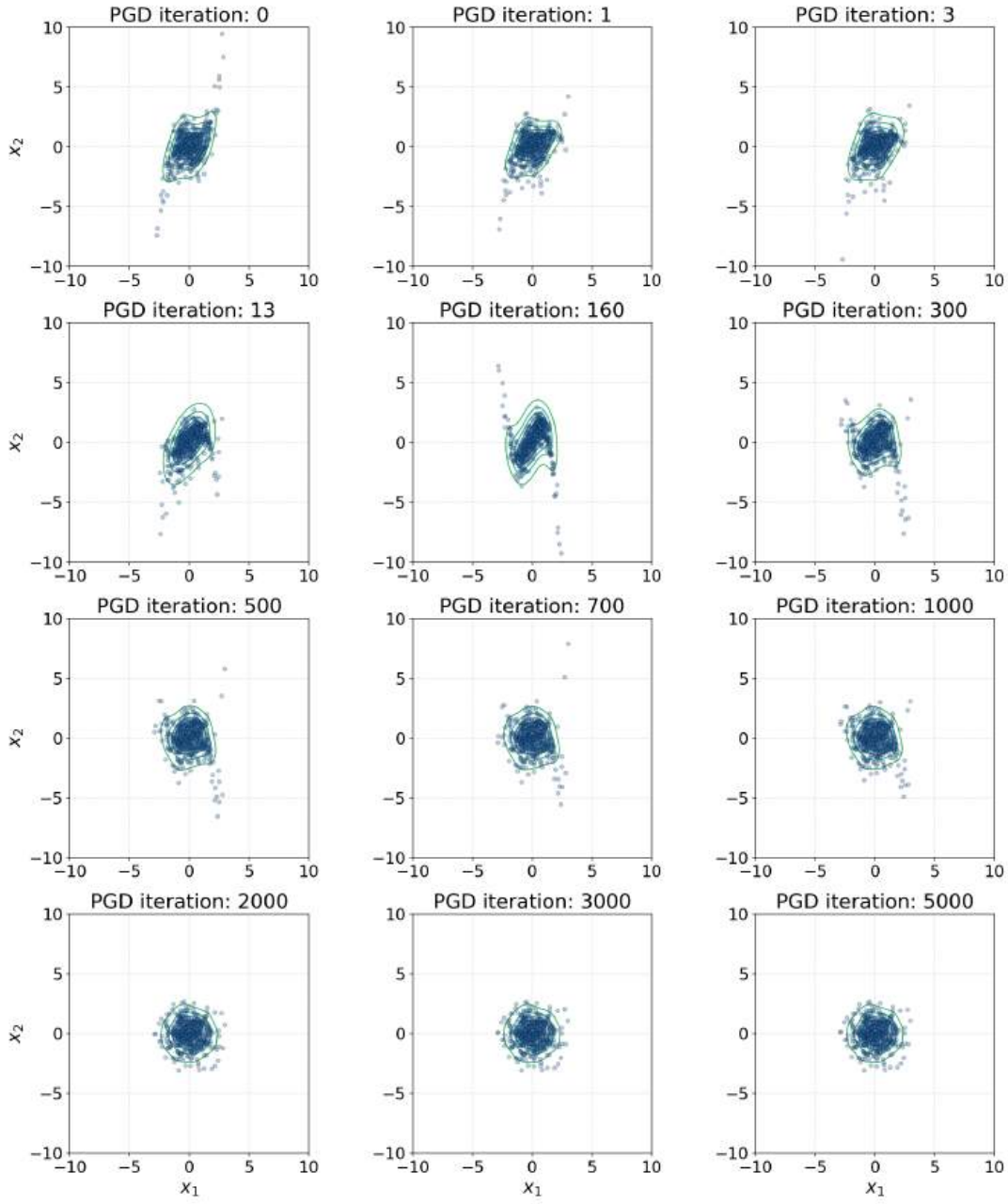


Figure 5.4: Progress plot for the cubic experiment (`SEED` = 507).

5.2 Application to uncertainty propagation

After establishing the geometric features of transport maps on synthetic domains, the analysis focuses on the primary aerospace application: tracking uncertainty in orbital dynamics. The propagation of orbital vectors over time results in nonlinear deformations of the associated probability density. Accurately capturing these structural changes is essential for robust space situational awareness and orbit determination. The subsequent analysis employs the inexact optimal transport framework to approximate the mappings arising from physical orbital perturbations.

5.2.1 Parameterisation of orbital state

The physical state of a satellite in orbit is conventionally defined by a set of parameters. A *minimal set* establishes the smallest number of parameters required to fully identify the state of an object in orbit. For planar two-body orbits, the state information is contained in the satellite's position and velocity vectors, both three-dimensional. Thus, a minimal set of parameters will contain six entries. These orbits, also known as the Keplerian orbits, can take the form of two-dimensional conic sections: either circles, ellipses, parabolas, or hyperbolas. In classical celestial mechanics, the state is parameterised using the Keplerian orbital elements, which provide an intuitive geometric description of the trajectory. These six elements are defined as follows, and are visualised in Figure 5.5 for a Low-Earth orbit satellite:

- a: Semi-major axis, defining the spatial dimension of the orbital ellipse.
- e: Eccentricity, describing the deviation of the orbit from a circular trajectory.
- i: Inclination, representing the vertical tilt of the orbital plane relative to a referential plane.
- Ω: Right ascension of the ascending node, specifying the horizontal orientation of the ascending intersection with the reference plane.
- ω: Argument of periapsis, defining the orientation of the orbit within the orbital plane, relative to the point of closest approach to the main body.
- θ: True anomaly, representing the angular position of the satellite along its trajectory at a specific epoch.

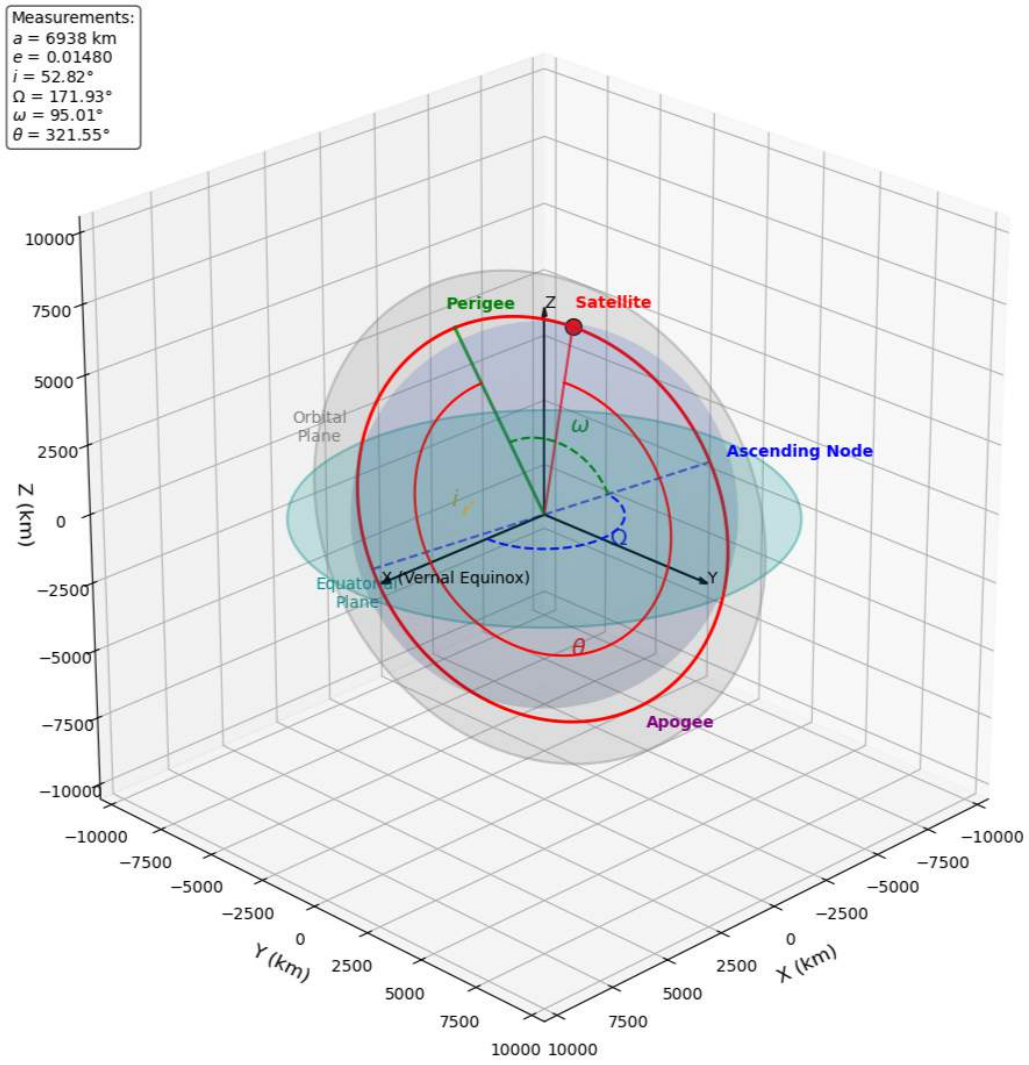


Figure 5.5: Geometric representation of the classical Keplerian orbital elements.

Additionally, the mean anomaly, M, is a mathematically convenient parameter that increases uniformly with time and represents the fraction of the orbital period elapsed since the satellite last passed periapsis. Although Keplerian elements provide geometric clarity, they exhibit mathematical singularities under specific conditions. For circular orbits ($e \to 0$), the argument of periapsis ω becomes undefined. For equatorial orbits ($i \to 0$ or $i \to 180°$), the right ascension of the ascending node Ω is similarly undefined. These singularities introduce numerical instabilities during nonlinear state propagation and density-tracking procedures.

To address these computational limitations, it is standard practice to parameterise the state in terms of *equinoctial elements*. This alternate formulation transforms the classical variables into a representation that remains nonsingular for circular and nonretrograde equatorial orbits. The equinoctial elements map the orbital state to a space defined by $\mathbb{R}^5 \times \mathbb{S}^1$, taking the following form:

- a: Semi-major axis, retained from the classical formulation.
- h: First component of the eccentricity vector, evaluated as $h = e\sin(\omega + \Omega)$.
- k: Second component of the eccentricity vector, evaluated as $k = e\cos(\omega + \Omega)$.
- p: First component of the ascending node vector, evaluated as $p = \tan(i/2)\sin(\Omega)$.
- q: Second component of the ascending node vector, evaluated as $q = \tan(i/2)\cos(\Omega)$.
- λ: Mean longitude, incorporating the mean anomaly as $\lambda = \Omega + \omega + M$.

Tracking the probability distribution of the satellite state in equinoctial element space reduces the risk of algorithmic failure caused by coordinate singularities. In our framework, this parameterisation preserves the structural integrity of the optimal transport map during continuous integration of the physical dynamics.

5.2.2 Filtering

In space situational awareness, maintaining a robust representation of satellite uncertainty requires continuous evaluation of a probability density function under physical dynamics. Although transforming the state space into equinoctial elements mitigates the numerical singularities associated with classical orbital elements, the underlying orbital dynamics and measurement models remain fundamentally nonlinear. Therefore, sequential orbit determination requires *filtering* techniques that can accommodate these nonlinearities. In the context of sequential orbit determination, filtering

is defined as the recursive mathematical procedure for estimating the true physical state and its associated probability density, which operates by continually fusing noisy observational measurements with a predictive propagation model.

The standard approach in the aerospace literature is the Extended Kalman Filter (EKF), which operates by linearising the system dynamics and measurement models about the current state estimate [1]. Although computationally efficient, this method assumes that the probability density function of the state uncertainty remains Gaussian throughout propagation. For typical orbital regimes, over extended propagation intervals, the distribution of uncertainty distributions undergoes nonlinear shearing, as visualised in Figure 1.1 [1]. Consequently, the EKF's uncertainty estimate will be structurally inappropriate for modelling the true probability distribution; at the same time, the filter may become "overconfident" in its estimate, resulting in degraded estimate accuracy and filter divergence.

To avoid explicit Jacobian calculations and reduce linearisation errors, the Unscented Kalman Filter (UKF) is often employed as an alternative to the EKF [1]. Rather than directly linearising the equations, this filter deterministically propagates a minimal set of sample points through the true nonlinear functions to estimate the posterior mean and covariance [1]. However, similarly to the EKF, the UKF assumes a Gaussian distribution. Thus, this filter also fails to represent the ground-truth probability distribution of the state, often assigning the highest probability mass to regions unoccupied by the satellite. This discrepancy results in a loss of realism in uncertainty representation.

When the Gaussian assumption is relaxed, Particle Filters (PF) provide a fully non-parametric alternative. By representing the uncertainty distribution as an ensemble of discrete, weighted Monte Carlo samples, PF methods can approximate arbitrary non-Gaussian geometries. However, standard particle filtering is susceptible to weight degeneracy, particularly in high-dimensional spaces or with sparse measurements, where a single particle accumulates most of the statistical weight, and the remainder are discarded. Advanced control-based techniques, such as the Nudged PF and its variational variants, have been developed to address sparse observations more robustly [24]. Nonetheless, these sophisticated particle-based methods can often impose a computational burden that is prohibitive for real-time space situational awareness.

5.2.3 Mapping Gauss von Mises distributions

To combine the efficiency of analytical methods with the ability to represent the stretching topology of orbital uncertainty, we propose a hybrid approach that augments a Kalman filter with an optimal transport map. In particular, in this section, we demonstrate the effectiveness of the established inexact projection framework for mapping high-fidelity probability distributions to standard normals. We focus our attention on one such family of distributions, the Gauss von Mises (GVM) family [1]. Defined on a cylindrical manifold, this distribution accommodates the nonlinear shearing inherent in long-term orbital propagations, thereby maintaining uncertainty realism over extended prediction horizons. Augmented with our optimal transport-derived deterministic transformation, the Kalman filter can perform its highly efficient measurement update on a theoretically Gaussian state, thereby integrating the high-fidelity orbital mechanics model into a computationally tractable architecture.

The formulation of the GVM distribution arises from the necessity to model uncertainties on the cylindrical manifold $\mathbb{R}^{D-1}\times\mathbb{S}^1$, where, as usual, D represents the full state dimension. As discussed in Section 5.2.1, for orbital element sets parameterising the state of a satellite in orbit, we adopt the equinoctial minimal set, and we thus have $D = 6$. Partitioning the full state vector as $(v^T, \lambda)^T$, where $v \in \mathbb{R}^5$ denotes the linear coordinates (the first five equinoctial elements as defined in 5.2.1), and $\lambda \in \mathbb{S}^1$ represents the bounded periodic angular coordinate (mean longitude). The probability density function of the GVM distribution is defined by coupling a multivariate Gaussian with a von Mises distribution [1]. The joint probability density function is expressed as

$$p(v,\lambda) = \frac{1}{Z}\exp\Bigg(-\frac{1}{2}(v-\mu_v)^T\Sigma_{vv}^{-1}(v-\mu_v) \\ -2\kappa\sin^2\frac{1}{2}\Big[\lambda-\mu_\lambda-\frac{1}{2}(L^{-1}(v-\mu_v))^T\Gamma(L^{-1}(v-\mu_v))\Big]\Bigg), \tag{5.2.1}$$

where μ_v and Σ_{vv} are the mean and covariance of the linear coordinate subset v, L is the lower-triangular Cholesky factor of Σ_{vv} such that $\Sigma_{vv} = LL^T$, μ_λ is the mean of the angular component λ, κ is the concentration parameter defining the angular dispersion, and Γ governs the structural correlation between the linear and periodic coordinates [1]. The term Z ensures the probability mass integrates to unity

over the manifold. As the set of orbital elements is propagated through time, the nonlinear shear is accurately captured by the structural parameters Γ and κ, allowing the probability contours to bend within the periodic domain without violating the topological boundaries.

5.2.4 Numerical results

To assess the capability of the optimal transport framework to invert these orbital uncertainty distributions onto an isotropic domain, we conducted three separate numerical experiments. The common experimental pipeline employs a data generation module that instantiates the n unmapped particles $\{x^{(i)}\}_{i=1}^{n}$ according to the GVM distribution in (5.2.1). For each experiment, we specify a highly correlated covariance structure Σ_{vv}, a low concentration κ, and a dominant correlation vector Γ, to produce discrete state particles exhibiting the nonlinear crescent geometry characteristic of long-term orbital uncertainty propagation. The first experiment, which we denote as (a), aims to mirror the state of the satellite described in [1] after propagation over 8 orbital epochs (see the bottom-right panel in Figure 1.1). The second experiment, denoted as (b), tests an even more extreme shearing case to simulate the effects of long-term propagation of orbital elements. The third experiment, denoted (c), evaluates the framework's robustness under a discontinuous, artificially truncated state distribution. This test was set up by generating the data according to the distribution in (5.2.1), then filtering it according to a hard threshold constraint $\lambda \leq \bar{\lambda}$. In so doing, some particles were removed from the distribution altogether, rendering the optimiser's task to identify the map components S^k significantly harder.

Table 5.3: Distribution parameters for the orbital uncertainty experiments.

Parameter	**Symbol**	**(a)**	**(b)**	**(c)**
Angular concentration	κ	5.2	7.0	5.2
Correlation coefficient	Γ_{11}	2.4	9.0	2.4

These three sets of generated particles constitute the source distribution for the mapping experiment. For a baseline Low-Earth orbit scenario, the initial state mean is initialised as $\mu_v = (7136.635\,\text{km}, 0_4)^T$ and $\mu_\lambda = 0$, where 0_4 denotes a four-dimensional zero vector. The linear covariance matrix is set to $\Sigma_{vv} = \text{diag}((20\,\text{km})^2, 10^{-6}1_4)$ for all experiments, where 1_4 represents the 4×1 vector of ones. Since the shearing effect is most prominent between the semi-major axis a and mean longitude λ elements in

practical experiments, the correlation matrix is set to $\Gamma = \mathrm{diag}(\Gamma_{11}, 10^{-1}1_4)$, where Γ_{11} dictates the specific magnitude of the along-track shear. These parameters are summarised in Table 5.3.

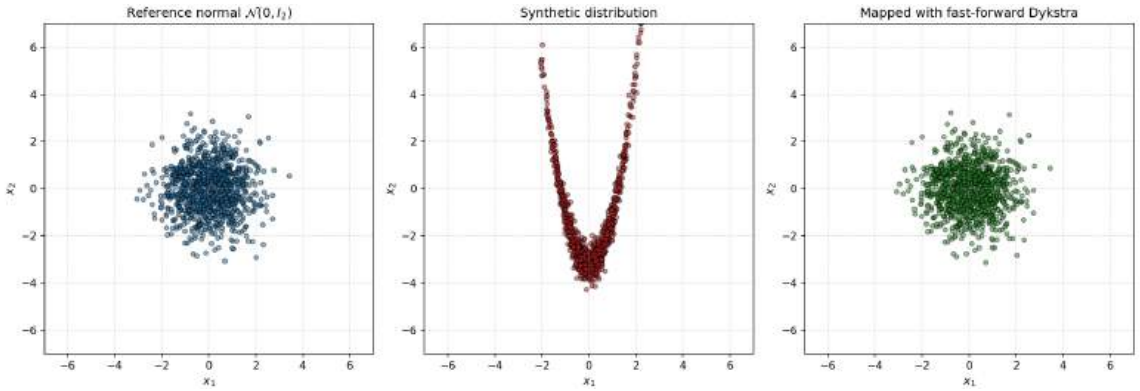
The remainder of the numerical setup is identical to that of previous experiments in this chapter. Simulation parameters for the three experiments are summarised in Table 5.4.

Table 5.4: Parameters for the orbital uncertainty experiments.

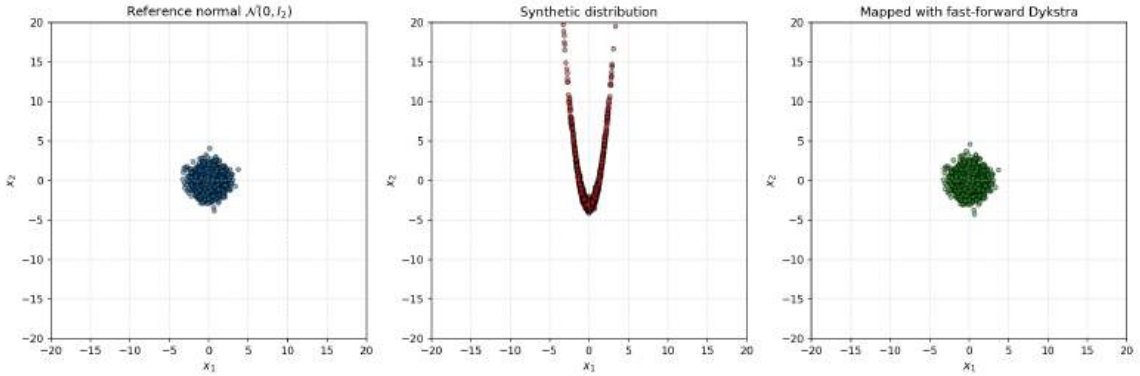
Parameter	**Symbol**	**(a)**	**(b)**	**(c)**
Random seed	–	8989	1234	1234567
State dimension	D	6	6	6
Hermite degree	$J_k + 1$	2	2	2
Particle count	n	1000	2500	1000
Mini-batch cardinality	B	100	700	100
Outer PGD iterations	T	1000	17,500	1000
Dykstra budget	M_{base}	100	100	100
Budget scaling exponent	p	0.67	0.67	0.67
Initial learning rate	η°	0.07	0.07	0.07
Learning rate decay	γ	0.01	0.01	0.01
IHT interval	K_{IHT}	100	100	100
IHT tolerance	ε_p	0.01	0.01	0.01
Monotonicity tolerance	ε_m	1e-12	1e-12	1e-12

To prevent the degeneration of the polynomial coefficients during optimisation, the empirical particle sets are preconditioned using a whitening procedure before training. Both the source and target distributions are initially transformed into a standardised space with zero mean and identity covariance. This affine projection eliminates severe structural correlations and scale disparities, resulting in a well-conditioned numerical environment in which Hermite basis functions can efficiently construct the transport map. In the whitened space, for the third experiment, we set the constraint value at $\bar{\lambda} = 5.0$, and matrix L is obtained through standard Cholesky decomposition. After optimising the map within this standardised domain, the transported particles are projected back into the original physical coordinates by inverting the affine transformation using the target distribution's original scaling and shifting parameters.

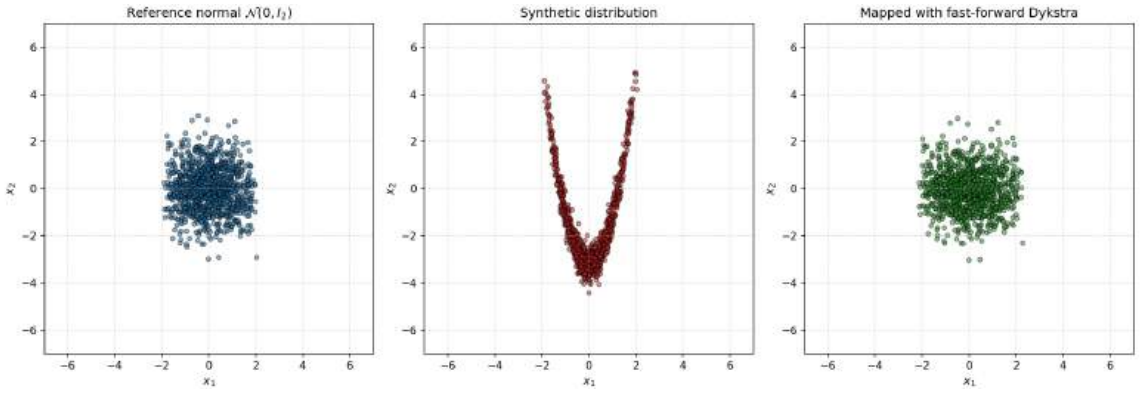
Results for the three experiments are visualised in Figure 5.6. The figure visualises a two-dimensional slice along the semi-major axis–mean longitude (a–λ) plane. The figure shows whitened values for ease of visualisation, since the large scaling of the dominant orbital element a renders the plots otherwise illegible.



(a) Results for the baseline orbital propagation experiment (`SEED` = 8989).



(b) Results for the extreme long-term orbital propagation experiment (`SEED` = 1234).



(c) Results for the discontinuous orbital distribution experiment (`SEED` = 1234567).

Figure 5.6: Results for the three GVM orbital uncertainty experiments.

Figure 5.6 confirms that the GVM distributions are successfully remapped to normal distributions. The empirical Wasserstein-2 distance between the mapped and unmapped distributions was consistently evaluated to be below 10^{-2} across all three configurations.

Static progress plots detailing the step-by-step structural deformation, which were provided for the lower-dimensional synthetic models, are omitted here for readability. Instead, complete temporal visualisations tracking the iterative convergence of the probability contours across all outer projected gradient descent loops have been rendered as animation videos. These are available for review, alongside those of synthetic and other runs, in the following shared folder:

https://drive.google.com/drive/folders/animation_videos

Table 5.5 details the empirical execution times required to optimise each independent scalar map component, S^1 through S^6, alongside the cumulative runtime, for the three experiments.

Table 5.5: Computational runtime for the orbital uncertainty experiments.

		Runtime (s)						
Experiment	Seed	S^1	S^2	S^3	S^4	S^5	S^6	Total
(a)	8989	8.0	22.6	36.6	49.1	61.0	81.0	258.3
(b)	1234	296.4	853.4	1440.2	1779.5	2374.7	3029.9	9774.1
(c)	1234567	11.8	32.4	54.1	74.4	92.0	116.6	381.3

As in previous runs, the component-level runtimes exhibit a monotonic scaling with increasing component dimension. This scaling reflects the growth of the Hermite polynomial basis necessary to parameterise the higher-dimensional conditional maps within the KR architecture. The severity of the initial nonlinear deformation directly influences the total algorithmic runtime. The baseline orbital scenario (a) and the artificially truncated state distribution (c) converged in 258.3 seconds and 381.3 seconds, respectively. Since these were stemming from the same distribution, with the only difference being the truncation constraint, we can conclude the optimiser found it harder to optimise the discontinuous distribution. By contrast, the extreme shear in scenario (b) required significantly greater computational effort, resulting in a cumulative runtime of 9774.1 seconds. Reversing stronger shears necessitates more outer stochastic gradient iterations to push the decision variables to higher overall values. We note that this example was deliberately made extreme, and that such vigorous distribution shears would only be encountered in very long-term propagations.

Since Kalman filters are typically run sequentially at high frequency, this scenario is unlikely to arise in practice.

Despite the increased computational demands associated with the most extreme propagation horizons, these results collectively confirm the robustness of the inexact projection solver when applied to high-fidelity, dimensionally complex aerospace distributions. By reliably enforcing monotonicity and recovering the standard normal geometry from highly sheared equinoctial states, this optimal transport framework demonstrates its suitability as a deterministic transformation mechanism for augmenting the standard Kalman filter architecture.

5.3 Application to geophysics

We proceed by showcasing the efficacy of our inexact projection framework with a geophysics-inspired application in *data assimilation*. Data assimilation in geophysics involves tracking probability measures for chaotic dynamical systems over time. Traditional Bayesian update methods, such as the continuous-discrete-time feedback particle filter, compute the posterior distribution by propagating particles along an observation flow. Although this method reduces the standard resampling degeneracy, computing continuous log-homotopies (continuous deformations from the prior forecast distribution to the updated posterior distribution) requires solving underdetermined linear partial differential equations, which are prone to numerical ill-conditioning and stiffness [25]. The optimal transport formulation introduced in this thesis offers an alternative projection mechanism. Rather than integrating a flow formulation to update physical weights, the KR map is learned directly to transport the prior forecast measure onto the analysis posterior. This approach eliminates the need for intermediate numerical integration of the Bayes update flow while preserving the deterministic mapping of state uncertainty.

5.3.1 Lorenz 63 model

The mathematical framework for this experiment is based on the stochastic Lorenz 1963 system [26]. We obtain a dataset of particles for both a prior and posterior distributions from a continuous flow model, which uses coefficients obtained from a truncation of an orthogonal expansion of the Rayleigh-Bénard convection partial differential equations [25]. To avoid notation ambiguity with the optimal transport unmapped and mapped spatial variables (x and z), the three-dimensional ($D = 3$)

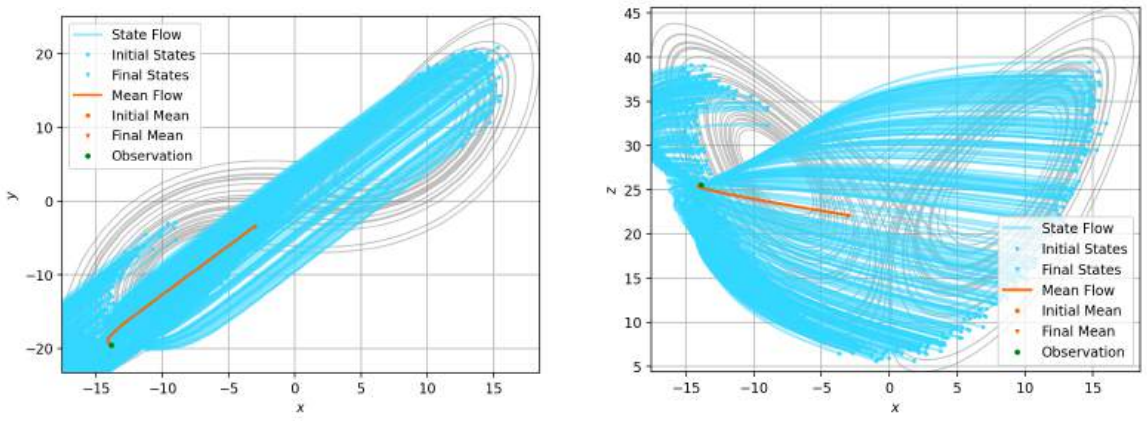
Lorenz state vector is denoted as $q = (q_1, q_2, q_3)^T$. The system evolution is governed by the following stochastic differential equation

$$\mathrm{d}\begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} -c_1 & c_1 & 0 \\ c_2 & -1 & 0 \\ 0 & 0 & -c_3 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} \mathrm{d}t + \begin{bmatrix} 0 \\ -q_1 q_3 \\ q_1 q_2 \end{bmatrix} \mathrm{d}t + \mathrm{d}W_t.$$

Here, W_t represents a standard Brownian motion, which is added as a perturbation to the deterministic Lorenz 1963 (L63) model. This model is popular in geophysics applications, particularly as a low-order model for atmospheric flow. We adopt the standard set of chaotic parameters, with $c_1 = 10$, $c_2 = 28$, and $c_3 = \frac{8}{3}$. This configuration results in a top Lyapunov exponent of $\lambda_1 \approx 0.86$ [25].

The baseline prior distribution is generated by integrating an initial sample ensemble forward for 0.5 time units. The initial distribution is Gaussian, with a mean vector of $(1.508870, -1.531271, 25.46091)^T$ and an isotropic standard deviation of $\sqrt{2}$. The discrete-time observation process assumes a full state measurement, represented as $Y_{t_k} = I_3 q_{t_k} + \xi_{t_k}$, where the measurement noise ξ_{t_k} follows a standard Gaussian distribution $\xi_{t_k} \sim \mathcal{N}(0, I_3)$.

The reference analysis (posterior) distribution is computed using the continuous-discrete time feedback particle filter evaluated over the observation flow. Figure 5.7 shows the projected sample point clouds for the prior and posterior states generated by this baseline method.



(a) Particle flow in the q_1–q_2 plane. (b) Particle flow in the q_1–q_3 plane.

Figure 5.7: Lorenz 1963 prior and posterior samples. Courtesy of Prof. Beeson [25].

Figure 5.7a shows a slice of the three-dimensional model in the q_1–q_2 plane, whereas Figure 5.7b shows a slice in the q_1–q_3 plane. For our experiment, we treat

the three-dimensional initial state samples (blue dots) as the unmapped particle ensemble $\{x^{(i)}\}_{i=1}^{n}$, and the final-state samples (blue triangles) as the target distribution ensemble $\{z^{(i)}\}_{i=1}^{n}$. Our goal is once again to optimise a transport map S to cast the prior distribution particles $x^{(i)}$ onto the posterior states $z^{(i)}$.

5.3.2 Numerical results

The optimal transport framework is evaluated on the L63 model dataset across two distinct sample regimes to assess algorithmic robustness to sample sparsity. The first experiment, denoted (a), considers a highly sparse ensemble with $n = 20$ particles, while the second, denoted (b), uses a denser ensemble of $n = 200$ particles. The remaining numerical setup is identical to that of the other experiments. In both configurations, the physical dimension is $D = 3$, and the total-degree truncation parameter is set to $J_k + 1 = 2$. The stochastic gradient descent mechanism is replaced by full-batch gradient evaluations, and the learning rate η remains constant with a decay rate of $\gamma = 0$. The complete hyperparameter configuration for both experimental regimes is provided in Table 5.6.

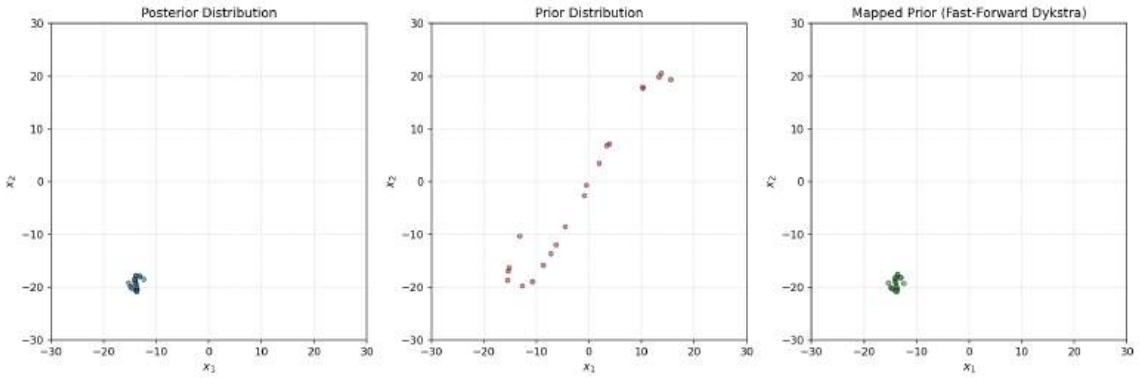
Table 5.6: Parameters for the L63 experiments.

Parameter	**Symbol**	**(a)**	**(b)**
Random seed	–	42	69
State dimension	D	3	3
Hermite degree	$J_k + 1$	2	2
Particle count	n	20	200
Mini-batch cardinality	B	20	200
Outer PGD iterations	T	1000	1000
Dykstra budget	M_{base}	100	100
Budget scaling exponent	p	0.67	0.67
Initial learning rate	η°	0.1	0.1
Learning rate decay	γ	0.0	0.0
IHT interval	K_{IHT}	100	100
IHT tolerance	ε_p	0.01	0.01
Monotonicity tolerance	ε_m	1e-12	1e-12

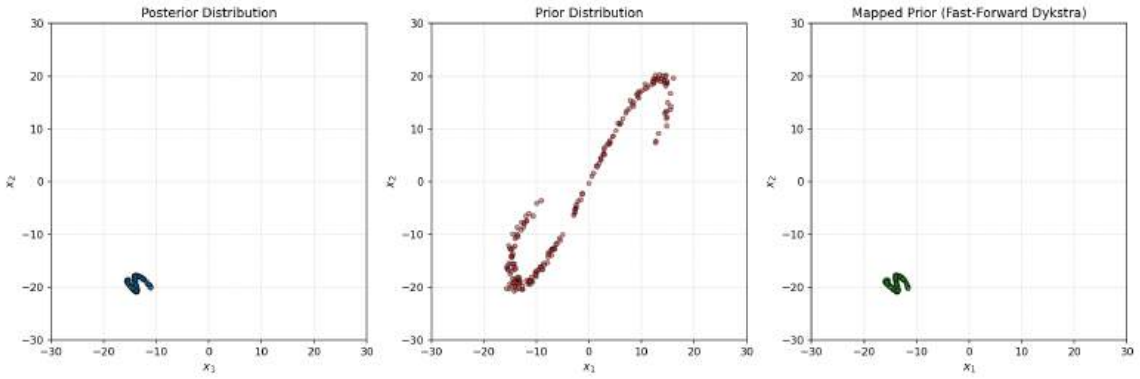
Consistent with the experiments in Section 5.2.4, the empirical data are normalised prior to the initiation of the projected gradient descent framework to prevent weight degeneracy. The unmapped prior particles $x^{(i)}$ and the target standard normal mapped particles $z^{(i)}$ are whitened to achieve zero mean and unit variance. Upon

completion of the outer learning loop at iteration $T-1$, the resulting KR map evaluations are de-whitened to restore the physical dimensions of the geophysical state space.

Results for experiments (a) and (b) are presented in Figure 5.8 on the top (5.8a), and bottom (5.8b) figures, respectively. These results are visualised in the q_1–q_2 plane.



(a) Reconstruction with a sparse particle ensemble (`SEED` = 42, $n = 20$).



(b) Reconstruction with a dense particle ensemble (`SEED` = 69, $n = 200$).

Figure 5.8: Optimal transport results for the L63 system.

Table 5.7 presents the computational runtimes for each configuration. Optimisation of the L63 system dataset maps resulted in reduced execution times relative to previous experiments. This computational efficiency is likely attributable to the lower order of the required transport map.

Progress videos of the optimised maps for a varying number of outer PGD iterations are made available at:

https://drive.google.com/drive/folders/animation_videos.

The empirical results from the L63 system demonstrate the generalisability of the optimal transport framework beyond synthetic and aerospace applications. Recon-

Table 5.7: Runtimes for the L63 experiments.

		Runtime (s)			
Experiment	Seed	S^1	S^2	S^3	Total
(a)	42	1.14	1.19	1.27	3.60
(b)	69	2.63	2.69	2.76	8.08

struction of the KR map to transport the prior forecast measure onto the analysis posterior enables the methodology to circumvent the numerical stiffness associated with traditional observation-flow integration. These findings indicate that addressing the theoretical stalling phenomenon within the foundational projection algorithm produces a robust computational pipeline capable of managing non-linear data assimilation tasks across a range of chaotic systems.

5.4 Performance scaling experiment

Integrating the closed-form stall-resolution algorithm within the optimal transport architecture provides a theoretical foundation for uncertainty propagation and data assimilation applications, as demonstrated throughout this chapter. Nevertheless, the practical utility of this framework depends on its computational feasibility for complex physical systems. Although tests on low-dimensional attractors confirm the algorithmic consistency of the custom projection operator, an effective map reconstruction pipeline must accommodate expanding model dimensions while avoiding the numerical bottlenecks that often impair standard iterative solvers.

As the physical dimensionality D of the state space increases, the number of basis functions grows combinatorially, as we have showcased in Chapter 3, Section 3.2.1. This expansion also raises the dimensionality d of the projection decision variables. For Hermite polynomial bases of degree $J_k + 1$ in D dimensions, the number of basis functions scales as $\mathcal{O}(D^{J_k+1})$, which directly affects the computational complexity of the optimisation problem. To establish the practical relevance of the optimal transport framework for high-dimensional problems, we evaluate the computational scaling of the fast-forward Dykstra algorithm relative to established optimisation software.

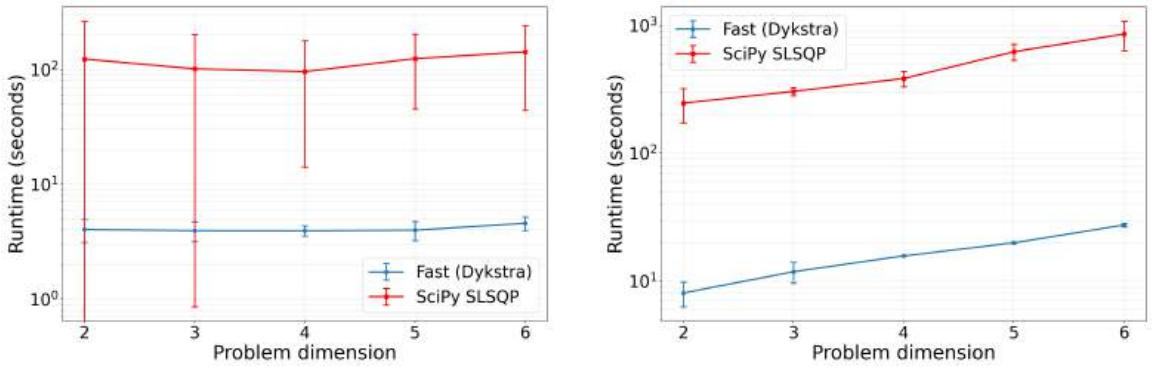
5.4.1 Experimental setup

The scaling study evaluates the computational cost of transport map reconstruction as the state dimension D ranges from 2 to 20. For each dimension, the system is initialised with hyperparameters: $J_k + 1 = 2$, $n = 200$ particles, and $T - 1 = 1000$ PGD iterations. Three independent realisations are performed for each dimension, each with a distinct random seed, to estimate the mean execution time and variance. For this test, we focus on benchmarking the execution time on the projection step specifically, rather than the entire optimisation problem. Since the main advancement brought forward by this thesis is on the stalling fast-forwarding and accelerations to Dykstra's algorithm, this benchmark isolates the performance of Algorithm 3 from the entire optimisation routine.

The fast-forward Dykstra algorithm is benchmarked against two established numerical routines: the Sequential Least Squares Programming (SLSQP) implementation from the SciPy library [27], which offers general-purpose convex optimisation via interior-point methods, and the Operator Splitting Quadratic Program (OSQP) solver [7], which is tailored for large-scale quadratic programs using the alternating direction method of multipliers (ADMM).

5.4.2 Numerical results

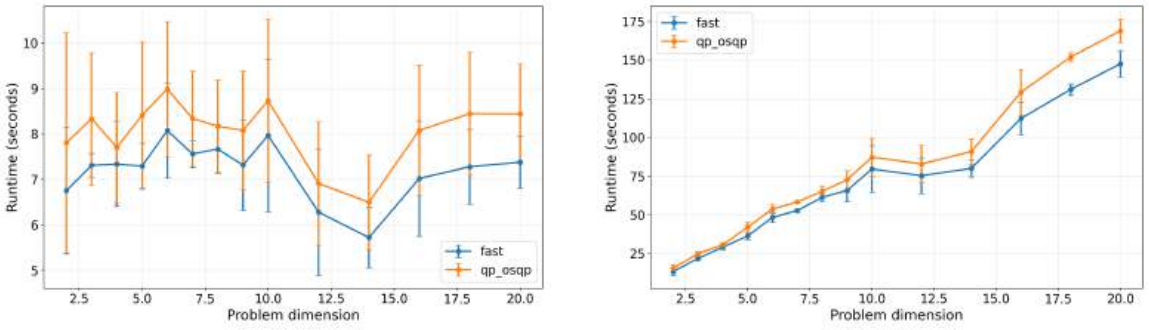
The runtime measurements record the total elapsed time required to solve the projection problem $\mathcal{P}_{\mathcal{H}}(w^{\circ})$ across all D components of the map. This includes initialisation, constraint evaluation, solver execution, and result assembly, thereby providing a realistic assessment of practical wall-clock time for the complete Dykstra pipeline.



(a) Per-component runtime (scalar map). (b) Full map runtime (all components).

Figure 5.9: Runtime scaling comparison: fast-forward Dykstra versus SciPy SLSQP.

Figure 5.9 shows the execution time required to project the weights onto the feasible set $\mathcal{H}$ using the fast-forward Dykstra algorithm compared to the SciPy SLSQP solver. Both component-wise evaluation (Figure 5.9a) and full map reconstruction (Figure 5.9b) are presented on a logarithmic scale to emphasise scaling differences across dimensions. As the state dimension increases, the SciPy routine exhibits substantial computational scaling, with execution time nearly doubling for each additional dimension in higher-dimensional settings. This superlinear scaling results from the increasing complexity of the constraint array and the computational overhead of general-purpose interior-point methods. In contrast, the custom fast-forward Dykstra solver displays much lower growth rates, with runtime scaling approximately linearly with dimension.



(a) Per-component runtime (scalar map). (b) Full map runtime (all components).

Figure 5.10: Runtime scaling comparison: fast-forward Dykstra versus OSQP.

To further assess performance in high-dimensional settings and evaluate scalability relative to a state-of-the-art sparse QP solver, a secondary benchmark is conducted against OSQP. Figure 5.10 displays runtime metrics on a linear scale for both individual map components and the overall system. While OSQP exhibits substantially better scaling than the standard SciPy implementation, the fast-forward Dykstra algorithm consistently achieves lower execution times across all tested dimensions. For example, at $D = 20$ dimensions, the custom solver completes full map reconstruction in approximately 148 seconds, compared to 169 seconds for OSQP, representing a 12% reduction. Notably, the improved scaling trajectory indicates that this performance gap is likely to increase further in higher dimensions.

The computational advantage of the fast-forward Dykstra algorithm stems from several architectural decisions that minimise the projection-loop overhead. First, the algorithm circumvents the dense matrix factorisations and iterative refinements required by general-purpose interior-point and splitting methods. Second, the stalling

detection and half-space removal enhancements help eliminate redundant constraint evaluations, reducing the total number of half-space projections from $\mathcal{O}(M \times D)$ to a number of operations empirically lower. These findings demonstrate that custom projection algorithms can provide significant performance improvements for structured optimisation problems encountered in optimal transport.

Chapter 6

Conclusion and future work

This thesis integrates advances in foundational mathematical optimisation with applications in aerospace and geophysics. We have presented a resolution of the stalling phenomenon and several other enhancements to Dykstra's algorithm, along with a highly scalable framework for computing optimal transport maps, demonstrating the efficacy and applicability of the architecture on several engineering examples.

6.1 Summary of research contributions

The stalling phenomenon in Dykstra's projection algorithm addressed in Chapter 2. The stalling period for polyhedral sets is formally defined, and its duration is derived in closed form (Theorem 2.4.2). Building on this mathematical result, a modified fast-forward algorithm is introduced to detect stalling and advance the auxiliary memory variables beyond the stalled cycle in a single computational step (Algorithm 1). This modification eliminates unpredictable execution times while preserving the asymptotic convergence guarantees of the Boyle-Dykstra theorem (Corollary 2.5.1).

The optimal transport backbone is introduced in Chapter 3. The theory of Knothe-Rosenblatt triangular maps is developed in this chapter, culminating in the formulation of the constrained optimisation problem with objective (3.3.1) and constraints (3.3.2).

The scalable, inexact projection framework is presented in Chapter 4. An accelerated version of the fast-forward Dykstra solver (Algorithm 3) is embedded within an outer projected gradient descent loop (Algorithm 2).

The empirical effectiveness of the combined framework is demonstrated in Chap-

ter 5. When applied to orbital mechanics, the engine reconstructs nonlinear ground-truth physical shears, capturing the heavy-tailed, coupled distributions that challenge classical linearised filtering techniques. In geophysics applications, the framework delivers reliable, fast performance for approximating continuous mappings without requiring the solution of differential equations. Performance scaling metrics confirm the computational viability of the fast-forward modification for large-scale applications.

6.2 Future work

The findings of this thesis suggest several directions for further research in both optimisation theory and aerospace engineering.

From the optimisation perspective, extending the stalling analysis beyond polyhedral constraints to encompass broader classes of convex sets would increase the generality of the fast-forward modification.

In the applied aerospace domain, future research should focus on integrating the optimal transport framework into a Kalman filter and an orbital elements propagator to build an end-to-end uncertainty estimation pipeline. In addition, incorporating real observational data into the stochastic gradient descent loop would assess the framework's robustness against measurement noise and sparse tracking data. Integrating this probability transport engine with operational space situational awareness software could establish a continuous pipeline for autonomous assessment of collision probability.

Bibliography

[1] J. T. Horwood and A. B. Poore. "Gauss von Mises distribution for improved uncertainty realism in space situational awareness". In: *SIAM-ASA Journal on Uncertainty Quantification* 2 (1 2014), pp. 276–304. DOI: 10.1137/130917296.

[2] G. Monge. "Mémoire sur la théorie des déblais et des remblais". In: *Histoire de l'Académie Royale des Sciences de Paris* (1781).

[3] A. Spantini, R. Baptista, and Y. Marzouk. "Coupling Techniques for Nonlinear Ensemble Filtering". In: *SIAM Review* 64.4 (2022), pp. 921–953.

[4] I. Kempf, P. Goulart, and S. Duncan. "Fast Gradient Method for Model Predictive Control with Input Rate and Amplitude Constraints." In: *IFAC-PapersOnLine* 53.2 (2020). 21st IFAC World Congress, pp. 6542–6547. DOI: https://doi.org/10.1016/j.ifacol.2020.12.070.

[5] H. H. Bauschke et al. "On Dykstra's algorithm: finite convergence, stalling, and the method of alternating projections". In: *Optimization Letters* 14.8 (2020), pp. 1975–1987. DOI: 10.1007/s11590-020-01600-4.

[6] S. Boyd and L. Vandenberghe. *Convex Optimization*. 1st ed. Cambridge University Press, 2004. DOI: 10.1017/CBO9780511804441.

[7] B. Stellato et al. "OSQP: an operator splitting solver for quadratic programs". In: *Mathematical Programming Computation* 12.4 (2020), pp. 637–672. DOI: 10.1007/s12532-020-00179-2.

[8] B. O'Donoghue et al. "Conic Optimization via Operator Splitting and Homogeneous Self-Dual Embedding". In: *Journal of Optimization Theory and Applications* 169.3 (June 2016), pp. 1042–1068. DOI: 10.1007/s10957-016-0892-3.

[9] Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. 1st ed. Applied Optimization. Springer Science & Business Media, 2003. DOI: 10.1007/978-1-4419-8853-9.

[10] 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*. Springer New York, 1986, pp. 28–47. DOI: 10.1007/978-1-4613-9940-7_3.

[11] J. Von Neumann. *Functional Operators: The Geometry of Orthogonal Spaces*. Annals of Mathematics Studies Bd. 2. Princeton University Press, 1951. DOI: 10.1515/9781400882250.

[12] L. M. Bregman. "The method of projections for finding the common point of convex sets". In: *Soviet Mathematics Doklady* 6 (1965), pp. 688–692.

[13] C. Perkins. "A Convergence Analysis of Dykstra's Algorithm for Polyhedral Sets". In: *SIAM Journal on Numerical Analysis* 40.2 (July 2002), pp. 792–804. DOI: 10.1137/S0036142900367557.

[14] F. Deutsch and H. Hundal. "The rate of convergence of Dykstra's cyclic projections algorithm: The polyhedral case". In: *Numerical Functional Analysis and Optimization* 15.6 (May 1994), pp. 537–565. DOI: 10.1080/01630569408816580.

[15] A. N. Iusem and A. R. de Pierro. "On the convergence properties of Hildreth's quadratic programming algorithm". In: *Math. Prog. Ser. A and B* 47.1 (May 1990), pp. 37–51. DOI: 10.1007/BF01580851.

[16] S. Xu. "Successive Approximate Algorithm for Best Approximation from a Polyhedron". In: *J. Approx. Theory* 93.3 (June 1998), pp. 415–433. DOI: 10.1006/jath.1998.3174.

[17] C. Vestini and I. Kempf. *Dykstra-Project: Repository for "Fast-Forwarding Stalling in Dykstra's Algorithm"*. `https://github.com/ClouD-161803/Dykstra-Project.git`. 2025.

[18] C. Vestini. *Optimal-Transport-Dykstra: Repository for "ADD TITLE HERE"*. `https://github.com/ClouD-161803/Optimal-Transport-Dykstra.git`. 2026.

[19] D. Goldfarb and A. Idnani. "A numerically stable dual method for solving strictly convex quadratic programs". In: *Mathematical Programming* 27.1 (1983), pp. 1–33.

[20] B. A. Turlach, A. Weingessel, and C. Moler. *quadprog: Functions to solve Quadratic Programming Problems*. R package version 1.5-8 / Python wrapper. 2019.

[21] C. Vestini and I. Kempf. "Fast-Forwarding Stalling in Dykstra's Algorithm". In: *arXiv preprint arXiv:2511.18132* (2025).

[22] H. Robbins and S. Monro. "A stochastic approximation method". In: *The Annals of Mathematical Statistics* (1951), pp. 400–407.

[23] L. Bottou. "Large-scale machine learning with stochastic gradient descent". In: *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[24] T. Karampela and R. Beeson. *A Variational Pseudo-Observation Guided Nudged Particle Filter*. 2026. arXiv: `2603.16705 [eess.SY]`.

[25] R. Beeson. *Filtering with Bayes Update Flow*. Unpublished presentation slides. Delivered on July 26, 2024. Princeton University, 2024.

[26] E. N. Lorenz. "Deterministic Nonperiodic Flow". In: *Journal of the Atmospheric Sciences* 20.2 (1963), pp. 130–141. DOI: `10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2`.

[27] P. Virtanen et al. "SciPy 1.0: fundamental algorithms for scientific computing in Python". In: *Nature Methods* 17.3 (2020), pp. 261–272. DOI: `10.1038/s41592-019-0686-2`.

APPENDIX