

Geometric Quadratic Separators for Symmetric Integer Programs

by

Seyedeh Maryam Abbasnia

Bachelor of Mathematics and Applications, University of Guilan, Iran, 2014

Master of Applied Mathematics, University of Guilan, Iran, 2018

A Thesis Submitted in Partial Fulfilment of
the Requirements for the Degree of

Master of Computer Science

In the Graduate Academic Unit of Computer Science

Supervisor: David Bremner, PhD, Computer Science

Examining Board: Michael Fleming, PhD, Computer Science, Chair
Kalinkar Mandal, PhD, Computer Science
Branimir Ćaćić, PhD, Mathematics and Statistics

This thesis is accepted by the
Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

July, 2026

© Seyedeh Maryam Abbasnia, 2026

Abstract

Symmetry in integer linear programming (ILP) often increases computational effort by generating many equivalent solutions under group actions. One effective approach is to focus on *core points*—distinguished integer representatives whose orbit polytopes are lattice-free—so that optimizing over these representatives preserves optimality while avoiding redundant symmetry. Symmetry groups acting on $\mathbb{R}^n$ naturally induce invariant subspaces through their coordinate-permutation actions. In previous core-point approaches, the case where core points lie in such invariant subspaces was particularly difficult to handle and motivates the main focus of this thesis.

This thesis begins with a two-subspace setting, consisting of the fixed space and Layer 0, and then extends the analysis to a multi-subspace framework based on invariant subspaces induced by cyclic symmetry. We introduce a cone-based separation scheme that constructs quadratic separators using the geometry of these invariant subspaces. The goal is to exclude a candidate point while keeping the relevant symmetry-invariant subspaces feasible. The theoretical framework is developed through coordinate transformations that reveal invariant structure and link algebraic symmetry to geometric feasibility conditions.

We integrate these separators into the SCIP (Solving Constraint Integer Programs) environment through its Python interface, PySCIPOpt, which provides program-

matic access to SCIP. This integration enables direct tests on both feasible and infeasible ILP instances. The experiments show that the cone method can reveal repeated symmetry-related incumbent patterns and convert them into additional quadratic cuts that are added during the search. These results support the method as a geometric cutting procedure and provide a geometric foundation for extending core-point methods to symmetric ILPs.

Dedication

To my parents and my husband, whose love, patience, and unwavering support have guided and inspired me through every step of this journey.

Acknowledgements

I am sincerely grateful to my supervisor, Professor David Bremner, for his guidance, encouragement, and insightful feedback throughout this research. I also thank the members of my examining committee for their valuable comments and support.

Finally, I wish to express my deepest gratitude to my parents and my husband, whose love and unwavering support have been a constant source of strength during my studies.

Table of Contents

Abstract	ii
Dedication	iv
Acknowledgments	v
Table of Contents	vi
List of Tables	x
List of Figures	xi
Abbreviations	xii
1 Introduction	1
2 Mathematical Background	4
2.1 Linear Algebra	4
2.1.1 Vector spaces, inner products, and norms	4
2.1.2 Orthogonality and orthonormal bases (Gram–Schmidt)	7
2.1.3 Subspaces and linear maps	9
2.1.4 Invariant subspaces and orthogonal projections	15
2.1.5 Determinants, block matrices, and the Schur complement	18
2.1.6 Eigenvalues, Diagonalization, and Hermitian matrices	21
2.2 Quadratic Forms and Quadratic Surfaces	25

2.2.1	Symmetric Matrices and Quadratic Forms	25
2.2.2	Quadratic Surfaces	26
2.3	Convexity and Optimization Preliminaries	27
2.3.1	Convex sets and polyhedra	27
2.3.2	Optimization Problem Structure	28
2.4	Symmetry in integer linear programming	29
2.4.1	Linear and integer programming basics	29
2.4.2	Group actions and symmetries in ILPs	31
2.4.3	Invariant subspaces from group actions	32
2.4.4	Core points and orbit polytopes	33
2.5	Circulant Matrices	34
2.5.1	Cyclic shifts, circulant matrices, and invariant subspaces . . .	35
2.5.2	Fixed Space and Layer 0	39
2.5.3	The Singular Case $\det(\text{Cir}(c))=0$	40
2.6	Real Fourier Vectors and the Discrete Fourier Transform	43
2.6.1	Real Fourier Geometry and the $\Delta(h, n)$ Operators	44
2.6.2	The Trigonometric Matrix $R(n)$	48
2.7	Δ -Matrices, Δ -Subspaces, and Projections	52
2.7.1	Invariant Subspaces defined by Δ -matrices	53
2.7.2	Orthogonal Decomposition of $\mathbb{R}^n$ into Fourier Mode Subspaces	57
2.7.3	Projections onto Δ -subspaces	68
2.8	Cones and Quadratic Forms for Symmetry-Based Separation	69
2.8.1	Spherical cones and quadratic forms for separation	71
3	Deriving Constraints	74
3.1	The Quadratic Separator	77
3.1.1	Setting and notation	77
3.1.2	Subspace decomposition	80

3.1.3	Building the basis B	90
3.1.4	Subspaces project to axis-orthogonal hyperplanes	92
3.1.5	Separators in the compressed coordinates	96
3.2	Hermitian-Based Separator	98
3.2.1	Properties of Hermitian Matrices	99
3.2.2	Real-Imaginary Decomposition and Construction of the Hermitian Matrix	100
3.2.3	Definition of the Hermitian-Based Separator	101
3.3	The Two-Subspace Quadratic Separator	106
3.3.1	Transformation to (y_0, y_1) -coordinates	111
3.3.2	Properties of the separator	112
3.3.3	Visualization in transformed coordinates	116
3.3.4	Numerical Experiments	121
3.4	Generalizing to Multiple Subspaces	124
3.4.1	Multi-Subspace Setup and Compressed Coordinates	125
3.4.2	Derivation of the Cone Constraint in k Dimensions	127
3.4.3	The Algebraic Cone Constraint	131
3.4.4	Acceptance and Rejection Properties of the Cone	134
3.4.5	Two-Subspace Prototype vs. Multi-Subspace Cone	137
4	Geometric Cone Separators in a Cut-and-Solve Framework	139
4.1	Motivation and Overview	139
4.2	Cone-Based Cutting Algorithm	143
4.2.1	Compressed coordinates	145
4.2.2	Full-Rank Properties of the Basis Matrix	147
4.2.3	Cone score and pulled-back quadratic cut	157
4.3	Pseudocode Description	160
4.3.1	Explicit Reformulation versus Cone Method	166

5	Conclusion and Future Work	176
5.1	Conclusion	176
5.2	Future Work: Cubic and Logarithmic barrier-based separators	178
5.2.1	Cubic Separators	179
5.2.2	Open Questions for Further Exploration	181
 Curriculum Vitae		

List of Tables

3.1	Separator evaluations across tested dimensions. Entries for $S(e)$ and $S(f)$ vanish up to numerical precision, while $S(p) \neq 0$ in all cases. Therefore, the normalized separator satisfies $\hat{S}(p) = 1$ by construction.	124
4.1	Run settings used in the comparison and follow-up experiments.	167
4.2	Model characteristics of the original instances and the corresponding explicit reformulations.	168
4.3	Main comparison between the explicit reformulation baseline and the cone method.	169
4.4	Default cone runs on additional cyclic instances.	172
4.5	Effect of parameter changes on selected cone runs.	173
4.6	Plain SCIP on binary reformulations under a 1-hour time limit.	174

List of Figures

2.1	First two steps of Gram–Schmidt: $u_1 = v_1$, $u_2 = v_2 - \text{proj}_{u_1}(v_2)$, and $q_1 = u_1/\ u_1\ $, $q_2 = u_2/\ u_2\ $	8
3.1	Hermitian-based cubic separator $Q(x) = (e^\top x)(x^\top Hx)$ (generated in Racket). The function $Q(x)$ vanishes on the Fixed space $\text{span}\{e\}$ and on Layer 0 ($e^\top x = 0$), and is used to cut off the candidate point p . This cutting behavior can eliminate multiple “corner” points at once in small illustrative linear-programming examples with only a few vertices, but it comes at the cost of increased algebraic degree. . .	104
3.2	Level set $2y_0y_1 = 1$ in the compressed coordinates $y = Px = (y_0, y_1)^\top$. The function $2y_0y_1$ is zero exactly on the coordinate axes $y_0 = 0$ and $y_1 = 0$. In the two–subspace setting, these axes correspond to Layer 0 and the Fixed space, so the function vanishes precisely on those two subspaces.	107
3.3	Contours of $\hat{S}(x) = 1$ and $\hat{S}(x) = 0.5$ in the (y_0, y_1) -plane for $n = 3$ (generated in Racket). The point p is plotted in compressed coordinates $y = MB^\top p = (y_0, y_1)$; the axes $y_0 = 0$ and $y_1 = 0$ correspond to Layer 0 and Fixed space.	117
3.4	Contours of $\hat{S}(x) = 1$ and $\hat{S}(x) = 0.5$ in the (y_0, y_1) -plane for $n = 208$ (generated in Racket). The point p is plotted in compressed coordinates $y = MB^\top p = (y_0, y_1)$; the axes $y_0 = 0$ and $y_1 = 0$ correspond to Layer 0 and Fixed space.	118

List of Symbols, Nomenclature or Abbreviations

Mathematical Symbols and Notation

B	Basis matrix
$\text{Cir}(c_0, \dots, c_{n-1})$	Circulant matrix with entries c_i
e, f	e : fixed direction; f : orthogonal complementary direction
Fix	Fixed space generated by e , i.e., $\text{Fix} = \text{span}\{e\}$
F_n (DFT)	Complex Discrete Fourier Transform matrix
I_n	The $n \times n$ identity matrix
L_0	Layer 0 subspace $L_0 = \{x : e^\top x = 0\}$, orthogonal to e

M	Projection matrix extracting coordinates
P_h	Orthogonal projector onto V_h
Q, Q'	Quadratic form matrices
Q_{cone}	Cone-separator matrix $Q_{\text{cone}} = (k - 1)I - cee^\top$
$R(n)$	Scott matrix in the trigonometric determinant formula
$\mathbb{R}^n$	n -dimensional real vector space
$S(x)$	Separator function $S(x) = x^\top Q'x$
$\widehat{S}(x)$	Normalized separator function $\widehat{S}(x) = S(x)/S(p)$
V_h	Invariant subspace $V_h = \ker \Delta(h, n)$
$V_h^\perp$	Fourier mode subspace $V_h^\perp = \text{span}\{V(h, n), U(h, n)\}$
Y	Compressed coordinates $Y = MB^\top x$
Y^*	Conjugate transpose of matrix Y
$\bar{z}$	Complex conjugate of the complex number z

$\Delta(h, n)$	Matrix built from the real Fourier vectors $V(h, n)$ and $U(h, n)$
$\bigoplus_h V_h^\perp$	Orthogonal decomposition of $\mathbb{R}^n$ into Fourier mode subspaces
$P(c)$	Orbit polytope of c , defined as $P(c) = \text{conv}(G \cdot c)$
$\langle x, y \rangle$	Inner product of x and y ; Euclidean by default, equal to $x^\top y$
$\ x\ $	Euclidean norm of x , i.e., $\sqrt{\langle x, x \rangle}$

Abbreviations

CPLEX	IBM optimization solver
ILP	Integer Linear Program
Knitro	Nonlinear optimization solver
PySCIPOpt	Python interface to SCIP
SCIP	Solving Constraint Integer Programs optimization solver

Chapter 1

Introduction

Integer Linear Programming (ILP) is a central framework in combinatorial optimization, widely used in logistics, scheduling, and network design. A significant source of computational difficulty arises when an ILP exhibits *symmetry*: multiple feasible solutions are equivalent under the action of a permutation group. For example, in many scheduling and logistics problems, if a set of identical machines or delivery vehicles can perform the same tasks, permuting their assignments often produces distinct mathematical solutions with the same objective value. These equivalent solutions introduce symmetry into the optimization problem and cause the solver to explore equivalent solutions repeatedly during the search. As first analyzed by Margot [4], such symmetries lead to redundant search in branch-and-bound algorithms and can severely increase solution times if left unaddressed.

A key theoretical advance was the introduction of *core points* by Bödi, Herr, and Joswig [5]. Core points form a subset of integer solutions whose orbit polytopes are lattice-free, ensuring that optimization restricted to this subset is equivalent to the full symmetric problem. This geometric interpretation connects symmetry reduction in ILPs with invariant structures in convex geometry. Herr, Rehn, and Schürmann [6]

further formalized these ideas by linking group-invariant subspace decompositions with polyhedral symmetry handling, laying a foundation for algebraic and geometric approaches to symmetry.

In some symmetry classes, the number of core points can be infinite, such as for cyclic groups. In this setting, standard core point methods struggle to provide a finite characterization. Shahverdi [1] addressed this by creating outer approximations of the core points. She did not enumerate points directly. Instead, she exploited the structure of circulant matrices arising from cyclic symmetry. By analyzing the eigenvalues of these matrices, she derived nonlinear constraints that define the outer approximations. These constraints allow the solver to exclude regions of the space that do not contain any core points, without listing all core points. She proposed three algorithms in order to add the nonlinear constraints to the ILP and solve it using nonlinear solvers such as Knitro. This work provided a clear geometric characterization of how symmetry influences the feasible region.

Banihashemi [2] implemented these theoretical algorithms proposed by Shahverdi [1] in the SCIP solver through PySCIPOpt, incorporating constraints derived from the symmetry structure of the problem. She conducted extensive computational experiments demonstrating practical reductions in redundant search caused by symmetry.

This thesis continues and generalizes this line of research by developing a cone-based quadratic framework for separating infeasible points in symmetric ILPs. Building on an initial two-subspace setting introduced in this work, the proposed method exploits invariant subspaces of $\mathbb{R}^n$ induced by the symmetry group, which in the cyclic case arise from the structure of circulant matrices. These invariant subspaces are described later through the Fourier-based constructions used in the thesis, and a quadratic separator is constructed using the geometry of these subspaces. The resulting quadratic inequalities can then be incorporated into an optimization solver

as nonlinear cutting constraints, providing a geometric mechanism for excluding candidate points during the search process. The separator is defined by a quadratic form $S(x) = x^\top Qx$, whose associated matrix encodes geometric feasibility conditions. This construction provides a unified geometric foundation for analyzing symmetry in integer programming and sets the stage for the dynamic separation framework developed in later chapters.

Chapter 2 presents the mathematical background, including group actions, circulant matrices, invariant subspaces, the Fourier-based constructions used later to define these subspaces, cone geometry, and quadratic forms. Chapter 3 develops the construction of cone-based quadratic separators, building the required basis transformations through Racket experiments and relating these separators to the fixed space and Layer 0 as well as to more general invariant subspaces. Chapter 4 describes the integration of these quadratic separators into the SCIP solver and reports computational experiments illustrating their behavior. Chapter 5 concludes with future work, including extensions to cubic separators and logarithmic barrier-based methods.

Chapter 2

Mathematical Background

This chapter provides the mathematical foundation needed to support the theoretical developments in this thesis. We present key concepts from linear algebra, integer programming with symmetry, core points and orbit polytopes, invariant subspaces and circulant matrices. The emphasis is on the material required later when we analyze symmetry, core points, and the special case where a circulant determinant vanishes.

2.1 Linear Algebra

The material in this section follows standard references in linear algebra [13, 14, 29].

2.1.1 Vector spaces, inner products, and norms

Vector spaces. A *vector space* (or linear space) over a field $\mathbb{F}$ (such as $\mathbb{R}$ or $\mathbb{C}$) is a set V , whose elements are called vectors, together with two operations:

1. vector addition: $V \times V \rightarrow V$, $(u, v) \mapsto u + v$;

2. scalar multiplication: $\mathbb{F} \times V \rightarrow V, (a, v) \mapsto av$;

satisfying the usual axioms of a vector space, including commutativity and associativity of addition, existence of a zero vector and additive inverses, and distributivity and compatibility of scalar multiplication.

Linear independence. A set of vectors $\{v_1, v_2, \dots, v_n\}$ in a vector space V over the field $\mathbb{F}$ is said to be *linearly independent* if the only scalars $a_1, a_2, \dots, a_n \in \mathbb{F}$ satisfying

$$a_1v_1 + a_2v_2 + \dots + a_nv_n = 0$$

are $a_1 = a_2 = \dots = a_n = 0$.

Span. Let V be a vector space and let $\{v_1, v_2, \dots, v_n\}$ be a set of vectors in V . The *span* of these vectors is the set of all linear combinations of them:

$$\text{span}(v_1, v_2, \dots, v_n) = \{a_1v_1 + a_2v_2 + \dots + a_nv_n \mid a_1, a_2, \dots, a_n \in \mathbb{F}\}.$$

Basis of a vector space. A subset $B = \{v_1, \dots, v_n\} \subseteq V$ is a *basis* of V if it is linearly independent and $\text{span}(B) = V$. Equivalently, every $v \in V$ is uniquely a linear combination of vectors in B .

We next equip our vector space with an inner product, which induces a norm and provides a convenient way to express orthogonality.

Inner product space. An *inner product space* is a vector space V together with an inner product $\langle \cdot, \cdot \rangle : V \times V \rightarrow \mathbb{F}$ (where $\mathbb{F}$ is $\mathbb{R}$ or $\mathbb{C}$) satisfying, for all $u, v, w \in V$ and all scalars $a, b \in \mathbb{F}$:

1. $\langle v, v \rangle \geq 0$ and $\langle v, v \rangle = 0$ if and only if $v = 0$;

$$2. \langle au + bv, w \rangle = a\langle u, w \rangle + b\langle v, w \rangle;$$

$$3. \langle v, w \rangle = \overline{\langle w, v \rangle};$$

$$4. \langle v, aw \rangle = \bar{a} \langle v, w \rangle.$$

Here $\bar{a}$ denotes the complex conjugate of a when $\mathbb{F} = \mathbb{C}$, and simply a itself when $\mathbb{F} = \mathbb{R}$.

Norm. Let V be an inner product space with inner product $\langle \cdot, \cdot \rangle$. For a vector $v \in V$, the *norm* (or length) of v is defined as

$$\|v\| = \sqrt{\langle v, v \rangle}.$$

This norm defines the length of v in the inner product space.

Standard inner product. In this thesis, we primarily work with the standard inner product on $\mathbb{R}^n$. For two vectors $v = (v_1, v_2, \dots, v_n)$ and $w = (w_1, w_2, \dots, w_n)$ in $\mathbb{R}^n$, the inner product is defined as

$$\langle v, w \rangle = \sum_{i=1}^n v_i w_i.$$

For vectors in $\mathbb{C}^n$, the standard inner product is

$$\langle v, w \rangle = \sum_{i=1}^n v_i \overline{w_i}.$$

Orthogonality is characterized by $\langle v, w \rangle = 0$ in Subsection 2.1.2.

Unit vectors. A vector $v \in V$ is said to have *unit length* (or be a *unit vector*) if its norm equals one, that is,

$$\|v\| = \sqrt{\langle v, v \rangle} = 1.$$

2.1.2 Orthogonality and orthonormal bases (Gram–Schmidt)

Orthogonal and orthonormal sets. A set $\{v_1, \dots, v_n\}$ is *orthogonal* iff $\langle v_i, v_j \rangle = 0$ for all $i \neq j$.

A set of vectors $\{v_1, \dots, v_n\}$ in an inner product space is called *orthonormal* if it is orthogonal and each vector has unit length. Equivalently,

$$\langle v_i, v_j \rangle = \delta_{ij} \quad \text{for all } i, j,$$

where δ_{ij} is the *Kronecker delta*: $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise.

Gram–Schmidt orthonormalization

Theorem 2.1.1 (Gram–Schmidt process). *Let $\{v_1, v_2, \dots, v_n\}$ be a linearly independent set of vectors in an inner product space V . Then there exists an orthonormal set $\{q_1, q_2, \dots, q_n\}$ in V such that*

$$\text{span}(q_1, q_2, \dots, q_n) = \text{span}(v_1, v_2, \dots, v_n).$$

A standard presentation of this construction can be found in [10, 13]. For the purposes of this thesis, the key step is the orthogonalization procedure: starting with

$u_1 = v_1$, one defines

$$u_k = v_k - \sum_{j=1}^{k-1} \frac{\langle v_k, u_j \rangle}{\langle u_j, u_j \rangle} u_j, \quad k = 2, \dots, n.$$

Thus u_k is obtained by subtracting from v_k its projection onto the span of the previously constructed vectors $u_1, \dots, u_{k-1}$, so that u_k is orthogonal to them. Normalizing these residual vectors,

$$q_k = \frac{u_k}{\|u_k\|}, \quad k = 1, \dots, n,$$

produces an orthonormal set $\{q_1, \dots, q_n\}$ spanning the same subspace as $\{v_1, \dots, v_n\}$.

Figure 2.1 illustrates the orthogonalization and normalization steps of the Gram–Schmidt process for the first two vectors.

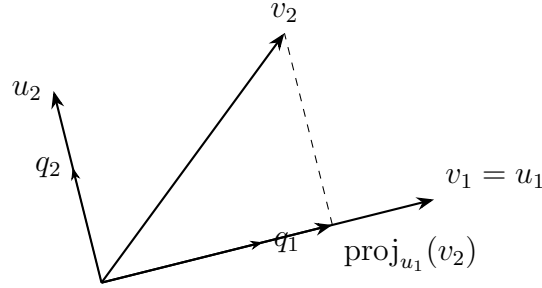


Figure 2.1: First two steps of Gram–Schmidt: $u_1 = v_1$, $u_2 = v_2 - \text{proj}_{u_1}(v_2)$, and $q_1 = u_1/\|u_1\|$, $q_2 = u_2/\|u_2\|$.

Orthonormal bases. Let V be an inner product space. A set of vectors $\{e_1, \dots, e_n\}$ in V is called an *orthonormal basis* of V if:

1. the set $\{e_1, e_2, \dots, e_n\}$ is orthonormal, i.e.

$$\langle e_i, e_j \rangle = \delta_{ij}, \quad \text{for all } i, j,$$

where δ_{ij} is the Kronecker delta;

2. the set spans V , i.e.

$$V = \text{span}(e_1, e_2, \dots, e_n).$$

Standard basis. The standard basis of $\mathbb{R}^n$ consists of the vectors $e_1, \dots, e_n$, where e_i has a 1 in the i -th coordinate and 0 in all other coordinates. The same definition applies to $\mathbb{C}^n$.

Example. [Standard basis of $\mathbb{R}^3$] Consider the set of vectors

$$e_1 = (1, 0, 0), \quad e_2 = (0, 1, 0), \quad e_3 = (0, 0, 1).$$

This set is called the *standard basis* of $\mathbb{R}^3$. With respect to the standard Euclidean inner product (dot product),

$$\langle e_i, e_j \rangle = \delta_{ij}, \quad \text{for all } i, j.$$

Thus, $\{e_1, e_2, e_3\}$ forms an orthonormal basis of $\mathbb{R}^3$.

2.1.3 Subspaces and linear maps

Subspaces and related constructions. Let V be a vector space over a field $\mathbb{F}$. A subset $W \subseteq V$ is called a *subspace* of V if W is itself a vector space under the same operations of vector addition and scalar multiplication as V . Equivalently, W is a subspace if:

1. $0 \in W$ (the zero vector belongs to W);
2. if $u, v \in W$, then $u + v \in W$ (closed under addition);
3. if $a \in \mathbb{F}$ and $v \in W$, then $av \in W$ (closed under scalar multiplication).

Two subspaces $W_1, W_2 \subseteq V$ of an inner product space V are said to be *orthogonal* if

$$\langle w_1, w_2 \rangle = 0 \quad \text{for every } w_1 \in W_1 \text{ and } w_2 \in W_2.$$

Example. [Orthogonal subspaces in $\mathbb{R}^3$] Consider the subspaces of $\mathbb{R}^3$:

$$W_1 = \text{span}\{(1, 0, 0)\}, \quad W_2 = \text{span}\{(0, 1, 0), (0, 0, 1)\}.$$

Here W_1 is the x -axis, and W_2 is the yz -plane.

For any $w_1 \in W_1$ and $w_2 \in W_2$, we have

$$\langle w_1, w_2 \rangle = 0,$$

with respect to the standard Euclidean inner product.

Thus, W_1 and W_2 are orthogonal subspaces of $\mathbb{R}^3$.

Definition 2.1.1. Let V be an inner product space and let $U \subseteq V$ be a subspace.

The orthogonal complement of U , denoted $U^\perp$, is defined as

$$U^\perp = \{v \in V \mid \langle v, u \rangle = 0 \text{ for all } u \in U\}.$$

That is, $U^\perp$ consists of all vectors in V that are orthogonal to every vector in U .

In the previous example, taking $U = W_1 = \text{span}\{(1, 0, 0)\}$ gives

$$U^\perp = \text{span}\{(0, 1, 0), (0, 0, 1)\} = W_2,$$

so the orthogonal complement of the x -axis is the yz -plane.

Definition 2.1.2. Let V be a vector space and let $U_1, \dots, U_m \subseteq V$ be subspaces. We

say that V is the sum of the U_i if every vector $v \in V$ can be expressed as

$$v = u_1 + \cdots + u_m, \quad u_i \in U_i.$$

If this representation is unique, meaning that

$$u_1 + \cdots + u_m = u'_1 + \cdots + u'_m \implies u_i = u'_i \text{ for all } i,$$

then V is called the direct sum of the U_i , written

$$V = U_1 \oplus U_2 \oplus \cdots \oplus U_m.$$

Definition 2.1.3 (Two-subspace case). If $U, W \subseteq V$ are subspaces such that $U \cap W = \{0\}$ and $U + W = V$, then every $v \in V$ has a unique decomposition $v = u + w$ with $u \in U$ and $w \in W$. In this case we write

$$V = U \oplus W.$$

Example. In $\mathbb{R}^2$, let $U = \text{span}\{(1, 0)\}$ and $W = \text{span}\{(0, 1)\}$. Every vector $v = (a, b)$ decomposes uniquely as $v = (a, 0) + (0, b)$ with $(a, 0) \in U$ and $(0, b) \in W$. Thus $\mathbb{R}^2 = U \oplus W$.

Linear maps and operators. Let V and W be vector spaces over the same field $\mathbb{F}$. A map $T : V \rightarrow W$ is called a *linear transformation* if for all $u, v \in V$ and all scalars $a \in \mathbb{F}$,

$$T(u + v) = T(u) + T(v), \quad T(av) = aT(v).$$

Equivalently,

$$T(au + bv) = aT(u) + bT(v), \quad \text{for all } u, v \in V, \ a, b \in \mathbb{F}.$$

A *linear operator* is a linear transformation $T : V \rightarrow V$ from a vector space V into itself. That is, T is a map satisfying

$$T(au + bv) = aT(u) + bT(v), \quad \text{for all } u, v \in V, \ a, b \in \mathbb{F}.$$

kernel of a linear map. Let $T : V \rightarrow W$ be a linear map between vector spaces. The *kernel* (also called the *null space*) of T is the set of all vectors that are mapped to the zero vector of W :

$$\ker(T) = \{v \in V \mid T(v) = 0\}.$$

The kernel is always a subspace of V , since, by linearity of T , it contains the zero vector and is closed under addition and scalar multiplication.

Example. Consider the linear map $T : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ defined by $T(x, y, z) = (x, y)$. Then

$$\ker(T) = \{(0, 0, z) : z \in \mathbb{R}\}.$$

Geometrically, the kernel is the z -axis in this case. All vectors in this subspace are mapped to the zero vector by T , while components orthogonal to the kernel (i.e., lying in the xy -plane) are preserved by the map.

Image and column space. For a linear map $T : V \rightarrow W$, the *image* of T is

$$\text{im}(T) = \{T(v) : v \in V\}.$$

For a matrix $A \in \mathbb{F}^{m \times n}$ with columns $a_1, \dots, a_n$,

$$\text{im}(A) = \{Ax : x \in \mathbb{F}^n\} = \text{span}(a_1, \dots, a_n).$$

Thus, for matrices, the image is also called the *column space*.

Linear transformations and matrices. Every matrix $A \in \mathbb{F}^{m \times n}$ defines a linear transformation

$$L_A : \mathbb{F}^n \rightarrow \mathbb{F}^m, \quad L_A(v) = Av.$$

Conversely, every linear transformation $T : \mathbb{F}^n \rightarrow \mathbb{F}^m$ can be represented by a matrix with respect to the standard bases. Under this correspondence, the image of L_A is the column space of A , and the kernel of L_A is the null space of A .

Transpose. Let $A = (a_{ij}) \in \mathbb{F}^{m \times n}$ be a matrix. The *transpose* of A , denoted $A^\top$, is the $n \times m$ matrix obtained by interchanging rows and columns, that is,

$$A^\top = (a_{ji})_{1 \leq j \leq n, 1 \leq i \leq m}.$$

Equivalently, the (i, j) -entry of $A^\top$ is a_{ji} .

A real square matrix $A \in \mathbb{R}^{n \times n}$ is called *symmetric* if

$$A^\top = A.$$

Conjugate transpose. For a complex matrix $A \in \mathbb{C}^{n \times n}$, the *conjugate transpose* (also called the Hermitian transpose), denoted A^* , is defined as

$$A^* = \overline{A^\top},$$

that is, the transpose of A followed by entrywise complex conjugation. If A has only real entries, then $A^* = A^\top$.

Example. [Conjugate transpose of a complex matrix] Let

$$A = \begin{bmatrix} 1 & -2-i & 5 \\ 1+i & i & 4-2i \end{bmatrix}.$$

First take the transpose:

$$A^\top = \begin{bmatrix} 1 & 1+i \\ -2-i & i \\ 5 & 4-2i \end{bmatrix}.$$

Then conjugate each entry:

$$A^* = \overline{A^\top} = \begin{bmatrix} 1 & 1-i \\ -2+i & -i \\ 5 & 4+2i \end{bmatrix}.$$

Adjoint of a linear map. Let V and W be finite-dimensional inner-product spaces over $\mathbb{F}$, where $\mathbb{F} = \mathbb{R}$ or $\mathbb{C}$, and let $T : V \rightarrow W$ be a linear transformation. The *adjoint* of T is the unique linear transformation

$$T^* : W \rightarrow V$$

satisfying

$$\langle Tv, w \rangle = \langle v, T^*w \rangle$$

for every $v \in V$ and $w \in W$.

If T is represented by a matrix A , then the adjoint map is represented by the transpose matrix $A^\top$ when A is real and by the conjugate transpose matrix A^* when A is complex. Thus,

$$(L_A)^* = L_{A^\top} \quad \text{for real matrices,}$$

and

$$(L_A)^* = L_{A^*} \quad \text{for complex matrices.}$$

A linear operator $T : V \rightarrow V$ is called *self-adjoint* if $T = T^*$. For real matrices this is equivalent to symmetry, $A = A^\top$, while for complex matrices it is equivalent to the Hermitian condition $A = A^*$.

2.1.4 Invariant subspaces and orthogonal projections

Let V be a vector space and let $T : V \rightarrow V$ be a linear operator. A subspace $W \subseteq V$ is said to be *T -invariant* (or invariant under T) if applying T to any vector in W produces another vector that still lies in W , that is, if

$$w \in W \Rightarrow T(w) \in W.$$

Equivalently, $T(W) \subseteq W$.

Projections. Let V be a vector space. A linear map $P : V \rightarrow V$ is called a projection if it satisfies

$$P^2 = P.$$

That is, applying P twice to any vector gives the same result as applying it once. In this case, the image of P is the subspace onto which vectors are projected, and the kernel of P consists of the directions that are mapped to zero by the projection.

Orthogonal projections

Definition 2.1.4. *Let V be an inner product space and let $U \subseteq V$ be a subspace. The orthogonal projection of a vector $x \in V$ onto U is the unique vector $y \in U$ such that $x - y \in U^\perp$. Equivalently, y is the element of U that minimizes $\|x - y\|$.*

We denote this vector by $\text{proj}_U(x)$, that is,

$$y = \text{proj}_U(x).$$

Orthogonal projections are used throughout this thesis, since they are the closest-point projections onto subspaces in an inner product space.

In finite-dimensional inner-product spaces, orthogonal projections admit an equivalent algebraic characterization. If

$$P : V \rightarrow V$$

is the linear map associated with the orthogonal projection onto a subspace U , then P is the unique projection onto U satisfying

$$P = P^*,$$

that is, P is self-adjoint. In the real case, this condition becomes

$$P = P^\top,$$

so a real projection matrix is orthogonal precisely when it is symmetric.

The following projection formulas and characterization follow standard results on orthogonal projections; see, for example, [29, Section 6C].

Projection onto a line. Let $u \in V$, $u \neq 0$. The orthogonal projection of $x \in V$ onto $\text{span}\{u\}$ is given by

$$\text{proj}_u(x) = \frac{\langle x, u \rangle}{\langle u, u \rangle} u.$$

Projection onto a subspace. Let $U \subseteq V$ be a k -dimensional subspace with an orthonormal basis $\{u_1, \dots, u_k\}$. Then for any $x \in V$, the orthogonal projection of x onto U is

$$\text{proj}_U(x) = \sum_{i=1}^k \langle x, u_i \rangle u_i.$$

Characterization of orthogonal projection. Let $U \subseteq V$ be a subspace and let $x \in V$. Define the residual vector

$$r = x - \text{proj}_U(x).$$

By Definition 2.1.4, the residual satisfies

$$r \in U^\perp.$$

Thus the residual vector is orthogonal to the subspace onto which x is projected.

This orthogonality condition is the same idea used in the Gram–Schmidt process (Theorem 2.1.1): the residual vector is orthogonal to the subspace spanned by the previously constructed vectors.

Projection onto the orthogonal complement. If U is a subspace of an inner product space V and $x \in V$, then

$$\text{proj}_{U^\perp}(x) = x - \text{proj}_U(x).$$

Equivalently,

$$\text{proj}_{U^\perp} = I_n - \text{proj}_U,$$

where I_n is the $n \times n$ identity matrix.

2.1.5 Determinants, block matrices, and the Schur complement

Determinants and invertibility

The determinant is a scalar value associated with a square matrix. It plays a central role in linear algebra, particularly in characterizing invertibility, eigenvalues, and solutions to linear equations.

Let $A = (a_{ij}) \in \mathbb{F}^{n \times n}$ be a square matrix over a field $\mathbb{F}$. The *determinant* of A , denoted $\det(A)$, is the scalar defined recursively as follows:

1. For $n = 1$, if $A = [a_{11}]$, then $\det(A) = a_{11}$.
2. For $n \geq 2$, the determinant is given by the *Laplace expansion* along the first row:

$$\det(A) = \sum_{j=1}^n (-1)^{1+j} a_{1j} \det(A_{1j}),$$

where A_{1j} is the $(n-1) \times (n-1)$ matrix obtained by deleting the first row and the j -th column of A .

As a basic example, the identity matrix I_n satisfies $\det(I_n) = 1$.

A square matrix $A \in \mathbb{F}^{n \times n}$ is called *invertible* (or *nonsingular*) if there exists a matrix $B \in \mathbb{F}^{n \times n}$ such that

$$AB = BA = I_n.$$

The matrix B is unique and is called the *inverse* of A , denoted A^{-1} .

For a matrix $A \in \mathbb{F}^{m \times n}$, the rank of A , denoted $\text{rank}(A)$, is the dimension of its row space (equivalently, of its column space), that is, the maximum number of linearly independent rows (or columns) of A . A square matrix $A \in \mathbb{F}^{n \times n}$ is called *rank-deficient* if $\text{rank}(A) < n$. For square matrices, rank deficiency is equivalent to non-invertibility. That is, A is rank-deficient if and only if A is not invertible.

Theorem 2.1.2 (Basic Properties of the Determinant). *For square matrices $A, B \in \mathbb{F}^{n \times n}$, the determinant satisfies the following:*

1. $\det(AB) = \det(A) \det(B)$;
2. $\det(A^\top) = \det(A)$;
3. *If A is rank-deficient, then $\det(A) = 0$;*
4. *Adding a scalar multiple of one row to another does not change the determinant.*

Theorem 2.1.3 (Equivalent Characterizations of Invertibility). *For a square matrix $A \in \mathbb{F}^{n \times n}$, the following statements are equivalent:*

1. *A is invertible;*
2. $\det(A) \neq 0$;

3. $\ker(A) = \{0\}$;
4. $Ax = 0$ has only the trivial solution;
5. the columns of A are linearly independent;
6. the columns of A span $\mathbb{F}^n$;
7. the equation $Ax = b$ has a unique solution for every $b \in \mathbb{F}^n$.

Block matrices and the Schur complement

A *block matrix* (or *partitioned matrix*) is a matrix that has been subdivided into rectangular submatrices called *blocks*. Block partitions allow matrix identities to be expressed and proved more clearly, especially when using elimination, inversion, or factorization arguments.

Let $p, q \geq 0$ with $p + q > 0$. A matrix $M \in \mathbb{C}^{(p+q) \times (p+q)}$ is said to be partitioned into blocks if it can be written as

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix},$$

where

$$A \in \mathbb{C}^{p \times p}, \quad B \in \mathbb{C}^{p \times q}, \quad C \in \mathbb{C}^{q \times p}, \quad D \in \mathbb{C}^{q \times q}.$$

Definition 2.1.5 (Schur Complement). *If A is invertible, the Schur complement of A in M is*

$$M/A := D - CA^{-1}B \in \mathbb{C}^{q \times q}.$$

If D is invertible, the Schur complement of D in M is

$$M/D := A - BD^{-1}C \in \mathbb{C}^{p \times p}.$$

The Schur complement appears throughout numerical linear algebra, statistics, and optimization. It also provides determinant factorizations that are frequently used in block Gaussian elimination.

Proposition. [Schur Determinant Formulas] Let M be the block matrix above.

1. If A is invertible, then

$$\det(M) = \det(A) \cdot \det(D - CA^{-1}B) = \det(A) \cdot \det(M/A).$$

2. If D is invertible, then

$$\det(M) = \det(D) \cdot \det(A - BD^{-1}C) = \det(D) \cdot \det(M/D).$$

This result is standard in block matrix theory; see, for example, [14].

2.1.6 Eigenvalues, Diagonalization, and Hermitian matrices

Let $A \in \mathbb{F}^{n \times n}$ be a square matrix over a field $\mathbb{F}$.

1. A scalar $\lambda \in \mathbb{F}$ is called an *eigenvalue* of A if there exists a nonzero vector $y \in \mathbb{F}^n$ such that

$$Ay = \lambda y.$$

Such a vector y is called an *eigenvector* of A associated with the eigenvalue λ .

Equivalently, λ is an eigenvalue of A if and only if $A - \lambda I$ is not invertible.

2. The *characteristic polynomial* of A is the polynomial

$$p_A(\lambda) = \det(A - \lambda I),$$

whose roots are precisely the eigenvalues of A .

Theorem 2.1.4. *Let $A \in \mathbb{F}^{n \times n}$ and suppose $\lambda_1, \dots, \lambda_m$ are distinct eigenvalues of A with corresponding eigenvectors $v_1, \dots, v_m$. Then the set $\{v_1, \dots, v_m\}$ is linearly independent.*

This is a standard result; see, for example, [29, Proposition 5.11].

Diagonalization

The following result explains why diagonalization preserves key spectral properties, and in particular why the diagonal entries of a diagonalized matrix correspond to the eigenvalues of the original matrix.

Theorem 2.1.5. *Let $A \in \mathbb{F}^{n \times n}$ and let $P \in \mathbb{F}^{n \times n}$ be invertible. Define $D := P^{-1}AP$. Then:*

1. $\det(D) = \det(A)$;
2. *A and D have the same characteristic polynomial, and therefore the same eigenvalues, each occurring with the same multiplicity.*

This is a standard result in linear algebra; see, for example, [29].

Definition 2.1.6. *A matrix $A \in \mathbb{F}^{n \times n}$ is called diagonalizable if there exists an invertible matrix $P \in \mathbb{F}^{n \times n}$ and a diagonal matrix $\Lambda \in \mathbb{F}^{n \times n}$ such that*

$$A = P\Lambda P^{-1}.$$

The diagonal entries of Λ are the eigenvalues of A , and the columns of P are the corresponding eigenvectors.

Eigenbasis. A basis consisting entirely of eigenvectors of a matrix A is called an *eigenbasis* of A .

Theorem 2.1.6 (Diagonalization via an eigenbasis). *Suppose $A \in \mathbb{F}^{n \times n}$ admits a basis of eigenvectors $\{y_0, \dots, y_{n-1}\}$ with corresponding eigenvalues $\lambda_0, \dots, \lambda_{n-1}$. Let*

$$P = [y_0 \ y_1 \ \cdots \ y_{n-1}], \quad \Lambda = \text{diag}(\lambda_0, \dots, \lambda_{n-1}).$$

Then A is diagonalizable, and in fact

$$AP = P\Lambda \iff P^{-1}AP = \Lambda.$$

Proof. Since $\{y_0, \dots, y_{n-1}\}$ is a basis of $\mathbb{F}^n$, the columns of P are linearly independent, and hence P is invertible. By the eigenvector relations $Ay_j = \lambda_j y_j$ for each j , we obtain

$$\begin{aligned} AP &= A[y_0 \ y_1 \ \cdots \ y_{n-1}] \\ &= [Ay_0 \ Ay_1 \ \cdots \ Ay_{n-1}] \\ &= [\lambda_0 y_0 \ \lambda_1 y_1 \ \cdots \ \lambda_{n-1} y_{n-1}] \\ &= P\Lambda. \end{aligned}$$

Multiplying both sides on the left by P^{-1} yields $P^{-1}AP = \Lambda$, which is diagonal. $\square$

Remark. In explicit block-matrix form, the diagonalization reads

$$A = P\Lambda P^{-1} = \begin{bmatrix} | & | & & | \\ y_0 & y_1 & \cdots & y_{n-1} \\ | & | & & | \end{bmatrix} \begin{bmatrix} \lambda_0 & 0 & \cdots & 0 \\ 0 & \lambda_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_{n-1} \end{bmatrix} P^{-1}.$$

If A has n distinct eigenvalues over $\mathbb{F}$, then it admits n linearly independent eigenvectors, hence a full eigenbasis, and is therefore diagonalizable; see, for example, [23, 13].

Hermitian matrices

We now focus on Hermitian matrices, since they have a well-behaved spectral theory—real eigenvalues and orthonormal eigenvectors—which will be used later when working with quadratic forms and change-of-basis constructions.

Definition 2.1.7. *A matrix $U \in \mathbb{C}^{n \times n}$ is unitary if $U^*U = I$ (equivalently $U^{-1} = U^*$). Over $\mathbb{R}$, this condition becomes $Q^\top Q = I$, and such matrices are called orthogonal.*

Proposition. Let $Y \in \mathbb{C}^{n \times n}$. The columns of Y form an orthonormal basis of $\mathbb{C}^n$ if and only if Y is unitary, that is, $Y^{-1} = Y^*$.

This follows directly from the definition of an orthonormal set, since the entries of Y^*Y are given by

$$(Y^*Y)_{ij} = \langle y_i, y_j \rangle,$$

where y_i and y_j are columns of Y .

Hermitian matrices. Recall from Section 2.1.3 that the adjoint of a complex matrix is represented by its conjugate transpose. A matrix $A \in \mathbb{C}^{n \times n}$ is called Hermitian, or self-adjoint, if

$$A = A^*.$$

In the real case, this condition reduces to

$$A = A^\top,$$

so Hermitian matrices generalize real symmetric matrices.

Remark. Hermitian matrices have two key properties [14]:

1. All eigenvalues of a Hermitian matrix are real.
2. Eigenvectors corresponding to distinct eigenvalues are orthogonal.

These facts are fundamental for both theoretical and computational purposes.

Theorem 2.1.7 (Spectral Theorem for Hermitian Matrices [23]). *If $A \in \mathbb{C}^{n \times n}$ is Hermitian, then there exists a unitary matrix U such that*

$$U^*AU = \Lambda,$$

where Λ is a diagonal matrix whose diagonal entries are the real eigenvalues of A . Equivalently, A admits an orthonormal basis of eigenvectors and is therefore unitarily diagonalizable.

2.2 Quadratic Forms and Quadratic Surfaces

The definitions and properties of quadratic forms and quadratic surfaces used here follow standard treatments in matrix analysis and optimization [14, 16].

2.2.1 Symmetric Matrices and Quadratic Forms

As recalled in Subsection 2.1.6, a real square matrix $A \in \mathbb{R}^{n \times n}$ is symmetric if $A^\top = A$. Equivalently, $a_{ij} = a_{ji}$ for all $1 \leq i, j \leq n$.

Definition 2.2.1. *Let $A \in \mathbb{F}^{n \times n}$ be a square matrix and let $x \in \mathbb{F}^n$. The expression*

$$Q(x) = x^\top Ax$$

is called a quadratic form in the variable x . When $\mathbb{F} = \mathbb{R}$ and A is symmetric, the quadratic form is associated with the homogeneous polynomial of degree two whose coefficients are given by the entries of A .

Geometrically, the level sets $\{x \in \mathbb{R}^n : x^\top Qx = c\}$ form quadratic surfaces, which we summarize next.

2.2.2 Quadratic Surfaces

Definition 2.2.2. A quadratic surface (or quadric) in $\mathbb{R}^n$ is the set of all points $x \in \mathbb{R}^n$ satisfying a polynomial equation of total degree two:

$$Q(x) = x^\top Ax + b^\top x + c = 0,$$

where $A \in \mathbb{R}^{n \times n}$ is a symmetric matrix ($A = A^\top$), $b \in \mathbb{R}^n$, and $c \in \mathbb{R}$.

Remark. Quadrics include ellipsoids, paraboloids, and hyperboloids. An orthogonal change of variables diagonalizes A and reveals the shape.

Example.

1. In $\mathbb{R}^3$, the unit sphere

$$x^2 + y^2 + z^2 = 1$$

corresponds to $A = I_3$, $b = 0$, and $c = -1$.

2. The paraboloid

$$z = x^2 + y^2$$

can be written as $Q(x, y, z) = x^2 + y^2 - z = 0$, where $A = \text{diag}(1, 1, 0)$, $b = (0, 0, -1)^\top$, $c = 0$.

3. The hyperboloid of one sheet

$$x^2 + y^2 - z^2 = 1$$

has $A = \text{diag}(1, 1, -1)$, $b = 0$, and $c = -1$.

Having developed the linear–algebraic foundations—including vector spaces, inner products, orthogonality, and projections—we now shift from algebraic structure to geometric structure. Convex sets provide a natural framework for describing feasible regions in optimization problems. The following section introduces the basic definitions needed to work with convex sets and related geometric objects.

2.3 Convexity and Optimization Preliminaries

Background on convex sets, polyhedra, and their basic properties follows [15].

2.3.1 Convex sets and polyhedra

Convex set. A set $X \subseteq \mathbb{R}^n$ is *convex* if $tx + (1 - t)y \in X$ for all $x, y \in X$ and all $t \in [0, 1]$. In other words, the line segment between any two points of X remains inside X .

Convex hull. Let $S \subseteq \mathbb{R}^n$ be a set. The *convex hull* of S , denoted by $\text{conv}(S)$, is the smallest convex set containing S . This is equivalent to the intersection of all convex sets that contain S .

A subset $A \subseteq \mathbb{R}^n$ is called *affine* if for all $x, y \in A$ and all $t \in \mathbb{R}$, the point

$$tx + (1 - t)y$$

also lies in A . Equivalently, A is a translation of a linear subspace.

A *closed halfspace* in $\mathbb{R}^n$ is a set of the form

$$H = \{x \in \mathbb{R}^n : a^\top x \leq b\},$$

where $a \in \mathbb{R}^n$ is a nonzero vector and $b \in \mathbb{R}$. Halfspaces are convex and form the basic building blocks of polyhedra.

Convex polyhedron. A *convex polyhedron* is an intersection of finitely many closed halfspaces.

A point $v \in P$ is a *vertex* of a polyhedron P if there exists $c \neq 0$ such that $c^\top v > c^\top y$ for all $y \in P \setminus \{v\}$.

The dimension $\dim P$ is the dimension of the smallest affine space containing P .

2.3.2 Optimization Problem Structure

Remark. In optimization, feasible regions are typically described using *halfspaces*, *affine sets*, and *intersections* of these objects. A linear constraint of the form $a^\top x \leq b$ defines a halfspace, while equations such as $Ax = b$ describe an affine set (the smallest affine space containing all solutions of the equation). The feasible region of a linear or integer program is the intersection of a finite number of such sets.

Basic optimization terminology. This section summarizes basic terminology used in optimization and will be used throughout the discussion of linear and integer programs. The optimization concepts reviewed in this section follow standard references in mathematical optimization [16].

Remark. A general optimization problem consists of an *objective function* to be

maximized or minimized, and a collection of *constraints* that restrict the allowable values of the decision variable $x \in \mathbb{R}^n$. The set of all points satisfying the constraints is called the *feasible region* (or feasible set). Any optimization problem is solved over its feasible region.

A general optimization problem can be written in the form

$$\min_{x \in X} f(x) \quad \text{or} \quad \max_{x \in X} f(x),$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is the objective function and $X \subseteq \mathbb{R}^n$ is the feasible region. If X is described by linear inequalities and the objective function f is linear, then the problem is called a *linear program*.

The geometric notions introduced above—convexity, convex combinations, and polyhedra—lead naturally to linear programming, whose feasible regions are convex polyhedral sets, and to integer programming, where the feasible set consists of integer points and is analyzed via the convex hull of those points. Because the optimization problems studied in this thesis inherit additional structure from symmetry, it is useful to recall the standard formulation of linear and integer programs before examining how group actions interact with these models. We therefore turn next to a brief review of linear and integer programming and the role that symmetry plays in their structure.

2.4 Symmetry in integer linear programming

2.4.1 Linear and integer programming basics

The definitions and terminology in this section follow standard integer programming references [17, 18].

Integer programming (IP) concerns optimization problems in which some or all decision variables are required to take integer values. A general *integer linear program (ILP)* can be written as

$$\max \quad c^\top x \quad \text{subject to } Ax \leq b, \quad x \in \mathbb{Z}^n,$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $c \in \mathbb{R}^n$.

The associated *linear programming (LP) relaxation* allows $x \in \mathbb{R}^n$:

$$\max \quad c^\top x \quad \text{subject to } Ax \leq b, \quad x \in \mathbb{R}^n,$$

which defines a polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$. The LP relaxation is efficiently solvable in polynomial time, and its optimal value provides an upper bound for maximization (or lower bound for minimization).

Remark. General ILPs are NP-hard. This means that, unless $P = NP$, there is no algorithm that can solve all ILPs in time polynomial in the input size. In the worst case, the solution time of known algorithms may grow exponentially with the number of variables and constraints. Practical solvers therefore rely on a combination of techniques: branch-and-bound to explore a search tree of subproblems, branch-and-cut to add cutting planes that tighten relaxations, and heuristics for finding good feasible solutions quickly. However, the presence of *symmetry* can cause large families of equivalent solutions to appear in the search tree, dramatically increasing the run time. This motivates modeling and reduction strategies based on the symmetry structure of the problem, which will be developed later (see Section 2.5).

2.4.2 Group actions and symmetries in ILPs

Basic definitions on group actions and symmetry follow standard group theory references [19, 20], while the discussion of symmetry in integer programming follows [4].

Let S_n denote the symmetric group on n elements. Each $g \in S_n$ acts on a vector $x = (x_1, \dots, x_n)^\top \in \mathbb{R}^n$ by permuting its coordinates:

$$(g(x))_i = x_{g^{-1}(i)}.$$

Definition 2.4.1. *An integer linear program (ILP) with polyhedron*

$$P = \{x \in \mathbb{R}^n : Ax \leq b\}$$

and objective vector $c \in \mathbb{R}^n$ has a symmetry if there exists $g \in S_n$ such that

$$g(x) \in P \quad \text{and} \quad \langle c, g(x) \rangle = \langle c, x \rangle \quad \text{for all } x \in P.$$

The set of all such permutations forms the symmetry group $G \leq S_n$ of the ILP.

Thus, a symmetry maps every feasible solution to another feasible solution with the same objective value.

Example. Let $P = \text{conv}\{(1, 2, 3, 4), (4, 1, 2, 3), (3, 4, 1, 2), (2, 3, 4, 1)\}$. The cyclic permutation $(1\ 2\ 3\ 4)$ maps each vertex to the next one. We consider the symmetry group generated by this permutation,

$$G = \langle (1\ 2\ 3\ 4) \rangle.$$

Orbit. Given a group $G \leq S_n$ acting on $\mathbb{R}^n$, the *orbit* of a vector $x \in \mathbb{R}^n$ is

$$\text{orb}_G(x) = \{g(x) : g \in G\}.$$

2.4.3 Invariant subspaces from group actions

Let $G \leq S_n$ act on $\mathbb{R}^n$ by permuting coordinates as defined above. A linear subspace $W \subseteq \mathbb{R}^n$ is *G-invariant* if $g \cdot W \subseteq W$ for all $g \in G$; equivalently, whenever $w \in W$ then $g \cdot w \in W$ for every $g \in G$. Two canonical G -invariant subspaces used throughout this thesis are

$$\text{Fix} := \{x \in \mathbb{R}^n : g \cdot x = x \ \forall g \in G\}, \quad L_0 := \{x \in \mathbb{R}^n : \langle x, e \rangle = 0\},$$

where $e = (1, \dots, 1)^\top$ is the all-ones vector.

In the case of permutation actions, $\text{Fix} = \text{span}\{e\}$, since the only vectors invariant under all coordinate permutations are constant vectors.

The fixed space and Layer 0. The fixed space $\text{Fix} = \text{span}\{e\}$ is G -invariant by definition: if $x \in \text{Fix}$ then $g \cdot x = x$ for all $g \in G$, so in particular $g \cdot x \in \text{Fix}$. The subspace $L_0 = \{x \in \mathbb{R}^n : \langle x, e \rangle = 0\}$ is also G -invariant. Indeed, each $g \in G$ acts by permuting coordinates, which does not change the sum of entries, so $\langle g \cdot x, e \rangle = \langle x, e \rangle$ for all $x \in \mathbb{R}^n$. Hence if $x \in L_0$ then $\langle g \cdot x, e \rangle = 0$, and therefore $g \cdot x \in L_0$. Finally, since $\text{Fix} = \text{span}\{e\}$ and $L_0 = \{x : \langle x, e \rangle = 0\}$, we have $L_0 = \text{Fix}^\perp$, and thus the orthogonal direct-sum decomposition $\mathbb{R}^n = \text{Fix} \oplus L_0$.

2.4.4 Core points and orbit polytopes

Symmetries in integer programs generate many equivalent feasible integer points. Core points play an important role in symmetry reduction by providing canonical representatives for certain symmetry orbits, namely those whose orbit polytopes contain no additional integer points. Orbit polytopes offer a geometric way to formalize this reduction. The definitions and properties of core points and orbit polytopes follow [5, 6].

Let G be a permutation group acting on $\mathbb{R}^n$ by permuting coordinates, and let $c \in \mathbb{R}^n$. The *orbit polytope* of c is

$$\mathcal{P}(c) = \text{conv}(G \cdot c).$$

Definition. Let G be a permutation group acting on $\mathbb{Z}^n$. An integer point $c \in \mathbb{Z}^n$ is called a *core point* for G if and only if its orbit polytope $\mathcal{P}(c) = \text{conv}(G \cdot c)$ is *lattice-free*; that is, the only integer points contained in $\mathcal{P}(c)$ are its vertices.

Remark. Core points serve as canonical representatives for G -orbits of integer points. This viewpoint provides background for the symmetry-reduction ideas used in this thesis.

The preceding sections reviewed the linear algebraic and optimization foundations needed throughout this thesis. We now turn to a class of structured matrices that play an important role later: circulant matrices. These matrices possess a high degree of regularity and admit explicit eigenvector formulas, properties that will be essential in constructing invariant subspaces and in developing the Fourier-type decompositions appearing in subsequent sections. Before introducing the real Fourier basis, we first recall the fundamental properties of circulant matrices and their connection with the fixed space and Layer 0.

2.5 Circulant Matrices

Circulant matrices naturally arise when studying cyclic symmetries in integer linear programs. Each circulant matrix encodes the action of the cyclic group C_n on a vector by cyclic shifts. Because circulant matrices are diagonalized by the discrete Fourier transform (DFT), they provide an efficient tool for analyzing invariant subspaces. In particular, circulant matrices clarify the connection between cyclic symmetry, the fixed space, and Layer 0, which will be used later in the construction of Fourier-based subspaces and quadratic separators. Background on circulant matrices and their spectral properties follows [21, 22].

Definition 2.5.1. A circulant matrix $C = \text{Cir}(c_0, \dots, c_{n-1}) \in \mathbb{R}^{n \times n}$ has entries

$$C_{i,j} = c_{(j-i) \bmod n}.$$

Equivalently,

$$\text{Cir}(c_0, \dots, c_{n-1}) = \begin{bmatrix} c_0 & c_{n-1} & c_{n-2} & \cdots & c_3 & c_2 & c_1 \\ c_1 & c_0 & c_{n-1} & \cdots & c_4 & c_3 & c_2 \\ c_2 & c_1 & c_0 & \cdots & c_5 & c_4 & c_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ c_{n-2} & c_{n-3} & c_{n-4} & \cdots & c_1 & c_0 & c_{n-1} \\ c_{n-1} & c_{n-2} & c_{n-3} & \cdots & c_2 & c_1 & c_0 \end{bmatrix}.$$

Observe that $\text{Cir}(c)$ is formed by stacking all coordinate shifts of c as columns; thus its columns represent the orbit of c under the standard coordinate-permutation action of C_n .

2.5.1 Cyclic shifts, circulant matrices, and invariant subspaces

The purpose of this section is to show how cyclic shifts determine an invariant subspace structure, and how circulant matrices provide a natural way to represent it.

In the cyclic case, let $\sigma = (0, 1, 2, \dots, n-1)$ denote the one-step cyclic shift, and let T_σ be its permutation matrix. Given a vector $c \in \mathbb{R}^n$, we will study the subspace spanned by its cyclic shifts. The circulant matrix $C = \text{circ}(c)$ packages these shifts as columns:

$$C = [c \ T_\sigma c \ T_\sigma^2 c \ \cdots \ T_\sigma^{n-1} c].$$

The span of these shifted vectors is exactly the column space of C , and we denote it by

$$W := \text{col}(C) = \text{span}\{c, T_\sigma c, \dots, T_\sigma^{n-1} c\}.$$

Thus W consists of all linear combinations of cyclic shifts of c . Equivalently, any nonzero vector $w \in W$ can be written in the form

$$w = C\lambda$$

for some coefficient vector $\lambda \neq 0$.

The next step is to show that W is T_σ -invariant. Now consider applying T_σ to a generic vector $w \in W$. Let $M(i)$ denote the i -th column of a matrix M (indices taken modulo n). Expanding $w = C\lambda$ by columns gives

$$w = C\lambda = \sum_{i=0}^{n-1} \lambda_i C(i),$$

and therefore

$$T_\sigma w = T_\sigma C \lambda = \sum_{i=0}^{n-1} \lambda_i (T_\sigma C)(i).$$

Because $C = \text{circ}(c)$ has cyclic shifts as columns, applying T_σ to a column produces another column of C (a reindexing of the orbit):

$$(T_\sigma C)(i) = C(\sigma(i)).$$

Substituting this into the sum gives

$$T_\sigma w = \sum_{i=0}^{n-1} \lambda_i C(\sigma(i)).$$

Reindexing the sum (equivalently, collecting the coefficient in front of each fixed column $C(j)$) by setting $j = \sigma(i)$, so that $i = \sigma^{-1}(j)$, yields

$$T_\sigma w = \sum_{j=0}^{n-1} \lambda_{\sigma^{-1}(j)} C(j) = C(T_\sigma^{-1} \lambda) \in \text{col}(C) = W.$$

In other words, T_σ maps the set of columns of C into itself. Since $W = \text{col}(C)$ consists of all linear combinations of these columns, shifting any vector in W keeps it in W . Equivalently, $T_\sigma W \subseteq W$, so W is T_σ -invariant.

It is worth noting that for any linear map (in particular, for T_σ) there are always two *trivial* invariant subspaces: $\{0\}$ and $\mathbb{R}^n$. In our setting, if $c \neq 0$ then at least one column of $C = \text{Cir}(c)$ is nonzero, so

$$W = \text{col}(C) \neq \{0\}.$$

Thus the only way for W to be *nontrivial* (i.e., $W \neq \mathbb{R}^n$) is if the columns of C fail to span all of $\mathbb{R}^n$.

This is captured by the *rank-nullity theorem*:

$$\dim(\operatorname{col}(C)) + \dim(\ker C) = n.$$

Since $\operatorname{col}(C) = W$, we have $\dim(W) < n$ if and only if $\dim(\ker C) > 0$, i.e., if and only if there exists a nonzero vector h such that

$$Ch = 0.$$

Equivalently, C does not have full rank. For an $n \times n$ matrix this is the same as saying that C is singular, or

$$\det(C) = 0.$$

In summary, W is a nontrivial T_σ -invariant subspace of $\mathbb{R}^n$ exactly when the circulant matrix $C = \operatorname{Cir}(c)$ is singular.

Remark. The fixed space $\operatorname{Fix} = \operatorname{span}\{e\}$ and the layer space $L_0 = e^\perp$ are G -invariant for permutation actions, and $L_0 = \operatorname{Fix}^\perp$, hence $\mathbb{R}^n = \operatorname{Fix} \oplus L_0$. In the cyclic case, circulant matrices package cyclic shifts into a single matrix, and the span of those shifts is an invariant subspace; singularity of the circulant matrix corresponds to the shifted columns failing to span all of $\mathbb{R}^n$. These facts motivate the Fourier-mode description of circulant matrices and, later, the $\Delta(h, n)$ constructions used to define the invariant subspaces that are used in the separator construction.

Definition 2.5.2. The discrete Fourier transform (DFT) matrix $\mathcal{F}_n = ((\mathcal{F}_n)_{j,k}) \in \mathbb{C}^{n \times n}$ is defined by

$$(\mathcal{F}_n)_{j,k} = \frac{1}{\sqrt{n}} \omega_n^{-jk}, \quad 0 \leq j, k \leq n-1,$$

where $\omega_n = e^{2\pi i/n}$ is the primitive n -th root of unity. This matrix is unitary, satis-

fyng $\mathcal{F}_n^* \mathcal{F}_n = I$ (which implies $|\det(\mathcal{F}_n)| = 1$).

Theorem 2.5.1. *Every circulant matrix C is diagonalizable by the discrete Fourier transform (DFT) matrix $\mathcal{F}_n$:*

$$C = \mathcal{F}_n^* \Lambda \mathcal{F}_n,$$

where $\Lambda = \text{diag}(\lambda_0, \dots, \lambda_{n-1})$ contains the eigenvalues of C . Moreover, if $w^{(k)}$ denotes the k -th column of $\mathcal{F}_n^*$, then

$$w^{(k)} = \frac{1}{\sqrt{n}} (1, \omega_n^k, \omega_n^{2k}, \dots, \omega_n^{(n-1)k})^\top,$$

and

$$C w^{(k)} = \lambda_k w^{(k)}, \quad k = 0, \dots, n-1,$$

so the columns of $\mathcal{F}_n^*$ form an eigenbasis of C .

In particular,

$$w^{(0)} = \frac{1}{\sqrt{n}} e,$$

where $e = (1, \dots, 1)^\top$ is the all-ones vector.

Circulant matrices in optimization. The use of circulant-matrix structure in symmetry-based optimization follows [3].

Let $\omega_n = e^{2\pi i/n}$. The eigenvalues of $C = \text{Cir}(c_0, \dots, c_{n-1})$ are

$$\lambda_k = \sum_{j=0}^{n-1} c_j \omega_n^{jk}, \quad k = 0, \dots, n-1. \quad (2.1)$$

So,

$$\det(C) = \prod_{k=0}^{n-1} \lambda_k.$$

This closed-form spectrum enables fast detection of singular cases and clarifies the geometry of symmetry-invariant subspaces. Since $w^{(0)} = \frac{1}{\sqrt{n}}e$, the eigenvalue λ_0 corresponds to the all-ones direction. From (2.1),

$$\lambda_0 = \sum_{j=0}^{n-1} c_j.$$

Therefore $\lambda_0 = 0$ if and only if $e^\top c = 0$, equivalently $c \in L_0$. In our work, the case $\lambda_0 = \sum_j c_j = 0$ (hence $\det(C) = 0$) plays a central role and forces the image of $\text{Cir}(c)$ into Layer 0; see Subsection 2.5.3.

To summarize, in this thesis circulant matrices encode cyclic orbits: for $c \in \mathbb{Z}^n$, the columns of $\text{Cir}(c)$ are cyclic shifts of c , and the convex hull of these columns defines a symmetric polytope associated with c .

2.5.2 Fixed Space and Layer 0

Let $e = (1, \dots, 1)^\top \in \mathbb{R}^n$. The fixed space is $\text{span}\{e\}$: it consists exactly of constant vectors, and the Layer 0 subspace is the orthogonal complement

$$L_0 := \{x \in \mathbb{R}^n : e^\top x = 0\} = (\text{span}\{e\})^\perp,$$

i.e., the vectors whose entries sum to zero.

Every $x = (x_1, \dots, x_n)^\top \in \mathbb{R}^n$ decomposes uniquely into a *constant part* plus a *zero-sum part*:

$$x = \sigma e + l, \quad \sigma = \frac{1}{n} e^\top x = \frac{1}{n} \sum_{i=1}^n x_i, \quad l := x - \sigma e \in L_0.$$

Thus σ is simply the average of the coordinates of x , and l is what remains after subtracting that average from every entry, so $e^\top l = 0$.

2.5.3 The Singular Case $\det(\text{Cir}(c)) = 0$

The characterization of singular circulant matrices via their eigenvalues follows [21, 22].

Let $c = (c_0, \dots, c_{n-1})^\top$ and $C = \text{Cir}(c)$. With $\omega_n = e^{2\pi i/n}$ and DFT matrix $\mathcal{F}_n$,

$$C = \mathcal{F}_n^* \Lambda \mathcal{F}_n, \quad \Lambda = \text{diag}(\lambda_0, \dots, \lambda_{n-1}), \quad \lambda_k = \sum_{j=0}^{n-1} c_j \omega_n^{jk}.$$

By Theorem 2.5.1, the diagonal entries $\lambda_0, \dots, \lambda_{n-1}$ of Λ are the eigenvalues of C , given explicitly by (2.1).

Since $\mathcal{F}_n$ is unitary, $\mathcal{F}_n^* \mathcal{F}_n = I$ and therefore $|\det(\mathcal{F}_n)| = 1$ (in particular, $\det(\mathcal{F}_n) \neq 0$). Taking determinants in the diagonalization $C = \mathcal{F}_n^* \Lambda \mathcal{F}_n$ gives

$$\det(C) = \det(\mathcal{F}_n^*) \det(\Lambda) \det(\mathcal{F}_n) = \det(\Lambda) = \prod_{k=0}^{n-1} \lambda_k.$$

(Indeed, for any square matrix the determinant equals the product of its eigenvalues; in the circulant case this is immediate from the explicit diagonalization.)

Proposition. $\det(C) = 0$ if and only if at least one $\lambda_k = 0$. Moreover, using (2.1), the $k = 0$ eigenvalue has the simple form

$$\lambda_0 = \sum_{j=0}^{n-1} c_j,$$

since for $k = 0$ we have $\omega_n^{jk} = \omega_n^0 = 1$ for all j , and the corresponding DFT eigen-

vector is the constant direction e .

Therefore, if $\sum_j c_j = 0$, then C is singular and e lies in $\ker(C)$. Moreover, every output of C lies in Layer 0, i.e., $\operatorname{im}(C) \subseteq L_0$, where

$$\operatorname{im}(C) := \{Cx : x \in \mathbb{R}^n\}.$$

Equivalently, $\operatorname{im}(C)$ is the set of all linear combinations of the columns of C . Indeed, when $\sum_j c_j = 0$ the rows of C sum to zero, so every vector Cx has coordinate sum zero and hence lies in L_0 .

Example. [A tiny singular circulant] Take $n = 3$ and $c = (1, -1, 0)^\top$. Then $\sum_j c_j = 0$, so $\lambda_0 = 0$ and $\det(\operatorname{Cir}(c)) = 0$. Indeed,

$$\operatorname{Cir}(c) = \begin{bmatrix} 1 & 0 & -1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}, \quad \operatorname{Cir}(c)e = 0, \quad e = (1, 1, 1)^\top.$$

The image of $\operatorname{Cir}(c)$ lies in $L_0 = \{x \in \mathbb{R}^3 : x_1 + x_2 + x_3 = 0\}$, so $\operatorname{im}(\operatorname{Cir}(c))$ has no component in the fixed direction $\operatorname{span}\{e\}$.

Eigenvectors and invariant subspaces. By definition, the eigenvalues λ and eigenvectors w of a matrix C satisfy

$$Cw = \lambda w.$$

In particular, if w is an eigenvector of C with eigenvalue λ , then its corresponding eigenspace is an invariant subspace of C ; see [1]. Indeed, for any $v \in \operatorname{span}\{w\}$ we

can write $v = \alpha w$ for some $\alpha \in \mathbb{R}$, and thus

$$C(v) = C(\alpha w) = \alpha C(w) = (\alpha \lambda) w \in \text{span}\{w\} \quad \text{for all } \alpha \in \mathbb{R}.$$

Remark. In the singular case, $\det(C) = 0$, so at least one eigenvalue of C is zero. For circulant matrices, the eigenvectors are the Fourier vectors $w^{(k)}$, and

$$Cw^{(k)} = \lambda_k w^{(k)}.$$

Thus, if $\lambda_k = 0$ for some k , then $Cw^{(k)} = 0$, meaning that the corresponding Fourier direction lies in the kernel $\ker(C)$ and is annihilated by C . Algebraically, this is equivalent to the cyclic shifts of c being linearly dependent; geometrically, the orbit polytope collapses along that Fourier direction. Equivalently, $\lambda_k = 0$ means that the corresponding Fourier eigenvalue

$$\lambda_k = \sum_{j=0}^{n-1} c_j \omega_n^{jk}$$

vanishes. In this case, every cyclic shift x of c satisfies $\langle w^{(k)}, x \rangle = 0$ (and hence, since the orbit polytope is the convex hull of these cyclic shifts, the orbit polytope is contained in that hyperplane).

Although the Fourier vectors $w^{(k)}$ are complex, in real space the vanishing of a Fourier mode is captured by requiring orthogonality to its cosine and sine parts, i.e., to $\Re(w^{(k)})$ and $\Im(w^{(k)})$. Consequently, for real orbit points x the condition $\langle w^{(k)}, x \rangle = 0$ is equivalent to the two real conditions $\langle \Re(w^{(k)}), x \rangle = 0$ and $\langle \Im(w^{(k)}), x \rangle = 0$. This is exactly the real formulation $x \in V_h = \ker \Delta(h, n)$ used in Subsection 2.7.1, where the associated real mode plane is $V_h^\perp = \text{span}(\Re(w^{(h)}), \Im(w^{(h)}))$. Thus, the orbit polytope is contained in the corresponding real invariant subspace $V_h = \ker \Delta(h, n)$.

In particular, when the zero eigenvalue occurs at the fixed mode ($k = 0$, corresponding to the direction e), the image of the circulant constraints lies in Layer 0, and the e -direction does not contribute additional information.

2.6 Real Fourier Vectors and the Discrete Fourier Transform

The Fourier-based constructions used in this section follow standard treatments of circulant matrices and discrete Fourier analysis [22].

As recalled in Section 2.5, circulant matrices are diagonalized by the discrete Fourier transform. We now extract from the complex DFT a convenient real family of vectors that will be used to define the subspaces appearing in later chapters.

Fix $n \in \mathbb{N}$ and let

$$\omega_n := e^{2\pi i/n}$$

be a primitive n -th root of unity. The *DFT matrix* $\mathcal{F}_n \in \mathbb{C}^{n \times n}$ is defined by

$$(\mathcal{F}_n)_{jk} = \frac{1}{\sqrt{n}} \omega_n^{-jk}, \quad j, k = 0, \dots, n-1.$$

The k -th column of $\mathcal{F}_n^*$ is

$$w^{(k)} = \frac{1}{\sqrt{n}} \left(1, \omega_n^k, \omega_n^{2k}, \dots, \omega_n^{(n-1)k} \right)^\top.$$

By the DFT diagonalization discussed in Section 2.5, the vectors $w^{(0)}, \dots, w^{(n-1)}$ form an orthonormal eigenbasis for every circulant matrix $C = \text{Cir}(c_0, \dots, c_{n-1})$; see

[1, Lemma 2.1.11]. More precisely,

$$Cw^{(k)} = \lambda_k w^{(k)}, \quad \lambda_k = \sum_{j=0}^{n-1} c_j \omega_n^{jk}, \quad k = 0, \dots, n-1.$$

2.6.1 Real Fourier Geometry and the $\Delta(h, n)$ Operators

The real Fourier representation plays a central role in the structure of circulant matrices and in the geometric separator developed in later chapters. For each integer frequency h the vectors $V(h, n), U(h, n) \in \mathbb{R}^n$ are given componentwise, for coordinate indices $k = 0, 1, \dots, n-1$, by

$$V(h, n)_k = \cos\left(\frac{2\pi hk}{n}\right), \quad U(h, n)_k = \sin\left(\frac{2\pi hk}{n}\right).$$

These vectors are the real and imaginary parts of the complex exponential $\exp\left(\frac{2\pi i h k}{n}\right)$, and together they describe the real Fourier modes of $\mathbb{R}^n$. Stacking them produces the operator

$$\Delta(h, n) = \begin{bmatrix} V(h, n)^\top \\ U(h, n)^\top \end{bmatrix},$$

a $2 \times n$ matrix whose kernel

$$V_h = \ker \Delta(h, n) = \{x \in \mathbb{R}^n : \Delta(h, n)x = 0\}.$$

defines the Fourier-invariant subspace associated with frequency h . We will discuss these invariant subspaces in Subsection 2.7.1.

We will only use frequencies h in the range $0 \leq h \leq \lfloor n/2 \rfloor$; higher frequencies are redundant, since it turns out that $V(n-h, n) = V(h, n)$ and $U(n-h, n) = -U(h, n)$.

Remark. [Index symmetry of $\Delta(h, n)$] The trigonometric definitions of $V(h, n)$ and

$U(h, n)$ imply a symmetry between the indices h and $n - h$. For any integer k we have

$$\cos\left(\frac{2\pi(n-h)k}{n}\right) = \cos\left(2\pi k - \frac{2\pi hk}{n}\right) = \cos\left(\frac{2\pi hk}{n}\right),$$

and

$$\sin\left(\frac{2\pi(n-h)k}{n}\right) = \sin\left(2\pi k - \frac{2\pi hk}{n}\right) = -\sin\left(\frac{2\pi hk}{n}\right),$$

since $\cos(2\pi k) = 1$ and $\sin(2\pi k) = 0$ for all integers k . Consequently,

$$V(n-h, n) = V(h, n), \quad U(n-h, n) = -U(h, n).$$

The corresponding Δ -matrices therefore differ only by a sign change in the second row:

$$\Delta(n-h, n) = \begin{bmatrix} V(n-h, n)^\top \\ U(n-h, n)^\top \end{bmatrix} = \begin{bmatrix} V(h, n)^\top \\ -U(h, n)^\top \end{bmatrix}.$$

Since multiplying a row of a matrix by -1 does not change its kernel, it follows that

$$\ker \Delta(h, n) = \ker \Delta(n-h, n).$$

As a result, it suffices to consider indices $0 \leq h \leq \lfloor n/2 \rfloor$ when enumerating distinct Δ -matrices and their kernels. This symmetry will be illustrated explicitly in the examples below.

Examples. We list the matrices $\Delta(h, n)$ for $n = 3$ and $n = 4$ in the explicit form used throughout this thesis. For each n , the case $h = 0$ has the same structure, so we illustrate it once for $n = 3$ and omit the similar example for $n = 4$. Since multiplying a row of $\Delta(h, n)$ by a nonzero scalar does not change its kernel, such rescalings may be applied to simplify the displayed matrices.

Example. $[\Delta(0, 3)$ and $\ker \Delta(0, 3)]$ The $h = 0$ mode gives the matrix

$$\Delta(0, 3) = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

Thus $\Delta(0, 3)x = 0$ is the single condition $x_1 + x_2 + x_3 = 0$, and a convenient basis for the kernel is

$$\ker \Delta(0, 3) = \text{span} \left\{ \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix} \right\}.$$

Example. $[\Delta(1, 3)$ and $\ker \Delta(1, 3)]$ The matrix for $h = 1$ is

$$\Delta(1, 3) = \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix}.$$

Solving $\Delta(1, 3)x = 0$ yields $x_1 = x_2 = x_3$, so

$$\ker \Delta(1, 3) = \text{span} \left\{ \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\}.$$

Example. $[\Delta(1, 4)$ and $\ker \Delta(1, 4)]$ The $h = 1$ mode produces

$$\Delta(1, 4) = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}.$$

Thus $x_1 = x_3$ and $x_2 = x_4$, and

$$\ker \Delta(1, 4) = \text{span} \left\{ \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \right\}.$$

Example. $[\Delta(2, 4)$ and $\ker \Delta(2, 4)]$ The matrix for $h = 2$ is

$$\Delta(2, 4) = \begin{bmatrix} 1 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Thus $\Delta(2, 4)x = 0$ gives $x_1 - x_2 + x_3 - x_4 = 0$, and one possible spanning set is

$$\ker \Delta(2, 4) = \text{span} \left\{ \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right\}.$$

Example. $[\Delta(3, 4)$ and $\ker \Delta(3, 4)]$ Using the symmetry relation, $\Delta(3, 4)$ is obtained from $\Delta(1, 4)$ by negating the second row. In particular,

$$\Delta(1, 4) = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}, \quad \Delta(3, 4) = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}.$$

Hence

$$\ker \Delta(3, 4) = \ker \Delta(1, 4) = \text{span} \left\{ \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \right\}.$$

2.6.2 The Trigonometric Matrix $R(n)$

The determinant arguments in Chapter 3 and Chapter 4 rely on a square trigonometric matrix built from the cosine and sine vectors that define the operators $\Delta(h, n)$.

Recall that for each integer frequency $h \in \{0, 1, \dots, \lfloor n/2 \rfloor\}$, the real Fourier vectors $V(h, n), U(h, n) \in \mathbb{R}^n$ are defined by

$$V(h, n)_k = \cos\left(\frac{2\pi hk}{n}\right), \quad U(h, n)_k = \sin\left(\frac{2\pi hk}{n}\right), \quad k = 0, 1, \dots, n-1.$$

For $h = 0$, we have $V(0, n) = e$ and $U(0, n) = 0$. If n is even, then $U(n/2, n) = 0$ as well. Therefore, sine rows occur only for $1 \leq h \leq \lfloor (n-1)/2 \rfloor$.

To obtain a square $n \times n$ matrix, we select exactly n real Fourier rows: the constant cosine row, paired cosine–sine rows for intermediate frequencies, and (when n is even) the highest-frequency cosine row.

If n is odd, we include $V(0, n)^\top$ and, for each $h = 1, \dots, (n-1)/2$, we include both $V(h, n)^\top$ and $U(h, n)^\top$. If n is even, we include $V(0, n)^\top$ and, for each $h = 1, \dots, n/2 - 1$, we include both $V(h, n)^\top$ and $U(h, n)^\top$, and finally we include the cosine row $V(n/2, n)^\top$.

We define matrix $R(n) \in \mathbb{R}^{n \times n}$ by stacking these rows.

This construction can be interpreted as sampling the real Fourier basis at equally spaced points on the unit circle.

Equivalently, $R(n)$ is the evaluation matrix of the real Fourier basis at the n sample angles

$$a_k = \frac{2\pi k}{n}, \quad k = 0, 1, \dots, n-1.$$

The k -th column of $R(n)$ records the values of these cosine and sine modes at a_k ; in particular, the sine rows take both positive and negative values as k varies around the circle. It will be shown in Chapter 4 (Subsection 4.2.2) that the nonsingularity of the matrix $R(n)$ plays a decisive role in guaranteeing the invertibility of the matrix $B = [F \ G]$ used in the cone separator.

Scott's determinant formula. We use Scott's determinant formula [12] to verify that $R(n)$ is non-singular. This formula establishes a closed-form expression for the determinant of a trigonometric matrix whose columns are evaluations of sine and cosine functions at arbitrary angles.

Given an angle vector $a = (a_1, \dots, a_{2n+1})$, the associated Scott matrix is

$$\text{Scott}(a_1, \dots, a_{2n+1}) = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \sin a_1 & \sin a_2 & \cdots & \sin a_{2n+1} \\ \cos a_1 & \cos a_2 & \cdots & \cos a_{2n+1} \\ \sin 2a_1 & \sin 2a_2 & \cdots & \sin 2a_{2n+1} \\ \cos 2a_1 & \cos 2a_2 & \cdots & \cos 2a_{2n+1} \\ \vdots & \vdots & \ddots & \vdots \\ \sin na_1 & \sin na_2 & \cdots & \sin na_{2n+1} \\ \cos na_1 & \cos na_2 & \cdots & \cos na_{2n+1} \end{bmatrix}.$$

Scott proved that

$$\det \text{Scott}(a_1, \dots, a_{2n+1}) = 2^{2n^2} \prod_{i>k} \sin\left(\frac{a_i - a_k}{2}\right), \quad (2.2)$$

which is nonzero precisely when no pair of angles a_i, a_k differ by a multiple of 2π . This identity provides a direct algebraic certificate that the matrix $\text{Scott}(a_1, \dots, a_{2n+1})$ has full rank whenever the angles $a_1, \dots, a_{2n+1}$ are distinct modulo 2π .

In the odd-dimensional case $n = 2k + 1$, the trigonometric matrix $R(n)$ coincides with a Scott matrix evaluated at the angle vector

$$a = \left(0, \frac{2\pi}{n}, \frac{4\pi}{n}, \dots, \frac{2\pi(n-1)}{n} \right).$$

This vector contains n distinct points on the unit circle, and no two differ by 2π because of the odd denominator.

Each difference between distinct entries is of the form

$$a_i - a_k = \frac{2\pi(i-k)}{n}, \quad i \neq k.$$

Since $i - k$ is a nonzero integer with $|i - k| \leq n - 1$, the largest possible magnitude of $a_i - a_k$ among distinct indices is

$$\max_{i \neq k} \left| \frac{2\pi(i-k)}{n} \right| = \frac{2\pi(n-1)}{n} < 2\pi.$$

Thus, for odd n , no two distinct angles differ by 2π (or any nonzero multiple of 2π), and all are distinct modulo 2π . This geometric separation ensures the Scott matrix with angle vector a is nonsingular. The first row of ones corresponds to the $V(0, n)$ cosine direct current component; subsequent sine/cosine pairs at $j = 1, \dots, k$ match $U(j, n)$ and $V(j, n)$, respectively. Therefore, the Scott matrix built from these angles is precisely $R(n)$,

$$\text{Scott}(a) \equiv R(n) \quad \text{for odd } n = 2k + 1,$$

and Scott's formula implies $\det R(n) \neq 0$, hence $R(n)$ is nonsingular. This fact is used in Chapter 4 (Subsection 4.2.2) to establish the linear independence of the columns of the matrix $B = [F \ G]$.

Dirichlet kernel and trigonometric identities. For even n , $R(n)$ is not exactly a Scott matrix because the highest-frequency sine row vanishes, so we use a standard trigonometric identity (the Dirichlet kernel) to handle the remaining determinant simplifications.

A classical identity from Fourier analysis, the Dirichlet kernel [11], plays an essential role in simplifying trigonometric sums. It is defined by

$$D_m(\theta) = 1 + 2 \sum_{k=1}^m \cos(k\theta).$$

This kernel admits the closed form

$$D_m(\theta) = \frac{\sin((m + \frac{1}{2})\theta)}{\sin(\theta/2)}.$$

The identity shows that trigonometric sums of the form $\sum_{k=1}^m \cos(k\theta)$ collapse to a single rational expression in sine functions.

In the determinant computation for even n , the following trigonometric cancellation must be verified when simplifying the cosine-sum entries of $R(n)$:

$$(-1)^m + 2 \sum_{j=1}^{\frac{n}{2}-1} \cos\left(\frac{2\pi jm}{n}\right) + 1 = 0. \quad (2.3)$$

Substituting the Dirichlet kernel identity with $\theta = \frac{2\pi m}{n}$ shows that (2.3) vanishes for all columns m . This cancellation justifies the construction of the special weight vector used in Chapter 4 (Subsection 4.2.2) for the analysis of the even-dimensional

matrix $R(n)$.

2.7 Δ -Matrices, Δ -Subspaces, and Projections

The Δ -matrix constructions presented here are derived from the theory of Fourier invariant-subspaces and follow standard spectral decompositions as in [22].

Section 2.6 constructed the real Fourier vectors $V(h, n), U(h, n)$ and the trigonometric matrix $R(n)$, and related them to the DFT eigenstructure of circulant matrices. In this section we use the operators $\Delta(h, n)$ built from $V(h, n), U(h, n)$ to define Fourier invariant-subspaces $V_h = \ker \Delta(h, n)$ and the corresponding orthogonal projectors $P(h, n)$ that are used later in the separator construction.

The real Fourier vectors $V(h, n)$ and $U(h, n)$ give rise to a family of simple row matrices that encode two linear conditions on $\mathbb{R}^n$. For each index h with $0 \leq h \leq \lfloor n/2 \rfloor$ we define the Δ -matrix

$$\Delta(h, n) := \begin{bmatrix} V(h, n)^\top \\ U(h, n)^\top \end{bmatrix} \in \mathbb{R}^{2 \times n}.$$

When n is even and $h = n/2$, the imaginary part $U(h, n)$ vanishes, and $\Delta(h, n)$ effectively has one nonzero row; in all other cases $\Delta(h, n)$ has rank 2.

For a vector $x \in \mathbb{R}^n$ the system $\Delta(h, n)x = 0$ is equivalent to the pair of orthogonality conditions

$$V(h, n)^\top x = 0 \quad \text{and} \quad U(h, n)^\top x = 0.$$

These linear conditions require that x be orthogonal to the two directions encoded by $V(h, n)$ and $U(h, n)$. Consequently, $\ker \Delta(h, n) = \{x \in \mathbb{R}^n : \Delta(h, n)x = 0\}$ consists of all vectors that are orthogonal to both $V(h, n)$ and $U(h, n)$; equivalently,

these are the vectors whose h -th cosine and sine Fourier coefficients vanish. This description is important for later constructions: it provides a simple linear test for membership in symmetry-invariant subspaces and serves as the basis for defining projections and separation directions in the multi-subspace framework developed in Chapters 3 and 4.

2.7.1 Invariant Subspaces defined by Δ -matrices

We now pass from the Δ -matrices themselves to the subspaces they define. The subspace constructions and projection operators developed in this section follow standard orthogonal decomposition techniques from linear algebra [13]. In particular, we use the Δ -matrices introduced earlier in Subsection 2.6.1 to define a family of invariant subspaces that extends the fixed-space and Layer 0 from Section 2.5.2.

Definition 2.7.1. *For each index h with $0 \leq h \leq \lfloor n/2 \rfloor$ the Δ -subspace V_h is defined by*

$$V_h := \ker \Delta(h, n) = \{x \in \mathbb{R}^n : \Delta(h, n)x = 0\}.$$

Thus $x \in V_h$ if and only if it is orthogonal to both $V(h, n)$ and $U(h, n)$. Each V_h is a linear subspace of $\mathbb{R}^n$.

The operators $\Delta(h, n)$ serve as explicit certificates of subspace membership, since

$$x \in V_h \iff \Delta(h, n)x = 0.$$

A direct justification for these conditions is given by the determinant factorization in [3]: the determinant of a circulant matrix can be written as a product over Fourier frequencies, and for each frequency h one scalar factor (denoted P_h in [3]) has the

form

$$P_h = \langle V(h, n), c \rangle^2 + \langle U(h, n), c \rangle^2,$$

where $c = (c_0, \dots, c_{n-1})^\top \in \mathbb{R}^n$ denotes the generating vector of the circulant matrix $C = \text{Cir}(c)$ (i.e., its first column).

Thus enforcing $\Delta(h, n)c = 0$ implies $\langle V(h, n), c \rangle = \langle U(h, n), c \rangle = 0$, hence $P_h = 0$, which means that the corresponding Fourier eigenvalue(s) are zero, so $\det(C) = 0$, and hence C is singular (not invertible) (see equations (7)–(8) in [3]).

In particular, when a circulant matrix has a zero eigenvalue at frequency h , then vectors aligned with the h -th cosine and sine directions are mapped to zero. Consequently, every output Cx has no component in that frequency, so $\text{im}(C) \subseteq V_h$. This explains the role of the subspaces V_h in the singular case discussed in Subsection 2.5.3.

Fixed space and Layer 0. We now summarize how the fixed space, Layer 0, and the Δ -subspaces fit together into a Fourier-type setting.

The zero-frequency case gives Layer 0. In particular, $V(0, n)$ is the constant vector (corresponding to e) and $U(0, n) = 0$, so

$$V_0 = \ker \Delta(0, n) = \{x \in \mathbb{R}^n : e^\top x = 0\}$$

is Layer 0 (dimension $n - 1$), while the fixed space is $\text{span}\{e\}$ (dimension 1).

The indices $h \geq 1$ define further Δ -subspaces associated with higher Fourier frequencies and are used later in Chapter 3. In the small cases $n = 2$ and $n = 3$, this next subspace is exactly the fixed space:

$$V_1 = \ker \Delta(1, n) = \text{span}\{e\}.$$

This is consistent with the dimension count below. When n is odd, for $1 \leq h \leq (n-1)/2$, the two rows of $\Delta(h, n)$ are independent (rank 2), so

$$\dim V_h = \dim \ker \Delta(h, n) = n - 2.$$

When n is even, the highest frequency $h = n/2$ is special: $U(n/2, n) = 0$, so $\Delta(n/2, n)$ provides only one independent cosine condition, leading to a different dimension count.

For nonzero frequencies $h \geq 1$, the cosine and sine vectors $V(h, n)$ and $U(h, n)$ are orthogonal to the constant vector e . This fact is proved in Lemma 2.7.1 and is used in the Hermitian-based construction of Chapter 3.

Invariant subspaces. The name “invariant” reflects the fact that these subspaces are preserved by the cyclic shift operator T_σ . Recall from Section 2.5.1 that the complex Fourier eigenvectors satisfy

$$T_\sigma W(h, n) = \omega(-h, n) W(h, n),$$

where

$$W(h, n) = V(h, n) - iU(h, n),$$

and

$$\omega(-h, n) = \cos\left(\frac{2\pi h}{n}\right) - i \sin\left(\frac{2\pi h}{n}\right).$$

Taking the real and imaginary parts of this identity yields

$$T_\sigma V(h, n) = \cos\left(\frac{2\pi h}{n}\right) V(h, n) - \sin\left(\frac{2\pi h}{n}\right) U(h, n),$$

and

$$T_\sigma U(h, n) = \sin\left(\frac{2\pi h}{n}\right) V(h, n) + \cos\left(\frac{2\pi h}{n}\right) U(h, n).$$

Hence the cyclic shift maps the plane

$$\text{span}\{V(h, n), U(h, n)\}$$

to itself. Consequently, the orthogonal complement

$$V_h = \ker \Delta(h, n)$$

is also invariant under cyclic shifts.

To verify this, let

$$x \in V_h = \ker \Delta(h, n).$$

By definition,

$$V(h, n)^\top x = 0, \quad U(h, n)^\top x = 0.$$

Since these orthogonality conditions are preserved by the cyclic shift, the shifted vector $T_\sigma x$ also satisfies

$$V(h, n)^\top (T_\sigma x) = 0, \quad U(h, n)^\top (T_\sigma x) = 0.$$

Therefore,

$$x \in V_h \implies T_\sigma x \in V_h,$$

showing that each Δ -subspace V_h is invariant under cyclic shifts of the coordinates. This is the relevant notion of invariance in the symmetric ILP setting, where the group action is given by coordinate permutations, as in Shahverdi's framework for symmetric ILPs [1].

2.7.2 Orthogonal Decomposition of $\mathbb{R}^n$ into Fourier Mode Subspaces

Subsection 2.6.1 introduced the operators $\Delta(h, n)$ and their geometric meaning in terms of the cosine and sine directions. Subsection 2.7.1 defined the Δ -subspaces $V_h = \ker \Delta(h, n)$ as Fourier-based invariant subspaces of $\mathbb{R}^n$. The purpose of the present subsection is to record the resulting orthogonal decomposition of $\mathbb{R}^n$ into Fourier mode subspaces, in the form needed later to define the projectors $P(h, n)$ used in the separator construction.

Recall from Section 2.7.1 that for each index h with $0 \leq h \leq \lfloor n/2 \rfloor$ we have the Δ -subspace

$$V_h = \ker \Delta(h, n) = \{x \in \mathbb{R}^n : V(h, n)^\top x = 0, U(h, n)^\top x = 0\},$$

the Fourier mode subspaces are defined by

$$V_h^\perp = \text{span}\{V(h, n), U(h, n)\}.$$

Since the kernel of a matrix equals the orthogonal complement of its row space, and the row space of $\Delta(h, n)$ is $\text{span}\{V(h, n), U(h, n)\}$, it follows that $V_h = (V_h^\perp)^\perp$, so the Δ -subspaces $V_h = \ker \Delta(h, n)$ are the orthogonal complements of these Fourier mode spaces.

To justify the orthogonal direct-sum decomposition of the Fourier mode spaces, we first prove the main orthogonality facts for the real Fourier vectors. We begin with the orthogonality of the nonzero real Fourier modes to the constant vector $e = (1, \dots, 1)^\top$, since this fact will also be used later in the Hermitian construction in Chapter 3. We then prove orthogonality between distinct nonzero real Fourier

modes. Since each subspace $V_h^\perp$ is generated by the cosine and sine vectors $V(h, n)$ and $U(h, n)$, it is enough to verify the corresponding orthogonality relations among these generators.

Lemma 2.7.1 (Orthogonality of nonzero real Fourier modes to the constant vector).

Let

$$e = (1, \dots, 1)^\top \in \mathbb{R}^n, \quad L_0 = \text{span}\{e\}^\perp = \{x \in \mathbb{R}^n : e^\top x = 0\}.$$

For a nonzero Fourier mode $h \in \{1, \dots, \lfloor n/2 \rfloor\}$, define

$$V(h, n)_k = \cos\left(\frac{2\pi hk}{n}\right), \quad U(h, n)_k = \sin\left(\frac{2\pi hk}{n}\right), \quad k = 0, 1, \dots, n-1.$$

Then

$$e^\top V(h, n) = 0 \quad \text{and} \quad e^\top U(h, n) = 0.$$

Equivalently,

$$V(h, n), U(h, n) \in L_0.$$

Proof. Set

$$\theta = \frac{2\pi h}{n}.$$

We restrict to nonzero Fourier modes because $h = 0$ gives the constant mode $V(0, n) = e$, which does not belong to L_0 , and also yields $\theta = 0$, for which the trigonometric identities used below are not applicable.

Since $h \in \{1, \dots, \lfloor n/2 \rfloor\}$, we have

$$\theta \not\equiv 0 \pmod{2\pi}.$$

Therefore, the following Lagrange trigonometric identities are applicable:

$$\sum_{k=0}^N \sin(k\theta) = \frac{\cos(\theta/2) - \cos((N + \frac{1}{2})\theta)}{2 \sin(\theta/2)}, \quad (2.4)$$

$$\sum_{k=1}^N \cos(k\theta) = \frac{-\sin(\theta/2) + \sin((N + \frac{1}{2})\theta)}{2 \sin(\theta/2)}. \quad (2.5)$$

These identities are standard; see [30].

We first prove that $U(h, n) \in L_0$. By definition,

$$e^\top U(h, n) = \sum_{k=0}^{n-1} \sin(k\theta).$$

Applying (2.4) with $N = n - 1$ gives

$$e^\top U(h, n) = \frac{\cos(\theta/2) - \cos((n - \frac{1}{2})\theta)}{2 \sin(\theta/2)}.$$

Now

$$(n - \frac{1}{2})\theta = (n - \frac{1}{2})\frac{2\pi h}{n} = 2\pi h - \frac{\pi h}{n} = 2\pi h - \frac{\theta}{2}. \quad (2.6)$$

Therefore

$$\cos((n - \frac{1}{2})\theta) = \cos\left(2\pi h - \frac{\theta}{2}\right) = \cos\left(\frac{\theta}{2}\right),$$

because $h \in \mathbb{Z}$ and $\cos(2\pi h - t) = \cos t$. Thus the numerator is zero. Also, since $1 \leq h \leq \lfloor n/2 \rfloor$, we have

$$0 < \theta = \frac{2\pi h}{n} \leq \pi,$$

so $0 < \theta/2 \leq \pi/2$, and therefore $\sin(\theta/2) \neq 0$. Hence

$$e^\top U(h, n) = 0.$$

Next we prove that $V(h, n) \in L_0$. Again by definition,

$$e^\top V(h, n) = \sum_{k=0}^{n-1} \cos(k\theta) = 1 + \sum_{k=1}^{n-1} \cos(k\theta).$$

Applying (2.5) with $N = n - 1$ yields

$$e^\top V(h, n) = 1 + \frac{-\sin(\theta/2) + \sin((n - \frac{1}{2})\theta)}{2 \sin(\theta/2)}.$$

Using (2.6), we obtain

$$\sin((n - \frac{1}{2})\theta) = \sin\left(2\pi h - \frac{\theta}{2}\right) = -\sin\left(\frac{\theta}{2}\right),$$

because $h \in \mathbb{Z}$ and $\sin(2\pi h - t) = -\sin t$. Therefore

$$e^\top V(h, n) = 1 + \frac{-\sin(\theta/2) - \sin(\theta/2)}{2 \sin(\theta/2)} = 1 - 1 = 0.$$

So both $V(h, n)$ and $U(h, n)$ lie in $L_0 = \text{span}\{e\}^\perp$ □

This also clarifies the connection with the fixed-space/Layer 0 split introduced in Section 2.5. The zero-frequency Fourier mode is the constant direction e , so it spans $\text{span}\{e\}$, while the nonzero real Fourier modes lie in $L_0 = \text{span}\{e\}^\perp$. Thus, from the Fourier viewpoint, circulant matrices preserve the constant direction and the zero-sum directions separately.

We now establish the orthogonality relations between distinct nonzero real Fourier modes.

Lemma 2.7.2 (Orthogonality of distinct nonzero real Fourier modes). *Let $V(h, n)$*

and $U(h, n)$ be the real Fourier vectors defined above, and let

$$V_h^\perp := \text{span}\{V(h, n), U(h, n)\}.$$

For any distinct $h, \ell \in \{1, \dots, \lfloor n/2 \rfloor\}$, the corresponding mode subspaces are orthogonal:

$$V_h^\perp \perp V_\ell^\perp.$$

Proof. Let

$$\alpha_k = \frac{2\pi h k}{n}, \quad \beta_k = \frac{2\pi \ell k}{n}, \quad k = 0, 1, \dots, n-1.$$

When n is even, the highest frequency $n/2$ is exceptional, because $U(n/2, n) = 0$. Accordingly, we first treat the ordinary frequencies $1 \leq h, \ell \leq \lfloor (n-1)/2 \rfloor$ with $h \neq \ell$, and then handle the case where one of the two frequencies equals $n/2$.

We first prove the orthogonality relations for the generators:

$$\begin{aligned} \langle V(h, n), V(\ell, n) \rangle &= 0, & \langle V(h, n), U(\ell, n) \rangle &= 0, \\ \langle U(h, n), V(\ell, n) \rangle &= 0, & \langle U(h, n), U(\ell, n) \rangle &= 0. \end{aligned}$$

Once these identities are established, the orthogonality $V_h^\perp \perp V_\ell^\perp$ follows by bilinearity of the inner product.

We will repeatedly use the zero-sum identities

$$\sum_{k=0}^{n-1} \cos\left(\frac{2\pi r k}{n}\right) = 0, \quad \sum_{k=0}^{n-1} \sin\left(\frac{2\pi r k}{n}\right) = 0 \quad \text{whenever } r \not\equiv 0 \pmod{n}. \quad (2.7)$$

These are exactly the zero-sum identities established in Lemma 2.7.1. Indeed, if a frequency r lies outside the standard range, we may replace it by an equivalent

representative modulo n . This is because replacing r by $r + mn$ changes the angle

$$\frac{2\pi rk}{n}$$

to

$$\frac{2\pi(r + mn)k}{n} = \frac{2\pi rk}{n} + 2\pi mk,$$

which differs by an integer multiple of 2π . Since sine and cosine are 2π -periodic, their values are unchanged. If needed, we may further replace r by $n - r$, because

$$\cos\left(\frac{2\pi(n - r)k}{n}\right) = \cos\left(\frac{2\pi rk}{n}\right), \quad \sin\left(\frac{2\pi(n - r)k}{n}\right) = -\sin\left(\frac{2\pi rk}{n}\right),$$

so the vanishing of the sums is unchanged.

We first compute $\langle V(h, n), V(\ell, n) \rangle$. Using the product-to-sum identity

$$\cos \alpha \cos \beta = \frac{1}{2}(\cos(\alpha - \beta) + \cos(\alpha + \beta)),$$

we obtain

$$\begin{aligned} \langle V(h, n), V(\ell, n) \rangle &= \sum_{k=0}^{n-1} \cos(\alpha_k) \cos(\beta_k) \\ &= \frac{1}{2} \sum_{k=0}^{n-1} \cos(\alpha_k - \beta_k) + \frac{1}{2} \sum_{k=0}^{n-1} \cos(\alpha_k + \beta_k) \\ &= \frac{1}{2} \sum_{k=0}^{n-1} \cos\left(\frac{2\pi(h - \ell)k}{n}\right) + \frac{1}{2} \sum_{k=0}^{n-1} \cos\left(\frac{2\pi(h + \ell)k}{n}\right). \end{aligned}$$

Since $h \neq \ell$, we have $h - \ell \not\equiv 0 \pmod{n}$. Also,

$$1 \leq h, \ell \leq \left\lfloor \frac{n-1}{2} \right\rfloor \implies 0 < h + \ell < n,$$

so $h + \ell \not\equiv 0 \pmod{n}$ as well. Therefore both cosine sums vanish by (2.7), equivalently

by Lemma 2.7.1. Hence

$$\langle V(h, n), V(\ell, n) \rangle = 0.$$

Next we compute $\langle V(h, n), U(\ell, n) \rangle$. Using

$$\cos \alpha \sin \beta = \frac{1}{2} (\sin(\alpha + \beta) - \sin(\alpha - \beta)),$$

we get

$$\begin{aligned} \langle V(h, n), U(\ell, n) \rangle &= \sum_{k=0}^{n-1} \cos(\alpha_k) \sin(\beta_k) \\ &= \frac{1}{2} \sum_{k=0}^{n-1} \sin(\alpha_k + \beta_k) - \frac{1}{2} \sum_{k=0}^{n-1} \sin(\alpha_k - \beta_k) \\ &= \frac{1}{2} \sum_{k=0}^{n-1} \sin\left(\frac{2\pi(h + \ell)k}{n}\right) - \frac{1}{2} \sum_{k=0}^{n-1} \sin\left(\frac{2\pi(h - \ell)k}{n}\right). \end{aligned}$$

Again $h + \ell \not\equiv 0 \pmod{n}$ and $h - \ell \not\equiv 0 \pmod{n}$, so both sine sums vanish by (2.7), equivalently by Lemma 2.7.1. Thus

$$\langle V(h, n), U(\ell, n) \rangle = 0.$$

Finally, for $\langle U(h, n), U(\ell, n) \rangle$, use

$$\sin \alpha \sin \beta = \frac{1}{2} (\cos(\alpha - \beta) - \cos(\alpha + \beta)).$$

Then

$$\begin{aligned}
\langle U(h, n), U(\ell, n) \rangle &= \sum_{k=0}^{n-1} \sin(\alpha_k) \sin(\beta_k) \\
&= \frac{1}{2} \sum_{k=0}^{n-1} \cos(\alpha_k - \beta_k) - \frac{1}{2} \sum_{k=0}^{n-1} \cos(\alpha_k + \beta_k) \\
&= \frac{1}{2} \sum_{k=0}^{n-1} \cos\left(\frac{2\pi(h - \ell)k}{n}\right) - \frac{1}{2} \sum_{k=0}^{n-1} \cos\left(\frac{2\pi(h + \ell)k}{n}\right).
\end{aligned}$$

Both cosine sums are zero by (2.7), equivalently by Lemma 2.7.1, so

$$\langle U(h, n), U(\ell, n) \rangle = 0.$$

By symmetry of the standard inner product on $\mathbb{R}^n$,

$$\langle U(h, n), V(\ell, n) \rangle = \langle V(\ell, n), U(h, n) \rangle.$$

Since the mixed cosine–sine case has already been proved for any distinct pair of frequencies, applying that result with h and ℓ interchanged gives

$$\langle V(\ell, n), U(h, n) \rangle = 0.$$

Hence

$$\langle U(h, n), V(\ell, n) \rangle = 0.$$

Now let

$$x = aV(h, n) + bU(h, n) \in V_h^\perp, \quad y = cV(\ell, n) + dU(\ell, n) \in V_\ell^\perp,$$

where $a, b, c, d \in \mathbb{R}$. Since the inner product is bilinear,

$$\begin{aligned}\langle x, y \rangle &= ac \langle V(h, n), V(\ell, n) \rangle + ad \langle V(h, n), U(\ell, n) \rangle \\ &\quad + bc \langle U(h, n), V(\ell, n) \rangle + bd \langle U(h, n), U(\ell, n) \rangle.\end{aligned}$$

Each of the four inner products on the right-hand side is zero, so

$$\langle x, y \rangle = 0.$$

Since $x \in V_h^\perp$ and $y \in V_\ell^\perp$ were arbitrary, it follows that

$$V_h^\perp \perp V_\ell^\perp.$$

It remains to consider the case where n is even and $\ell = n/2$. In this case,

$$U(n/2, n) = 0,$$

and therefore $U(n/2, n)$ is trivially orthogonal to all vectors.

Now let $1 \leq h \leq n/2 - 1$. Applying

$$\cos \alpha \cos \beta = \frac{1}{2} (\cos(\alpha - \beta) + \cos(\alpha + \beta))$$

gives

$$\langle V(h, n), V(n/2, n) \rangle = \frac{1}{2} \sum_{k=0}^{n-1} \cos\left(\frac{2\pi(h - n/2)k}{n}\right) + \frac{1}{2} \sum_{k=0}^{n-1} \cos\left(\frac{2\pi(h + n/2)k}{n}\right).$$

Since $1 \leq h \leq n/2 - 1$, neither $h - n/2$ nor $h + n/2$ is congruent to 0 (mod n).

Hence both cosine sums vanish by (2.7), so

$$\langle V(h, n), V(n/2, n) \rangle = 0.$$

Similarly, using

$$\sin \alpha \cos \beta = \frac{1}{2} (\sin(\alpha + \beta) + \sin(\alpha - \beta)),$$

we obtain

$$\langle U(h, n), V(n/2, n) \rangle = \frac{1}{2} \sum_{k=0}^{n-1} \sin\left(\frac{2\pi(h + n/2)k}{n}\right) + \frac{1}{2} \sum_{k=0}^{n-1} \sin\left(\frac{2\pi(h - n/2)k}{n}\right).$$

Again the corresponding frequencies are nonzero modulo n , so both sine sums vanish by (2.7). Therefore

$$\langle U(h, n), V(n/2, n) \rangle = 0.$$

Since $U(n/2, n) = 0$, we have

$$V_{n/2}^\perp = \text{span}\{V(n/2, n)\}.$$

The calculations above show that $V(n/2, n)$ is orthogonal to both $V(h, n)$ and $U(h, n)$ for every $1 \leq h \leq n/2 - 1$. Therefore

$$V_{n/2}^\perp \perp V_h^\perp \quad \text{for all } 1 \leq h \leq n/2 - 1.$$

Hence the highest-frequency mode subspace is orthogonal to all lower nonzero mode subspaces.

□

Orthogonal Direct-Sum Decomposition. The family of *Fourier mode subspaces*

$$V_h^\perp := \text{span}\{V(h, n), U(h, n)\}$$

forms a decomposition of $\mathbb{R}^n$ corresponding to the standard Fourier modes. Orthogonality of distinct mode subspaces follows from Lemma 2.7.2. Completeness follows from the full rank of the trigonometric matrix $R(n)$, established in Subsection 2.6.2. The rows of $R(n)$ are exactly the real Fourier vectors used to define these mode subspaces. Since $R(n)$ has full rank, these rows form a basis of $\mathbb{R}^n$. Therefore, the corresponding Fourier mode subspaces together span $\mathbb{R}^n$.

By construction, the mode subspaces $V_h^\perp$ satisfy the orthogonal direct-sum decomposition [24]

$$\mathbb{R}^n = \bigoplus_{h=0}^{\lfloor n/2 \rfloor} V_h^\perp = V_0^\perp \oplus V_1^\perp \oplus \cdots \oplus V_{\lfloor n/2 \rfloor}^\perp.$$

Here $V_0^\perp = \text{span}\{e\}$ is the fixed space, while $V_0 = \ker \Delta(0, n) = \{x \in \mathbb{R}^n : e^\top x = 0\}$ is Layer 0. For $h \geq 1$, the space $V_h^\perp$ is the component spanned by the h -th cosine-sine directions, and V_h is its orthogonal complement. In contrast, the Δ -subspaces

$$V_h = \ker \Delta(h, n)$$

are the *orthogonal complements* of these mode subspaces, i.e., $V_h = (V_h^\perp)^\perp$, and they are *not* intended to form a direct sum. (See Lemma 3.1.1)

We will use this decomposition as the organizing framework for the multi-subspace projectors defined in Subsection 2.7.3. This decomposition underlies the geometric separator in Chapter 3: the projection operators $P(h, n)$ used there are the orthogonal projectors onto the subspaces V_h , and the vectors $f_h = p - P(h, n)p$ measure the component of p lying outside V_h , and also these Δ -subspaces $V_h = \ker \Delta(h, n)$

are the nullspaces on which our separator is constructed to vanish.

2.7.3 Projections onto Δ -subspaces

Subsection 2.1.4 introduced projections and orthogonal projections in an abstract inner-product setting. We now specialize this notion to the Δ -subspaces $V_h = \ker \Delta(h, n)$.

Let $V \subseteq \mathbb{R}^n$ be a subspace of dimension d , and let $D \in \mathbb{R}^{n \times d}$ have columns forming a basis of V . As reviewed earlier, the orthogonal projection onto V is the linear map $P_V: \mathbb{R}^n \rightarrow \mathbb{R}^n$ given by

$$P_V = D(D^\top D)^{-1}D^\top.$$

This matrix is symmetric and idempotent ($P_V^\top = P_V$ and $P_V^2 = P_V$), with image equal to V . Equivalently, since D has full column rank, the matrix $(D^\top D)^{-1}D^\top$ is the Moore–Penrose pseudoinverse of D , and $P_V = DD^+$ is the orthogonal projection onto the subspace V . This is consistent with the characterization of orthogonal projections presented in Section 2.1.4, where orthogonal projections are precisely the self-adjoint projections.

Definition 2.7.2. *For each index h we fix a basis matrix D_h for the subspace $V_h = \ker \Delta(h, n)$ and define the orthogonal projection*

$$P(h, n) := D_h(D_h^\top D_h)^{-1}D_h^\top.$$

The operator $P(h, n)$ projects any vector $x \in \mathbb{R}^n$ onto the Δ -subspace V_h . In Chapter 3 and Chapter 4 we apply these projections to the candidate points p proposed by the ILP solver, and combine the residual vectors $p - P(h, n)p$ into a basis matrix $B(p)$ that reflects the underlying subspace structure. For the moment, we record

only that each $P(h, n)$ is an orthogonal projection onto V_h , with kernel $V_h^\perp$.

Accordingly, by the projection formula established in Section 2.1.4, the complementary projection

$$I_n - P(h, n)$$

is the orthogonal projection onto $V_h^\perp$.

2.8 Cones and Quadratic Forms for Symmetry-Based Separation

The geometric separator developed in Chapter 3 will take the form of a quadratic inequality describing a cone in a transformed coordinate system. We briefly recall the basic notions needed for that construction and connect them with the quadratic surfaces introduced in Definition 2.2.2. The geometric properties of cones and their representation via quadratic forms follow standard treatments in convex optimization [16] and matrix analysis [14].

Cones

Definition 2.8.1. *A set $K \subseteq \mathbb{R}^k$ is called a cone if it is closed under nonnegative scalar multiplication, that is,*

$$x \in K, \lambda \geq 0 \implies \lambda x \in K.$$

If, in addition, K is convex (so that $x, y \in K$ and $\alpha \in [0, 1]$ imply $\alpha x + (1 - \alpha)y \in K$), then K is called a convex cone.

Typical examples of convex cones include the nonnegative orthant $\mathbb{R}_{\geq 0}^k$ and the

second-order (or Lorentz) cone

$$\{(t, z) \in \mathbb{R} \times \mathbb{R}^{k-1} : \|z\|_2 \leq t\}.$$

In this thesis, however, the cone appears in a different role. The separator excludes points lying inside a cone around a distinguished direction, and the accepted region is the complement of that cone. This complement is generally nonconvex, so the resulting quadratic constraint is not a standard convex cone constraint (see subsection 2.8.1).

Quadratic forms

Definition 2.2.2 introduced general quadratic surfaces of the form $Q(x) = x^\top Ax + b^\top x + c$. The separators used later will be defined by *homogeneous* quadratic expressions, obtained by dropping the linear and constant terms.

Definition. *A real symmetric matrix $Q \in \mathbb{R}^{n \times n}$ is called positive semidefinite if*

$$x^\top Qx \geq 0 \quad \text{for all } x \in \mathbb{R}^n.$$

Equivalently, all eigenvalues of Q are nonnegative. In particular,

$$\det(Q) \geq 0.$$

Definition 2.8.2. *Let $Q \in \mathbb{R}^{n \times n}$ be a symmetric matrix. The associated quadratic form is the map $S: \mathbb{R}^n \rightarrow \mathbb{R}$ given by*

$$S(x) = x^\top Qx.$$

The level sets $\{x : S(x) = \gamma\}$ are special cases of the quadratic surfaces in Definition 2.2.2. When Q is positive semidefinite, the set $\{x : S(x) \leq 1\}$ is a convex region. However, in our construction the matrix Q need not have this property, and the resulting region is generally not convex.

In Chapter 3, we will construct a matrix Q from the subspace structure described above. In the two-subspace case, this leads to a normalized quadratic separator of the form

$$S(x) = x^\top Q x \leq \delta,$$

where $0 < \delta < 1$.

In the multi-subspace case, the separator is expressed as a cone inequality in the transformed coordinates,

$$y^\top Q_{\text{cone}} y \geq 0,$$

and is then pulled back to the original x -coordinates. In our construction, the resulting accepted region is generally not convex. These inequalities serve as quadratic separators: candidate points that violate the corresponding quadratic test are cut off by adding the associated constraint to the ILP.

2.8.1 Spherical cones and quadratic forms for separation

Spherical cones and angle constraints. The cone separator developed in Chapter 3 uses a geometric condition expressed in terms of the angle between vectors. For vectors $u, v \in \mathbb{R}^k$, the cosine of the angle between them is

$$\cos \angle(u, v) = \frac{u^\top v}{\|u\| \|v\|}.$$

A *spherical cone* is the set of all vectors making an angle at most ϕ with a fixed direction v ; equivalently,

$$u \in \mathcal{C}(v, \phi) \iff \frac{u^\top v}{\|u\| \|v\|} \geq \cos \phi.$$

Remark. In the separator construction developed later in this thesis, the cone itself represents the infeasible region. Candidate points lying inside the cone are cut off, while the accepted region is the complement of the cone.

In Chapter 3 (subsection 3.4.2), the compressed coordinates $y = MB^{-1}x$ satisfy a cone inequality of the form

$$(k-1)\|y\|^2 \geq c(y^\top e)^2,$$

where $e = (1, \dots, 1)^\top$. Dividing both sides by $\|y\|^2 \|e\|^2$ shows that this inequality is equivalent to

$$\frac{y^\top e}{\|y\| \|e\|} \leq \sqrt{\frac{k-1}{c \|e\|^2}},$$

which is an angle constraint between y and the barycentric direction e .

The quadratic form $Q_{\text{cone}} = (k-1)I - cee^\top$ therefore encodes, in algebraic form, a spherical cone whose interior is excluded by the separator. The boundary of this cone is a quadratic surface. Candidate points lying inside this cone are cut off, while points outside the cone are retained. This choice has implications for computational efficiency, which are discussed in Chapter 3.

Change of basis and transformation of quadratic forms. If $B \in \mathbb{R}^{n \times n}$ is an invertible matrix (not necessarily orthonormal), the change of coordinates

$$y = B^{-1}x$$

induces a corresponding transformation of any quadratic form $q(x) = x^\top Qx$.

Substituting $x = By$ gives

$$q(x) = (By)^\top Q(By) = y^\top (B^\top QB)y.$$

Thus the matrix representing the quadratic form in the y -coordinates is

$$Q_y = B^\top QB.$$

Conversely, if a quadratic inequality is expressed in the y -coordinates as

$$y^\top Q_y y \geq 0,$$

then the corresponding inequality in the original coordinates is

$$x^\top (B^{-\top} Q_y B^{-1}) x \geq 0.$$

This pullback of quadratic forms is central to the geometric separator constructed in Chapter 3, where the cone inequality in y -space is converted into a valid quadratic constraint in x -space via the matrix $Q'_{\text{cone}} = B^{-\top} M^\top Q_{\text{cone}} M B^{-1}$.

This chapter introduced the mathematical background required for the remainder of the thesis. We reviewed the linear-algebraic foundations, symmetry in integer linear programming, circulant matrices, and the Fourier-based constructions used to define the invariant subspaces considered throughout this work. We also introduced the projection operators and cone geometry that form the basis of the quadratic separator developed in the next chapter.

Chapter 3

Deriving Constraints

Symmetric integer programs produce large families of equivalent feasible points: if x is feasible and g is a symmetry (a coordinate permutation in the group G), then every point in the orbit $\{g \cdot x : g \in G\}$ is also feasible. While it is often possible to exclude an individual candidate point or a single orbit representative using a linear constraint, such constraints do not address the broader geometric structure created by symmetry. In this thesis, we focus on invariant subspaces such as the fixed space, Layer 0, and the Δ -subspaces $\ker(\Delta(h, n))$, since previous core-point approaches for cyclic symmetry showed that cyclic groups can have infinitely many core points, making the symmetry structure difficult to handle directly [1]. These invariant subspaces arise naturally in the broader symmetry-reduction framework developed for cyclic ILPs. This motivates the construction of a geometric separator adapted to these subspaces.

For cyclic and more general permutation symmetries, the ambient space $\mathbb{R}^n$ decomposes into a sum of invariant subspaces. In the simplest nontrivial case these are the fixed space $\text{Fix} = \text{span}\{e\}$ and the Layer 0 subspace $L_0 = \text{span}\{e\}^\perp$. In the more general Fourier-based setting of Chapter 2, additional invariant subspaces arise, such

as the Δ -subspaces $\ker(\Delta(h, n))$. Treating these invariant subspaces individually is expensive in practice, since a typical Δ -subspace is defined by two linear conditions and therefore reduces the dimension only from n to $n - 2$. This motivates the construction of a single separator that remains feasible on all relevant invariant subspaces simultaneously and rejects the candidate points that do not lie on any of them.

The geometric constructions developed in this chapter are motivated by the goal of generating quadratic separators for symmetric integer programs. By exploiting the structure of the invariant subspaces associated with cyclic symmetry, we construct quadratic inequalities that can later be added as nonlinear cuts in the cut-and-solve framework of Chapter 4. Thus, the subspace analysis developed here provides the geometric foundation for the separation procedure implemented in Chapter 4.

The separator also admits a disjunctive interpretation. In the two-subspace case, the separator is designed around two preserved subspaces: the Fixed space and Layer 0. Equivalently, a point belongs to one of the preserved subspaces if it lies in the Fixed space *or* in Layer 0. Similar disjunctive structures appear in symmetry-based formulations for cyclic ILPs [3] and in recent developments in disjunctive optimization and convex relaxations [8, 9]. Since this interpretation is not central to the construction or implementation of the separator, we do not pursue it further in this thesis.

The purpose of this chapter is to develop such a separator in progressively general stages:

- We begin with the two-subspace case (Fix and L_0), where all ideas can be expressed explicitly in two coordinates.
- We summarize the Hermitian-based attempt, which provided useful intuition

but led to cubic expressions that, while supported by some solvers, are expected to be less efficient in practice [25]. This motivates our focus on quadratic formulations, which are handled particularly effectively by SCIP due to specialized treatment of bilinear constraints and nonlinear relaxations [27, 28].

- We then construct the general multi-subspace separator using the $\Delta(h, n)$ operators introduced in Section 2.7.
- Finally, we show how the two-subspace separator is implemented in `Racket`, validated numerically for $n = 3, 4, 5, 207, 208, 209$, and how its geometric construction extends to the quadratic cuts used in SCIP in Chapter 4. The larger values of n are included to illustrate the behaviour of the construction in higher dimensions.

Implementation note (`Racket`). All numerical experiments and plots in this chapter were produced using a prototype implementation in `Racket` [26]. In particular, the two-subspace case (Fixed space and Layer 0) is implemented directly in `Racket`. Chapter 4 then translates the multi-subspace separator into a SCIP cut-generation routine.

Throughout the chapter the mathematical derivations are aligned with the implementation: the basis matrices $B(p)$, projections, compressed coordinates y , and quadratic forms $S(x)$ appear in both the theory and the code. The result is a single quadratic inequality adapted to the invariant-subspace structure, which accepts all relevant invariant subspaces in the construction and rejects the candidate point p only if it does not lie on any of them.

3.1 The Quadratic Separator

3.1.1 Setting and notation

Let $n \in \mathbb{N}$ and let $\{V_j\}_{j=0}^{k-1}$ be linear subspaces of $\mathbb{R}^n$. In our applications, these are concrete subspaces such as $\text{Fix} = \text{span}\{e\}$, $L_0 = \{x \in \mathbb{R}^n : e^\top x = 0\}$, and the Δ -subspaces $V_h = \ker \Delta(h, n)$. We write $d_j = \dim V_j$. In many places of this thesis, each V_j is invariant under a linear map T , that is, $T(V_j) \subseteq V_j$. This invariance explains why these subspaces arise naturally in the symmetry setting. However, the linear-algebraic constructions developed below—such as projections, orthogonal decompositions, and the associated quadratic forms—depend only on the subspace structure of the V_j and do not require invariance under T . We use the standard inner product $\langle x, y \rangle = x^\top y$, and the induced norm $\|x\|^2 = \langle x, x \rangle$. The j -th standard basis (elementary) vector is e_j .

In our circulant setting, the *Fourier mode subspaces* $V_h^\perp = \text{span}\{V(h, n), U(h, n)\}$ form an orthogonal direct-sum decomposition of $\mathbb{R}^n$. However, the separator construction in this thesis is based on the Δ -subspaces

$$V_h = \ker \Delta(h, n) = (V_h^\perp)^\perp,$$

which are used as large invariant kernel subspaces and are not intended to form a direct sum.

Indexing and notation. Throughout this chapter we let $k = \lfloor n/2 \rfloor + 1$, the number of distinct Δ -subspaces arising from the Fourier-mode construction. We use $j = 0, \dots, k-1$ to label these k subspaces in the separator construction. In the Fourier-mode setting, for each frequency index h we have the Δ -subspaces $V_h = \ker \Delta(h, n)$ from Chapter 2. Because $\ker \Delta(h, n) = \ker \Delta(n-h, n)$, the indices h

and $n - h$ correspond to the same Δ -subspace (see the remark in Subsection 2.6.1). Thus, to avoid listing the same subspace twice, we only use the distinct frequency indices $h = 0, 1, \dots, \lfloor n/2 \rfloor$. We keep the symbol h when referring to Fourier-mode objects such as $V(h, n)$, $U(h, n)$, and $\Delta(h, n)$.

Given a reference point $p \in \mathbb{R}^n$, we want to construct a *single* analytic constraint that accepts the union

$$\bigcup_{j=0}^{k-1} V_j$$

while excluding the point p .

For each subspace V_j , let P_j denote the orthogonal projector onto V_j and define

$$f_j := p - P_j p.$$

Then $f_j \in V_j^\perp$ for each j .

The construction only requires that the vectors $f_0, \dots, f_{k-1}$ are linearly independent, which we will ensure later via the rank of the matrix $R(n)$ (defined in Chapter 2, subsection 2.6.2).

To construct such a constraint, we proceed as follows.

1. Define the directions $f_j := p - P_j p$ using orthogonal projection onto each V_j .
2. Build a basis B aligned with these directions.
3. Compress any x to k coordinates y such that, whenever $x \in V_j$, the corresponding compressed coordinate satisfies $y_j = 0$. Hence the image of each invariant subspace is contained in a coordinate hyperplane (and, in the two-subspace case, in a coordinate axis).

4. Define a single separator that remains feasible on the union $\bigcup_{j=0}^{k-1} V_j$ and rejects a candidate point p that does not lie on any of them.

In Subsection 3.3, we single out the *Fixed space*

$$\text{Fix} = \text{span}\{e\}, \quad e = (1, \dots, 1)^\top,$$

and the *Layer 0* subspace

$$L_0 = \{x \in \mathbb{R}^n : e^\top x = 0\}.$$

These satisfy $\text{Fix} \perp L_0$ and $\mathbb{R}^n = \text{Fix} \oplus L_0$ (the special case $k = 2$ of the general framework below). Given a reference point $p \in \mathbb{R}^n$, its Layer 0 component is

$$f = p - \frac{e^\top p}{e^\top e} e,$$

and, we normalize e and f by setting $\hat{e} := e/\|e\|_2$ and $\hat{f} := f/\|f\|_2$, and then extend $\{\hat{e}, \hat{f}\}$ to an orthonormal basis $B = [\hat{e}, \hat{f}, g_1, \dots, g_{n-2}]$ using Gram–Schmidt. Hence $B^\top B = I_n$ and $B^{-1} = B^\top$.

The two–subspace construction provides the orthogonal decomposition

$$\mathbb{R}^n = \text{Fix} \oplus L_0.$$

In the general multi–subspace setting, however, the Δ -subspaces $V_j = \ker(\Delta(j, n))$ are generally not mutually orthogonal, even though their associated Fourier mode subspaces $V_j^\perp$ satisfy orthogonality properties established in Subsection 2.7.2. Consequently, the separator construction is no longer based on decomposing p into vectors lying in mutually orthogonal subspaces. Instead, for each subspace V_j , the vector $f_j = p - P_j p$ is defined as the residual obtained after projecting p onto V_j .

3.1.2 Subspace decomposition

For each j , let $P_j : \mathbb{R}^n \rightarrow V_j$ denote the orthogonal projector onto V_j (in our circulant setting, P_j coincides with $P(j, n)$); see Subsection 2.7.3. For the reference point p , we define the associated directions

$$f_j := p - P_j p \in V_j^\perp. \quad (3.1)$$

As basic consistency checks, we consider the k quantities

$$\|f_0\|, \|f_1\|, \dots, \|f_{k-1}\|,$$

together with

$$\left\| p - \sum_{j=0}^{k-1} f_j \right\|.$$

- **“The first k quantities are nonzero.”** Since $f_j = p - P_j p$, we have $\|f_j\| = 0$ if and only if $p \in V_j$. Thus $\|f_j\| > 0$ for all j certifies that $p \notin V_j$ for every j , and hence $p \notin \bigcup_{j=0}^{k-1} V_j$.
- **“The last quantity is zero.”** This says $p = \sum_j f_j$, i.e., p is exactly the sum of its mode components. In our setting, this reconstruction is justified using the real Fourier mode decomposition (Lemma 3.1.2), and it does *not* require the kernel spaces $V_j = \ker(\Delta(j, n))$ to form a direct sum.

Fourier-based justification of the reconstruction diagnostic. The identity

$$\left\| p - \sum_{j=0}^{k-1} f_j \right\| = 0$$

should be interpreted as a mathematical reconstruction diagnostic, not as an additional constraint imposed by the separator. It does not rely on a direct-sum decom-

position of the subspaces $V_j = \ker(\Delta(j, n))$. In general, these Δ -subspaces are large and may overlap, and consequently $\{V_j\}$ do not form a direct sum. Instead, the reconstruction identity is justified using the real Fourier decomposition: the *Fourier mode subspaces* $V_j^\perp$ (spanned by the real Fourier vectors for each frequency) form a direct sum, and we can reconstruct p by summing its mode components. The Fourier decomposition established in Subsection 2.7.2, together with the orthogonality relations from Lemma 2.7.2, can be re-expressed in terms of the block columns of the real Fourier basis matrix R . We record that block-matrix version here, since it is the form needed for the reconstruction map $F(p)$, whose columns isolate individual Fourier modes.

In Subsection 2.6.2, the matrix $R(n)$ was introduced by stacking the real Fourier vectors as rows. Here we use the equivalent column-block viewpoint, in which the same Fourier vectors are grouped into column blocks corresponding to the Fourier mode subspaces.

Lemma 3.1.1 (Direct sum of real Fourier mode subspaces). *Let $R \in \mathbb{R}^{n \times n}$ be the real Fourier basis matrix (as in Chapter 2, subsection 2.6.2), written in column blocks*

$$R = \begin{bmatrix} R(0) & R(1) & \cdots & R(k-1) \end{bmatrix},$$

where the columns of $R(j)$ form a basis for the j -th mode subspace $V_j^\perp \subseteq \mathbb{R}^n$. Then the family of subspaces $\{V_0^\perp, \dots, V_{k-1}^\perp\}$ forms a direct-sum decomposition of $\mathbb{R}^n$:

$$\mathbb{R}^n = V_0^\perp \oplus V_1^\perp \oplus \cdots \oplus V_{k-1}^\perp.$$

Equivalently, if $\sum_{j=0}^{k-1} v_j = 0$ with $v_j \in V_j^\perp$, then $v_j = 0$ for all j .

Proof. We argue by contradiction, expanding every step.

We use the standard uniqueness-of-representation argument based on the invertibility

of the basis matrix R .

Assume the subspaces $V_j^\perp$ do *not* form a direct sum. Then there exists a vector $x \in \mathbb{R}^n$ that admits two *distinct* decompositions as sums of vectors from these mode spaces:

$$x = \sum_{j=0}^{k-1} u_j = \sum_{j=0}^{k-1} u'_j, \quad u_j, u'_j \in V_j^\perp, \quad \text{and not all } u_j = u'_j.$$

Subtracting the two representations gives

$$\sum_{j=0}^{k-1} (u_j - u'_j) = 0.$$

Define

$$v_j := u_j - u'_j \in V_j^\perp.$$

Then

$$\sum_{j=0}^{k-1} v_j = 0 \quad \text{with not all } v_j = 0,$$

since the two decompositions were assumed to be distinct.

Now write the vectors u_j and u'_j in real Fourier coordinates. Since $V_j^\perp$ is spanned by the columns of $R(j)$, there exist coefficient vectors λ_j, λ'_j such that

$$u_j = R(j)\lambda_j, \quad u'_j = R(j)\lambda'_j.$$

Stack these block coefficients into vectors

$$\lambda := \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_{k-1} \end{bmatrix}, \quad \lambda' := \begin{bmatrix} \lambda'_0 \\ \lambda'_1 \\ \vdots \\ \lambda'_{k-1} \end{bmatrix}.$$

Because $R = [R(0) \ \cdots \ R(k-1)]$, we have

$$R\lambda = \sum_{j=0}^{k-1} R(j)\lambda_j = \sum_{j=0}^{k-1} u_j = x, \quad R\lambda' = \sum_{j=0}^{k-1} R(j)\lambda'_j = \sum_{j=0}^{k-1} u'_j = x.$$

Hence

$$R\lambda = R\lambda'.$$

Moreover $\lambda \neq \lambda'$, because if $\lambda = \lambda'$ then each $\lambda_j = \lambda'_j$, hence $u_j = R(j)\lambda_j = R(j)\lambda'_j = u'_j$ for all j , contradicting that the two decompositions are distinct.

From $R\lambda = R\lambda'$ we get $R(\lambda - \lambda') = 0$. Since R is invertible, $\lambda - \lambda' = 0$, i.e., $\lambda = \lambda'$, contradicting $\lambda \neq \lambda'$. Therefore the assumption was false, and the sum of the mode subspaces is direct.

In particular, applying this uniqueness to the vectors u_j, u'_j above, we must have $u_j = u'_j$ for every j . Hence $v_j = u_j - u'_j = 0$ for all j , which contradicts the earlier conclusion that not all v_j are zero. Therefore no nontrivial relation $\sum_{j=0}^{k-1} v_j = 0$ with $v_j \in V_j^\perp$ can exist. Equivalently, if $\sum_{j=0}^{k-1} v_j = 0$ with $v_j \in V_j^\perp$, then $v_j = 0$ for all j . $\square$

Mode-grouping and the reconstruction map $F(p)$. Let $y := R^{-1}p$ denote the real Fourier coefficient vector of p . The goal is to construct a matrix $F(p) \in \mathbb{R}^{n \times k}$ whose j -th column is the *single-mode component* of p in the mode subspace $V_j^\perp$, so that summing these columns reconstructs p .

Diagonal scaling. If $y = (y_0, y_1, \dots, y_{n-1})^\top$, then $\text{diag}(y)$ denotes the $n \times n$ diagonal matrix with entries y_i on the diagonal. Right multiplication by $\text{diag}(y)$ scales columns independently: the i -th column of $R \text{diag}(y)$ equals y_i times the i -th column of R . Thus $R \text{diag}(y)$ may be viewed as the real Fourier basis vectors weighted by the Fourier coefficients of p .

The grouping matrix M_{grp} . The real Fourier basis R contains one column for the constant mode, and for each nonzero frequency it contains a cosine–sine pair; when n is even there is also a final unpaired cosine mode. To treat each frequency as a *single* mode contribution, we group the corresponding Fourier-coordinate indices using the matrix M_{grp} . Let e_i denote the i -th standard basis vector in $\mathbb{R}^n$. Define

$$M_{\text{grp}} := \begin{cases} [e_0 \mid e_1 + e_2 \mid e_3 + e_4 \mid \cdots \mid e_{n-3} + e_{n-2} \mid e_{n-1}], & n \bmod 2 = 0, \\ [e_0 \mid e_1 + e_2 \mid e_3 + e_4 \mid \cdots \mid e_{n-2} + e_{n-1}], & n \bmod 2 = 1, \end{cases}$$

so each column of M_{grp} either selects a single Fourier-coordinate index (for unpaired modes) or selects the cosine–sine pair belonging to one frequency by summing the corresponding standard basis vectors.

Recall that any $x \in \mathbb{R}^n$ has a unique coordinate vector α in the real Fourier basis, meaning $x = R\alpha$ with $\alpha = R^{-1}x$. In Lemma 3.1.1 we denoted such coordinate vectors by λ ; here we apply the same idea to $x = p$ and write $y := R^{-1}p$.

Definition of $F(p)$. Now define

$$\boxed{F(p) := R \operatorname{diag}(y) M_{\text{grp}} = R \operatorname{diag}(R^{-1}p) M_{\text{grp}}.} \quad (3.2)$$

Intuitively: (i) $y = R^{-1}p$ extracts Fourier coefficients, (ii) $R \operatorname{diag}(y)$ scales each real Fourier basis vector by its coefficient, and (iii) M_{grp} groups those scaled basis vectors into mode blocks, so each resulting column contains only one Fourier mode.

Remark (used in Lemma 3.1.2). Let $e^{(k)} \in \mathbb{R}^k$ and $e^{(n)} \in \mathbb{R}^n$ denote the all-ones vectors in their respective spaces, where k is the number of columns of M_{grp} . By construction, each Fourier-coordinate index appears in exactly one column of M_{grp} , and therefore

$$M_{\text{grp}} e^{(k)} = e^{(n)}.$$

This identity simply expresses that summing all mode groups accounts for every Fourier coefficient exactly once.

Example ($n = 4$). For $n = 4$, the real Fourier basis matrix R has $n = 4$ columns, which may be ordered as the constant mode, the cosine–sine pair for frequency 1, and the final unpaired cosine mode. Equivalently, the associated Fourier-coordinate indices split into mode groups

$$\{0\}, \quad \{1, 2\}, \quad \{3\}.$$

The grouping matrix $M_{\text{grp}} \in \mathbb{R}^{4 \times 3}$ is defined to have one column per mode group, with a 1 in each row-index belonging to that group (and 0 elsewhere). Thus its columns are the selector vectors

$$e_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad e_1 + e_2 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}, \quad e_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

and stacking these gives

$$M_{\text{grp}} = [e_0 \mid e_1 + e_2 \mid e_3] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

If $e^{(3)} \in \mathbb{R}^3$ denotes the all-ones vector, then multiplying by $e^{(3)}$ sums the columns

of M_{grp} , so

$$M_{\text{grp}} e^{(3)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = e^{(4)}.$$

This illustrates that the grouping operation accounts for each Fourier coordinate exactly once: indices 1 and 2 (the cosine–sine pair) are merged into a single mode contribution, while indices 0 and 3 remain unpaired.

The next lemma shows that the matrix $F(p)$ decomposes p into its individual real Fourier mode components, and that for each fixed mode j , this decomposition is compatible with the orthogonal splitting $\mathbb{R}^n = V_j^\perp \oplus V_j$.

Lemma 3.1.2 (Columns of $F(p)$ isolate modes and reconstruct p). *Let $F(p)$ be defined by (3.2). Write its columns as*

$$F(p) = [f_0 \ f_1 \ \cdots \ f_{k-1}].$$

Then:

1. *For each j , the column f_j lies in the mode subspace $V_j^\perp$.*
2. *The reconstruction identity is*

$$F(p) e^{(k)} = p. \tag{3.3}$$

Consequently,

$$p = \sum_{j=0}^{k-1} f_j.$$

Moreover, for each fixed j , the partial sum of all other mode columns belongs to the

orthogonal complement of $V_j^\perp$, i.e.,

$$\sum_{i \neq j} f_i \in (V_j^\perp)^\perp = V_j.$$

Proof. By construction, each column of M_{grp} selects exactly the indices of the real Fourier coefficients corresponding to one mode. The j -th column of $\text{diag}(y)M_{\text{grp}}$ contains only the Fourier coefficients from mode j , and multiplying by R gives

$$f_j = R \text{diag}(y) M_{\text{grp}}(:, j).$$

Thus f_j is a linear combination of the columns in $R(j)$, and hence

$$f_j \in \text{span}(R(j)) = V_j^\perp.$$

This proves (1).

For (2), note that $M_{\text{grp}} e^{(k)} = e^{(n)}$, hence

$$F(p)e^{(k)} = R \text{diag}(y) M_{\text{grp}} e^{(k)} = R \text{diag}(y) e^{(n)} = Ry = p,$$

which is (3.3). Expanding $F(p)e^{(k)}$ as a column sum gives $p = \sum_j f_j$.

Finally, fix j . Since $P_j p$ is the orthogonal projection of p onto V_j , we have the orthogonal decomposition

$$p = w + P_j p, \quad \text{with } w \in V_j^\perp \text{ and } P_j p \in V_j.$$

On the other hand, by (3.3) we also have the mode decomposition

$$p = f_j + \sum_{i \neq j} f_i.$$

Here $f_j \in V_j^\perp$ by (1), since f_j is formed using only the basis vectors from the j -th real Fourier block (equivalently, f_j is a linear combination of the vectors in $\Delta(j, n)$, and $V_j^\perp := \text{span}(\Delta(j, n))$).

Now fix j and consider any $i \neq j$. By Lemma 3.1.1, the subspaces $\{V_h^\perp\}$ form a direct sum. In particular, no nonzero vector can lie in two different mode spaces at once. Since $f_i \in V_i^\perp$ and $i \neq j$, the direct-sum property of the subspaces $\{V_h^\perp\}$ implies that $V_i^\perp \cap V_j^\perp = \{0\}$. Therefore f_i cannot lie in $V_j^\perp$ unless $f_i = 0$ (which causes no difficulty, since the zero vector belongs to every subspace). Equivalently,

$$f_i \notin V_j^\perp \setminus \{0\}.$$

On the other hand, for any subspace $W \subseteq \mathbb{R}^n$ we have the orthogonal direct sum decomposition $\mathbb{R}^n = W \oplus W^\perp$. Taking $W = V_j^\perp$, then $\mathbb{R}^n = V_j^\perp \oplus (V_j^\perp)^\perp$ shows that every vector of $\mathbb{R}^n$ lies either in $V_j^\perp$ or in its orthogonal complement $(V_j^\perp)^\perp$. Since f_i is not a nonzero vector in $V_j^\perp$, it must lie in the orthogonal complement:

$$f_i \in (V_j^\perp)^\perp = V_j \quad \text{for all } i \neq j.$$

Therefore the sum of all other mode contributions lies in V_j ,

$$\sum_{i \neq j} f_i \in V_j.$$

Thus we have expressed the same vector p as a sum of a vector in $V_j^\perp$ and a vector in V_j in two different ways:

$$p = w + P_j p = f_j + \sum_{i \neq j} f_i, \quad w, f_j \in V_j^\perp, \quad P_j p, \sum_{i \neq j} f_i \in V_j.$$

Since for any subspace $W \subseteq \mathbb{R}^n$ we have the orthogonal direct sum decomposition

$$\mathbb{R}^n = W \oplus W^\perp,$$

it follows that the decomposition of any vector in $\mathbb{R}^n$ into a component in W and a component in $W^\perp$ is unique. Taking $W = V_j^\perp$ (so that $W^\perp = V_j$) gives

$$w = f_j, \quad P_j p = \sum_{i \neq j} f_i.$$

□

Remark. While the Fourier mode subspaces $\{V_j^\perp\}$ form a direct sum decomposition of $\mathbb{R}^n$, their orthogonal complements $\{V_j\}$ do not. Indeed, $V_j^\perp$ consists of the j -th Fourier mode only, whereas $V_j = (V_j^\perp)^\perp$ contains all remaining modes. Consequently, different choices of j lead to substantial overlap among the Δ -subspaces V_j . This distinction is crucial in the proof of Lemma 3.1.2, which relies on the direct-sum structure of the $V_j^\perp$ but not on any direct-sum property of the V_j .

Relation to the projector-based definition used in the code. The reconstruction identity $p = \sum_j f_j$ in Lemma 3.1.2 is a statement about *mode components* and relies on the direct-sum structure of the Fourier mode subspaces $V_j^\perp$ (Lemma 3.1.1), not on any direct-sum property of the Δ -subspaces $V_j = \ker(\Delta(j, n))$.

In the separator implementation, however, we retain the orthogonal projectors P_j onto V_j because they give the geometric interpretation needed for cut construction: for any reference point p ,

$$f_j := p - P_j p$$

is precisely the component of p orthogonal to the Δ -subspaces V_j , and $P_j p \in V_j$ is the component of p lying in the kernel $\ker(\Delta(j, n))$. While the Fourier-based construction

could also be used computationally, the projector-based implementation was retained for simplicity and geometric transparency. Lemma 3.1.2 is a structural reconstruction identity: it shows that p can be written as the sum of its mode components, $p = \sum_j f_j$. This follows purely from the Fourier-mode reconstruction and does not rely on any property of the Δ -subspaces $\{V_j\}$. Consequently, in the compressed coordinates used by the separator (see Subsection 3.4.3 and equation (3.26)), p corresponds to the all-ones vector $e^{(k)}$.

Remark. [Numerical considerations] The projector-based definition $f_j := p - P_j p$ and the Fourier-based construction via $F(p)$ are mathematically equivalent in exact arithmetic. In floating-point arithmetic, however, the two viewpoints may exhibit small numerical differences due to conditioning and rounding effects. In the present work, the Fourier-based viewpoint is used only for theoretical justification of the reconstruction identity, while the projector-based formulation is retained in the implementation for its geometric clarity.

In summary, we obtain a family of mode directions $f_0, \dots, f_{k-1} \in \mathbb{R}^n$ associated with the reference point p , where each f_j captures the contribution of p in the Fourier mode space $V_j^\perp$. We now use these directions as the leading columns of a basis matrix $B(p)$, which aligns the subsequent coordinate compression with the invariant subspace geometry.

3.1.3 Building the basis B

Choose nonzero *vectors* $f_0 \in V_0^\perp, \dots, f_{k-1} \in V_{k-1}^\perp$ (for example, take the f_j from (3.1) for the current p), and *complete* them to a full basis of $\mathbb{R}^n$:

$$B = [f_0 \ f_1 \ \cdots \ f_{k-1} \ g_k \ \cdots \ g_{n-1}] \in \mathbb{R}^{n \times n}, \quad \det B \neq 0. \quad (3.4)$$

(In our orthogonal two-subspace case, Fixed space and Layer 0, we take $\hat{e} := e/\|e\|_2$ and $\hat{f} := f/\|f\|_2$, and complete $\{\hat{e}, \hat{f}\}$ to an orthonormal basis B (e.g., via Gram–Schmidt). Then $B^\top B = I$ and $B^{-1} = B^\top$.)

As shown in Subsection 2.7.2, the real Fourier mode subspaces form an orthogonal direct-sum decomposition of $\mathbb{R}^n$; hence we may choose B orthogonal with its first k columns aligned with these mode directions.

Define the coordinate selector

$$M = \begin{bmatrix} I_k & 0_{k \times (n-k)} \end{bmatrix} \in \mathbb{R}^{k \times n}. \quad (3.5)$$

For any $x \in \mathbb{R}^n$, the coordinates in the B -basis are

$$c = B^{-1}x \in \mathbb{R}^n.$$

The first k entries of c are exactly the coefficients of x along the f_j -directions. Once we are in the B -coordinates, “project onto the first k axes” is trivial: we simply keep the first k entries of c and drop the rest. This is implemented by the selector matrix M , and motivates the *compression map*

$$y = Px := MB^{-1}x \in \mathbb{R}^k. \quad (3.6)$$

If we instead want the *projected vector back in the original* $\mathbb{R}^n$ coordinates, take those k coefficients, pad with zeros, and change back:

$$x_{\text{proj}} = B \begin{bmatrix} y \\ 0 \end{bmatrix} = B \begin{bmatrix} I_k & 0 \\ 0 & 0 \end{bmatrix} B^{-1}x. \quad (3.7)$$

Here the block matrix $\begin{bmatrix} I_k & 0 \\ 0 & 0 \end{bmatrix}$ (equivalently $\text{diag}(1, \dots, 1, 0, \dots, 0)$) keeps the first k coordinates and zeroes out the remaining $n - k$ coordinates.

In particular, Lemma 3.1.2 gives $p = \sum_{j=0}^{k-1} f_j$, so p lies in $\text{span}\{f_0, \dots, f_{k-1}\}$, and has no component in the remaining $g_k, \dots, g_{n-1}$ directions. Hence applying (3.7) to $x = p$ returns p itself:

$$p_{\text{proj}} = p.$$

3.1.4 Subspaces project to axis-orthogonal hyperplanes

In this section, we clarify how subspaces are represented in compressed coordinates, and why the appropriate geometric interpretation is in terms of axis-orthogonal hyperplanes.

Throughout Chapters 3 and 4 we work with the B -coordinates (coordinates of x in the basis B), defined by the invertible matrix $B \in \mathbb{R}^{n \times n}$:

$$y = B^{-1}x.$$

For any index j , the condition

$$y_j = 0$$

defines a hyperplane in the y -space orthogonal to the j -th coordinate direction. Consequently, when membership in a subspace forces a specific B -coordinate to vanish, the image of that subspace in compressed coordinates is contained in an axis-orthogonal hyperplane.

The geometric reason for this correspondence is the distance-to-subspace interpretation of orthogonal projection. Let P_j denote the orthogonal projection onto a

subspace V_j (with respect to the standard inner product). Then for every $p \in \mathbb{R}^n$,

$$\text{dist}(p, V_j) = \|p - P_j p\|_2,$$

and in particular $\|p - P_j p\|_2 = 0$ if and only if $p \in V_j$ [29, Fact 6.61]. In B -coordinates $y = B^{-1}x$, membership in a subspace naturally leads to the vanishing of a coordinate: when $x \in V_j$, the associated coordinate satisfies $y_j = 0$, which describes an axis-orthogonal hyperplane. This interpretation is compatible with the separator constructions used in this chapter.

Multi-subspace kernel setting. In the multi-subspace construction, the separator is built from the kernel subspaces

$$V_h = \ker \Delta(h, n),$$

which are large invariant subspaces. The compressed coordinates are chosen so that, for each index h , membership in V_h implies

$$y_h = 0.$$

Thus, for each h , the image of the Δ -subspace V_h in B -coordinates is contained in the axis-orthogonal hyperplane $y_h = 0$ (that is, $x \in V_h \Rightarrow y_h = 0$). This does not require a unique decomposition across Δ -subspaces: a vector may belong to several V_h 's, in which case several coordinates may vanish simultaneously.

Example. Consider B -coordinates $y = (y_0, y_1, y_2) \in \mathbb{R}^3$ defined by $y = B^{-1}x$. The condition $y_1 = 0$ defines the plane

$$\{(y_0, y_1, y_2) \in \mathbb{R}^3 : y_1 = 0\},$$

which is a two-dimensional hyperplane orthogonal to the y_1 direction.

Suppose that $V_1 = \ker \Delta(1, n)$ is one of the Δ -subspaces used in the separator construction, and that the B -coordinates are chosen so that membership in V_1 implies $y_1 = 0$. Then every vector $x \in V_1$ is mapped to a point lying in this plane (i.e., its B -coordinate vector $y = B^{-1}x$ satisfies $y_1 = 0$).

Unlike a coordinate axis, this hyperplane allows variation in the remaining coordinates (y_0, y_2) , reflecting the fact that the image of V_1 lies inside a higher-dimensional hyperplane rather than along a one-dimensional direction. Moreover, if a vector lies in the intersection of two Δ -subspaces, then in B -coordinates it lies in the intersection of the corresponding hyperplanes (e.g., $y_1 = 0$ and $y_2 = 0$), so no uniqueness of decomposition across the Δ -subspaces is needed.

Connection to the two-subspace case. The two-subspace setting consisting of the Fixed space and Layer 0 provides a concrete illustration of this principle. In this case, the matrix $B = [\hat{e}, \hat{f}, g_1, \dots, g_{n-2}]$ is chosen to be orthonormal ($B^{-1} = B^T$) with $\hat{e} := e/\|e\|_2$ as its first column. The B -coordinates are therefore

$$y = B^T x.$$

Since B is orthonormal, the coordinates are dot products with the basis vectors:

$$y_0 = \hat{e}^\top x, \quad y_1 = \hat{f}^\top x, \quad y_2 = g_1^\top x, \quad \dots, \quad y_{n-1} = g_{n-2}^\top x.$$

Each y_j measures the component of x in the corresponding basis direction.

Layer 0 is defined by the orthogonality condition

$$e^\top x = 0.$$

Since $y_0 = \hat{e}^\top x$, the first B -coordinate is

$$y_0 = \hat{e}^\top x = \frac{1}{\|e\|_2} e^\top x.$$

Consequently, $e^\top x = 0$ is equivalent to $y_0 = 0$, so Layer 0 is represented in B -coordinates by the axis-orthogonal hyperplane $y_0 = 0$.

Geometrically, this corresponds to a hyperplane orthogonal to the y_0 direction. In the two-dimensional visualizations used later in this chapter, this hyperplane appears as a coordinate axis, reflecting the low-dimensional setting rather than a distinct geometric principle.

In contrast, the Fixed space $\text{span}(e)$ consists of vectors that point entirely in the direction e and have no component in any orthogonal direction. If $x = \alpha e$ lies in the Fixed space, then it has zero component along $\hat{f}$ and along each g_i , and therefore all B -coordinates except the first vanish:

$$y_1 = y_2 = \cdots = y_{n-1} = 0.$$

Each condition $y_j = 0$ defines an axis-orthogonal hyperplane in B -coordinates, and the Fixed space is represented as the intersection of these hyperplanes.

3.1.5 Separators in the compressed coordinates

Our goal is to construct a separator $S : \mathbb{R}^n \rightarrow \mathbb{R}_{\geq 0}$ that vanishes on the union of invariant subspaces but takes the value 1 at the reference point p , namely

$$S(x) = 0 \text{ for all } x \in \bigcup_j V_j, \quad S(p) = 1.$$

With this normalization, the inequality

$$\boxed{S(x) \leq \delta}, \quad \delta \in (0, 1), \quad (2)$$

keeps every V_j feasible (since $S(x) = 0$ for every $x \in V_j$) while excluding p (since $S(p) = 1 > \delta$).

To design such an S , we use the compression map $y = Px$ (with $P = MB^{-1}$). Under this map, each V_j maps into the coordinate hyperplane $y_j = 0$, so $\bigcup_j V_j$ maps into the union of these axis-orthogonal hyperplanes. We therefore design a function $\tilde{S}$ in the y -space that is 0 on these coordinate hyperplanes and equals 1 at $y = Pp$, and then lift the separator back to $\mathbb{R}^n$ by setting $S(x) := \alpha \tilde{S}(Px)$ and choosing α so that $S(p) = 1$.

Specialization to our *two orthogonal* invariant subspaces. Let $k = 2$. Recall that in the multi-subspace setting we define $V_h := \ker \Delta(h, n)$. Since $\Delta(0, n) = e$, it follows that

$$V_0 = \ker \Delta(0, n) = \{x : e^\top x = 0\} = e^\perp,$$

which is Layer 0, and the other invariant subspace is the Fixed space

$$V_1 = \text{span}\{e\}.$$

With $B = [\hat{e}, \hat{f}, g_1, \dots, g_{n-2}]$ chosen orthonormal, we have $B^{-1} = B^\top$. Then $y = (y_0, y_1)^\top := Px$ are the first two B -coordinates of x (with $P = MB^{-1}$), and we use the simplest symmetric bilinear form on (y_0, y_1) ,

$$Q_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

which defines the quadratic separator

$$\tilde{S}(y) := y^\top Q_2 y = 2y_0 y_1.$$

This vanishes on the two axes ($y_0 = 0$ or $y_1 = 0$), which correspond to Layer 0 and the Fixed space in the two-subspace case. At the reference point p we define its compressed coordinates by

$$y(p) := Pp = (y_0(p), y_1(p))^\top.$$

Since $\tilde{S}(y) = 2y_0y_1$ vanishes whenever $y_0 = 0$ or $y_1 = 0$, we have $\tilde{S}(y) = 0$ on the two axes. In the two-subspace case these axes correspond to Layer 0 ($y_0 = 0$) and the Fixed space ($y_1 = 0$). Hence, if $p \notin \text{Fix} \cup L_0$, then both coordinates $y_0(p)$ and $y_1(p)$ are nonzero and

$$\tilde{S}(y(p)) = 2y_0(p)y_1(p) \neq 0,$$

with sign determined by whether $y_0(p)$ and $y_1(p)$ have the same or opposite signs.

We lift this separator back to the original variables by setting

$$S(x) := \alpha \tilde{S}(Px),$$

and choose the scaling factor α so that $S(p) = 1$. With this normalization, the

inequality

$$S(x) \leq \delta \quad (0 < \delta < 1)$$

cuts off the reference point p (since $S(p) = 1 > \delta$), while keeping both invariant subspaces feasible (since $S(x) = 0$ for all $x \in \text{Fix} \cup L_0$).

The two-subspace construction above is a special case of the general k -subspace separator framework used throughout this chapter. In summary, we (i) define the directions $f_j := p - P_j p$ using orthogonal projection onto each V_j ; (ii) build a basis B aligned with the subspaces and compress by $P = MB^{-1}$ to a k -coordinate summary; (iii) show that each V_j maps into the coordinate hyperplane $\{y \in \mathbb{R}^k : y_j = 0\}$ (which appears as a coordinate axis only in the two-subspace (2D) visualization; see Subsection 3.1.4); and (iv) design separators in y -space that vanish on the union of these coordinate hyperplanes and, after normalization, cut off p via (2). In the special case $k = 2$ (Fixed space and Layer 0) this recovers the quadratic form $S(x) = 2y_0y_1$ with $y = Px$, consistent with the constructions already used in this chapter.

3.2 Hermitian-Based Separator

Before arriving at the quadratic separator used throughout this thesis, our initial experiments in **Racket** explored an alternative construction based on Hermitian matrices. This was motivated by the spectral structure of circulant matrices, whose eigenvectors are generally complex. To work in real space while retaining spectral information, we built a real symmetric (hence Hermitian) matrix H from the real and imaginary parts of a complex eigenvector, so that the expression $x^\top Hx$ is a real-valued quadratic form.

To obtain a separator that vanishes on both Layer 0 (where $e^\top x = 0$) and the Fixed space, this quadratic factor was combined with the linear factor $e^\top x$, leading to the

cubic separator

$$Q(x) = (e^\top x)(x^\top Hx).$$

Although this method successfully produced a separator whose zero set aligns with both the Fixed space and Layer 0, its cubic degree made solver integration more inconvenient. We therefore treat this approach as a historical attempt: conceptually valuable, but replaced by our quadratic separator.

3.2.1 Properties of Hermitian Matrices

A matrix $H \in \mathbb{C}^{n \times n}$ is called *Hermitian* if it equals its conjugate transpose,

$$H = H^*,$$

where $H^* = \overline{H}^\top$. When H is real, this condition reduces to symmetry ($H = H^\top$).

Hermitian matrices play a crucial role in our early separator construction because they guarantee that quadratic forms of the type $x^* H x$ are real. We use the following standard properties of Hermitian matrices; see, for example, [14].

- For any $x \in \mathbb{C}^n$ and Hermitian matrix H , the quadratic form $x^* H x$ is real.
- If H is positive semidefinite, then

$$x^* H x \geq 0$$

for all $x \in \mathbb{C}^n$.

- For any real vector $y \in \mathbb{R}^n$, the outer product $yy^\top$ is symmetric and positive semidefinite.

Remark. [Positive semidefiniteness criterion] A Hermitian matrix is positive semidefinite if and only if all its eigenvalues are nonnegative, equivalently if $x^* H x \geq 0$ for all x .

These properties justify the Hermitian construction in Section 3.2. By decomposing a complex eigenvector $Y \in \mathbb{C}^n$ into its real and imaginary parts,

$$Y = Y_{\text{real}} + iY_{\text{imag}},$$

and forming

$$H = Y_{\text{real}} Y_{\text{real}}^\top + Y_{\text{imag}} Y_{\text{imag}}^\top,$$

we obtain a real symmetric matrix. Since each outer-product term is positive semidefinite, the matrix H is positive semidefinite, and therefore

$$x^\top H x \geq 0$$

for all real vectors x . In particular, the quadratic form used in the Hermitian-based separator is always real and nonnegative.

3.2.2 Real–Imaginary Decomposition and Construction of the Hermitian Matrix

Eigenvectors of circulant and Fourier-based matrices are often complex-valued. If we were to use such a complex eigenvector $Y \in \mathbb{C}^n$ directly, expressions like $x^\top Y$ or $Y^\top x$ could be complex, and the resulting quadratic form might not remain real-valued.

To avoid this, we decompose Y into its real and imaginary components:

$$Y = Y_{\text{real}} + i Y_{\text{imag}}, \quad Y_{\text{real}}, Y_{\text{imag}} \in \mathbb{R}^n.$$

With this decomposition, we work entirely in real space and define the Hermitian matrix used in our formulation. Given a complex eigenvector

$$Y = Y_{\text{real}} + i Y_{\text{imag}}, \quad Y_{\text{real}}, Y_{\text{imag}} \in \mathbb{R}^n,$$

we set

$$H = Y_{\text{real}} Y_{\text{real}}^\top + Y_{\text{imag}} Y_{\text{imag}}^\top. \quad (3.8)$$

This construction has three immediate guarantees:

- Each outer product $yy^\top$ with a real vector y is symmetric and positive semidefinite.
- Hence H is symmetric (and therefore Hermitian).
- For any real vector x ,

$$x^\top H x = (Y_{\text{real}}^\top x)^2 + (Y_{\text{imag}}^\top x)^2 \in \mathbb{R}_{\geq 0}.$$

Thus the quadratic form is always real and nonnegative, independent of the complex phases of Y .

Equation (3.8) therefore provides a principled way to translate complex spectral information into a real quadratic form suitable for our separator construction.

3.2.3 Definition of the Hermitian-Based Separator

With the all-ones vector $e = (1, \dots, 1)^\top$, we define a Hermitian-based *separator* by combining the quadratic form $x^\top H x$ with the linear factor $e^\top x$:

$$Q(x) = (e^\top x) (x^\top H x). \quad (3.9)$$

Note that $Q(x)$ is *not* itself a quadratic form: it is a cubic function built from the quadratic factor $x^\top Hx$.

The role of the factor $e^\top x$ is to force the separator to vanish on Layer 0. Indeed, if $x \in L_0$, then $e^\top x = 0$, and therefore $Q(x) = 0$ automatically. If we were to omit this factor and use only $x^\top Hx$, then the resulting expression would remain real and nonnegative, but it would no longer vanish on all of Layer 0 in general.

The quadratic factor $x^\top Hx$ encodes the spectral direction used to build H . Since

$$H = Y_{\text{real}} Y_{\text{real}}^\top + Y_{\text{imag}} Y_{\text{imag}}^\top,$$

we have

$$x^\top Hx = (Y_{\text{real}}^\top x)^2 + (Y_{\text{imag}}^\top x)^2 \geq 0,$$

and $x^\top Hx = 0$ holds exactly when $Y_{\text{real}}^\top x = 0$ and $Y_{\text{imag}}^\top x = 0$.

In the circulant/Fourier setting, if Y is chosen from a nonzero Fourier mode, then its real and imaginary parts are orthogonal to the constant vector e ; see Lemma 2.7.1.

In particular, for

$$Y_{\text{real}} = V(h, n), \quad Y_{\text{imag}} = U(h, n),$$

we have

$$Y_{\text{real}}^\top e = 0, \quad Y_{\text{imag}}^\top e = 0.$$

Hence, for any $x = \alpha e \in \text{Fix}$,

$$Y_{\text{real}}^\top x = \alpha Y_{\text{real}}^\top e = 0, \quad Y_{\text{imag}}^\top x = \alpha Y_{\text{imag}}^\top e = 0,$$

so

$$x^\top Hx = 0.$$

Consequently,

$$Q(x) = (e^\top x)(x^\top Hx) = 0 \quad \text{for all } x \in \text{Fix}.$$

Thus $Q(x)$ vanishes on both Layer 0 and the Fixed space.

In particular, since

$$x^\top Hx = (Y_{\text{real}}^\top x)^2 + (Y_{\text{imag}}^\top x)^2,$$

we have $x^\top Hx > 0$ whenever

$$Y_{\text{real}}^\top x \neq 0 \quad \text{or} \quad Y_{\text{imag}}^\top x \neq 0.$$

Now choose a reference point p such that

$$e^\top p \neq 0 \quad \text{and} \quad p^\top Hp > 0.$$

Then

$$Q(p) = (e^\top p)(p^\top Hp) \neq 0.$$

Since $Q(x) = 0$ for all $x \in L_0 \cup \text{Fix}$, this shows that the Hermitian-based separator vanishes on the two invariant subspaces while taking a nonzero value at the reference point p . Hence the subspaces lie in the zero set $Q(x) = 0$, while the reference point p does not. In an optimization setting, the inequality direction can then be chosen according to the sign of $Q(p)$ (equivalently, according to the sign of $e^\top p$, since $p^\top Hp > 0$). Thus, if $Q(p) > 0$, one uses the side $Q(x) \geq 0$, whereas if $Q(p) < 0$, one uses the side $Q(x) \leq 0$. Because Q is cubic, however, this approach is less convenient to integrate into the solver than the quadratic separator used in the remainder of the chapter.

Geometric and analytical interpretation. The surface $Q(x) = 0$ is the union of the sets where either factor in (3.9) vanishes:

$$e^\top x = 0 \quad (\text{Layer 0}), \quad \text{or} \quad x^\top Hx = 0 \quad (Y_{\text{real}}^\top x = 0 \text{ and } Y_{\text{imag}}^\top x = 0).$$

When Y is chosen from a nonzero Fourier mode, Lemma 2.7.1 shows that Y_{real} and Y_{imag} are orthogonal to e , so the zero set contains both Layer 0 and the Fixed space $\text{span}\{e\}$. Figure 3.1 shows a typical surface produced by our `Racket` prototype.

The Hermitian construction successfully carves out the Fixed space and Layer 0 subspaces while isolating the candidate point p .

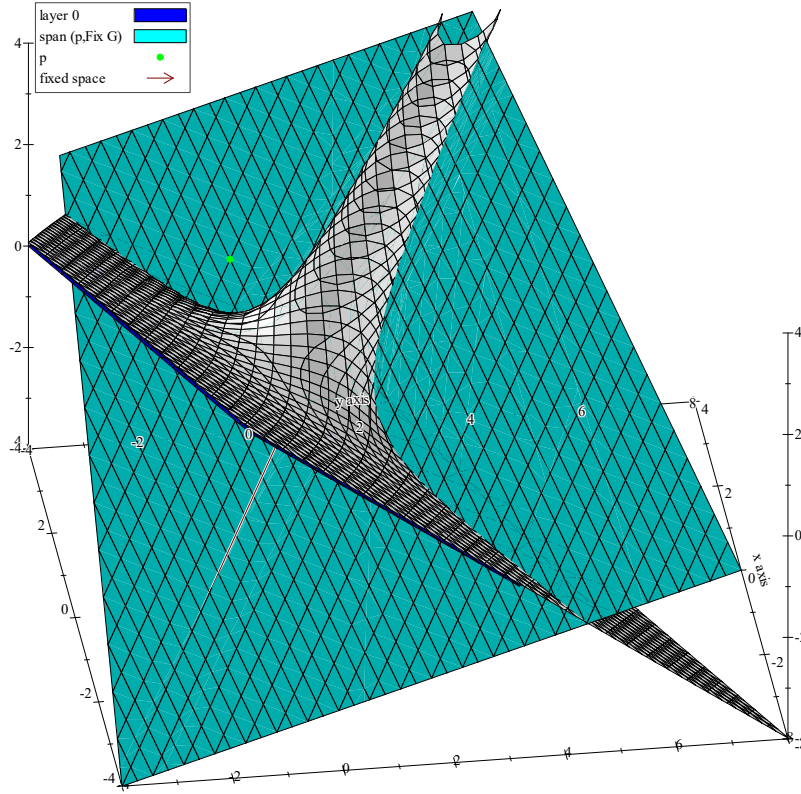


Figure 3.1: Hermitian-based cubic separator $Q(x) = (e^\top x)(x^\top Hx)$ (generated in `Racket`). The function $Q(x)$ vanishes on the Fixed space $\text{span}\{e\}$ and on Layer 0 ($e^\top x = 0$), and is used to cut off the candidate point p . This cutting behavior can eliminate multiple “corner” points at once in small illustrative linear-programming examples with only a few vertices, but it comes at the cost of increased algebraic degree.

Advantages and limitations. The Hermitian formulation has two clear advantages:

- **Guaranteed reality.** By construction, $x^\top Hx$ is always real, even though Y is complex.
- **Strong cutting power.** In experiments, Hermitian-based cuts could eliminate entire sets of infeasible candidates simultaneously, which is useful for corner cases.

However, the approach also has limitations:

- The separator $Q(x)$ is *cubic*, complicating integration with solvers that are optimized for quadratic constraints. In a multi-subspace setting, enforcing vanishing on additional subspaces would require additional factors (corresponding to further eigenvector or mode directions), increasing the algebraic degree and making the resulting constraints more difficult to analyze and integrate into the solver.
- The interpretation of the geometry becomes less transparent: level sets are higher-degree algebraic varieties, harder to visualize and analyze systematically.
- Numerical stability may also become more challenging in higher dimensions, since cubic expressions involve higher-degree interactions than quadratic forms.

Motivation for switching to the quadratic form. Because of these disadvantages, particularly the cubic degree of $Q(x)$, we transitioned to a quadratic separator of the form

$$S(x) = x^\top Q'x, \quad Q' = BQB^\top.$$

In the two-subspace case $k = 2$, this separator has the simple product form in compressed coordinates $y = Px = (y_0, y_1)$:

$$S(x) = 2y_0y_1.$$

More generally (for k subspaces), the separator remains a quadratic form that is designed to vanish on the union of coordinate hyperplanes $y_j = 0$ and can be normalized so that $S(p) = 1$ at the reference point p . This quadratic structure has lower algebraic degree, a clearer geometric interpretation, and integrates more directly into solver frameworks that natively support quadratic constraints, which motivates the switch from the Hermitian (cubic) construction to the quadratic formulation.

3.3 The Two-Subspace Quadratic Separator

After exploring the Hermitian-based approach, we now turn to the quadratic separator that forms the backbone of the rest of this thesis. The goal is to build a simple quadratic function that vanishes exactly on the two invariant subspaces Layer 0 and the Fixed space, while satisfying $S(p) \neq 0$ for the candidate point p . In our implementation we then normalize by a scalar factor and work with $\hat{S}(x) = S(x)/S(p)$, so that $\hat{S}(p) = 1$.

This section corresponds to the two-subspace **Racket** prototype: we implement the basis construction and the separator evaluation in **Racket** and use it for the numerical checks and visualizations in this chapter. At the heart of the construction lies the 2×2 matrix

$$Q_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

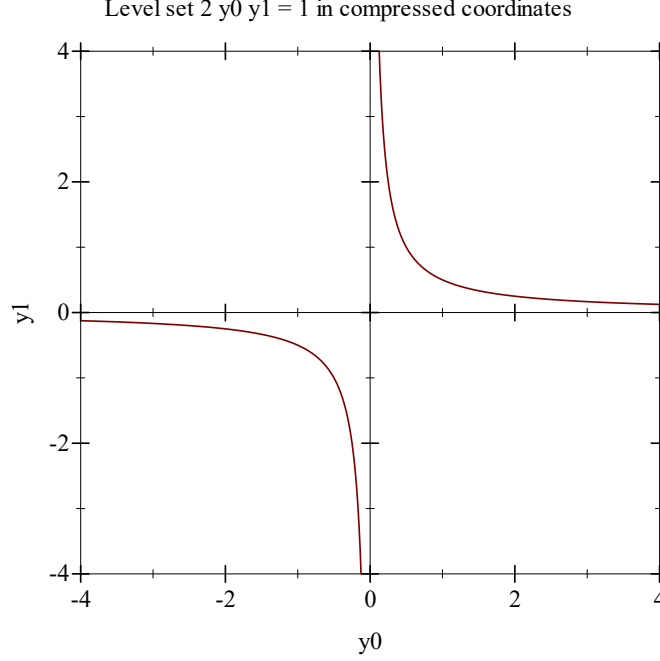


Figure 3.2: Level set $2y_0y_1 = 1$ in the compressed coordinates $y = Px = (y_0, y_1)^\top$. The function $2y_0y_1$ is zero exactly on the coordinate axes $y_0 = 0$ and $y_1 = 0$. In the two-subspace setting, these axes correspond to Layer 0 and the Fixed space, so the function vanishes precisely on those two subspaces.

This is the simplest symmetric bilinear form that couples two coordinates without introducing squares. In coordinates (y_0, y_1) it yields

$$(y_0, y_1)^\top Q_2(y_0, y_1) = 2y_0y_1,$$

which is the defining hyperbolic form: it vanishes whenever either $y_0 = 0$ or $y_1 = 0$. This property matches perfectly with our two invariant subspaces in the compressed coordinates $y = Px = (y_0, y_1)^\top$: points in Layer 0 map to $y_0 = 0$, while points in the Fixed space map to $y_1 = 0$, as will be shown in Lemma 3.3.3. Thus Q_2 is the natural building block for our separator.

Figure 3.2 shows the level set $2y_0y_1 = 1$ in the compressed coordinates $y = Px = (y_0, y_1)^\top$.

To work in $\mathbb{R}^n$, we embed Q_2 as the upper-left block of an $n \times n$ matrix Q and pad the rest with zeros:

$$Q = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}.$$

This ensures that Q only couples the first two coordinates of the transformed system and ignores the remaining $n - 2$.

Next, we build an orthonormal basis aligned with the two invariant subspaces

$$V_1 := \text{span}\{e\} \quad (\text{Fixed space}),$$

and

$$V_0 := L_0 := \ker(e^\top) = \{x \in \mathbb{R}^n : e^\top x = 0\} \quad (\text{Layer 0}).$$

Let

$$\hat{e} := \frac{e}{\|e\|_2}.$$

Given a candidate point p , project it onto $V_1 = \text{span}\{e\}$ and subtract that component:

$$\text{proj}_{V_1}(p) := \frac{e^\top p}{e^\top e} e, \quad f := p - \text{proj}_{V_1}(p) = p - \frac{e^\top p}{e^\top e} e. \quad (3.10)$$

Then $f \in V_0 = L_0$ since

$$e^\top f = e^\top p - \frac{e^\top p}{e^\top e} e^\top e = 0. \quad (3.11)$$

Example. [Projection onto the Fixed space and Layer 0] To illustrate the projection

construction, consider the case $n = 3$ and let

$$e = (1, 1, 1)^\top, \quad \text{Fix} = \text{span}\{e\}, \quad L_0 = \{x \in \mathbb{R}^3 : e^\top x = 0\}.$$

Take the point

$$p = (2, 1, 0)^\top.$$

The orthogonal projection of p onto the Fixed space is

$$\text{proj}_{\text{Fix}}(p) = \frac{e^\top p}{e^\top e} e = \frac{3}{3}(1, 1, 1)^\top = (1, 1, 1)^\top.$$

The corresponding residual vector is

$$f = p - \text{proj}_{\text{Fix}}(p) = (2, 1, 0)^\top - (1, 1, 1)^\top = (1, 0, -1)^\top.$$

Since

$$e^\top f = 1 + 0 - 1 = 0,$$

the vector f lies in Layer 0, illustrating the decomposition of a point into its Fixed-space component and its Layer 0 component. This is precisely the projection construction used throughout the separator framework developed in this chapter.

The preceding example illustrates the projection construction in a simple three-dimensional setting. We now continue with the general construction.

Assuming $p \notin V_1$, we have $f \neq 0$, so we may normalize

$$\hat{f} := \frac{f}{\|f\|_2}. \tag{3.12}$$

Finally, we complete $\{\hat{e}, \hat{f}\}$ to an orthonormal basis of $\mathbb{R}^n$ via Gram–Schmidt process

(see Theorem 2.1.1), producing $g_1, \dots, g_{n-2}$, and define

$$B = [\hat{e}, \hat{f}, g_1, \dots, g_{n-2}]. \quad (3.13)$$

By construction, $B^\top B = I_n$, so B is orthonormal and therefore $B^{-1} = B^\top$. This is convenient because the coordinate transformation and the compressed map can then be written using a transpose instead of a separate matrix inverse, which simplifies both the formulas and the computations.

Using the change of coordinates $y = B^\top x$, we express the quadratic form $y^\top Q y$ in the original x -coordinates by defining

$$Q' := B Q B^\top. \quad (3.14)$$

The quadratic separator is then

$$S(x) = x^\top Q' x, \quad x \in \mathbb{R}^n. \quad (3.15)$$

Finally, to normalize relative to the candidate point p , we assume

$$p \notin \text{Fix} \cup L_0. \quad (3.16)$$

As will be shown in Lemma 3.3.2, this condition implies

$$S(p) \neq 0.$$

We therefore define

$$\hat{S}(x) = \frac{S(x)}{S(p)}. \quad (3.17)$$

In the two-subspace case, this is equivalent to requiring $y(p) = Pp = (y_0(p), y_1(p))$

to satisfy $y_0(p) \neq 0$ and $y_1(p) \neq 0$, hence $S(p) = 2y_0(p)y_1(p) \neq 0$. By construction, and as will be shown in Theorem 3.3.1, the normalized separator satisfies

$$\hat{S}(p) = 1 \quad \text{and} \quad \hat{S}(x) = 0 \quad \text{for } x \in \text{Fix} \cup L_0.$$

3.3.1 Transformation to (y_0, y_1) -coordinates

In this subsection we connect the earlier compression map $y = Px$ (with $P = MB^{-1}$) to the orthonormal basis B , and show that the separator depends only on the first two compressed coordinates (y_0, y_1) in the two-subspace case.

Define $y = B^{-1}x \in \mathbb{R}^n$. Since B is an orthonormal basis, $B^{-1} = B^\top$, and therefore $y = B^\top x$. The vector y gives the coordinates of x in the B -basis $(\hat{e}, \hat{f}, g_1, \dots, g_{n-2})$. As in the compressed-coordinate framework introduced earlier (with $P = MB^{-1}$), we will keep only the first two B -coordinates using the selector matrix M .

$$\begin{aligned} S(x) &= x^\top Q' x = x^\top B Q B^\top x \\ &= (B^\top x)^\top Q (B^\top x) = y^\top Q y. \end{aligned} \tag{3.18}$$

Since Q only couples the first two coordinates of $y = (y_0, y_1, \dots)^\top$,

$$S(x) = y^\top Q y = 2y_0 y_1. \tag{3.19}$$

Thus the separator depends only on the two scalar coordinates

$$y_0 = \hat{e}^\top x, \quad y_1 = \hat{f}^\top x. \tag{3.20}$$

The remaining coordinates associated with $g_1, \dots, g_{n-2}$ do not appear in the separator and therefore do not affect its value.

Matrix form of extraction. To make the compression explicit, define

$$M = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{2 \times n}, \quad Y(x) = MB^\top x = \begin{bmatrix} y_0 \\ y_1 \end{bmatrix}.$$

Since $B^{-1} = B^\top$, this agrees with the earlier definition $P = MB^{-1}$, i.e., $Y(x) = MB^\top x = Px$ in the two-subspace case. Equation (3.19) can now be written compactly as

$$S(x) = Y(x)^\top Q_2 Y(x).$$

3.3.2 Properties of the separator

Lemma 3.3.1 (Vanishing on invariant subspaces). *If $x \in \text{Fix} \cup L_0$, then $S(x) = 0$.*

Proof. Recall that in the two-subspace setting

$$\text{Fix} = \text{span}\{e\}, \quad L_0 = \ker(e^\top) = \{x \in \mathbb{R}^n : e^\top x = 0\}.$$

Let $\hat{e} := e/\|e\|_2$. Since $\hat{e}$ is a nonzero scalar multiple of e , it spans the same one-dimensional space, hence $\hat{e} \in \text{Fix}$.

Recall from (3.10) and (3.12) that

$$f := p - \frac{e^\top p}{e^\top e} e, \quad \hat{f} := \frac{f}{\|f\|_2}.$$

By (3.11), we have $e^\top f = 0$, so $f \in L_0$, and scaling does not change orthogonality, therefore also $\hat{f} \in L_0$. In particular,

$$\hat{e}^\top \hat{f} = \left(\frac{e}{\|e\|_2} \right)^\top \hat{f} = \frac{e^\top \hat{f}}{\|e\|_2} = 0.$$

Let B be the orthonormal basis from (3.13), and define $y := B^\top x$. Then by (3.19) and (3.20),

$$S(x) = 2y_0y_1, \quad y_0 = \hat{e}^\top x, \quad y_1 = \hat{f}^\top x.$$

(1) Case $x \in \text{Fix}$. Then $x = \alpha e$ for some $\alpha \in \mathbb{R}$. Since $\hat{f} \in L_0 = \ker(e^\top)$, we have $\hat{f}^\top e = 0$. Hence

$$y_1 = \hat{f}^\top x = \hat{f}^\top(\alpha e) = \alpha \hat{f}^\top e = 0.$$

Therefore $S(x) = 2y_0y_1 = 0$.

(2) Case $x \in L_0$. Then $e^\top x = 0$, so

$$y_0 = \hat{e}^\top x = \left(\frac{e}{\|e\|_2} \right)^\top x = \frac{e^\top x}{\|e\|_2} = 0.$$

Therefore $S(x) = 2y_0y_1 = 0$.

In both cases at least one of the coordinates y_0 or y_1 is zero, and since $S(x) = 2y_0y_1$, it follows that $S(x) = 0$ for all $x \in \text{Fix} \cup L_0$. $\square$

Lemma 3.3.2 (Non-vanishing at p). *If $p \notin \text{Fix} \cup L_0$, then*

$$S(p) = 2y_0(p)y_1(p) \neq 0.$$

Proof. Let $B = [\hat{e}, \hat{f}, g_1, \dots, g_{n-2}]$ be the orthonormal basis, and set $y = B^\top p$.

- Since $p \notin L_0 = \ker(e^\top)$, we have $e^\top p \neq 0$, hence

$$y_0(p) = \hat{e}^\top p \neq 0.$$

- Since $\mathbb{R}^n = \text{span}\{e\} \oplus \ker(e^\top)$ as an orthogonal direct sum, every vector p

decomposes uniquely as

$$p = \frac{e^\top p}{e^\top e} e + \left(p - \frac{e^\top p}{e^\top e} e \right),$$

where the second term lies in $\ker(e^\top) = L_0$. Because $p \notin \text{Fix} = \text{span}\{e\}$, this L_0 -component is nonzero. By definition, $\hat{f}$ is the normalization of this L_0 -component, so $\hat{f}^\top p \neq 0$. Hence

$$y_1(p) = \hat{f}^\top p \neq 0.$$

Therefore both $y_0(p)$ and $y_1(p)$ are nonzero, and thus

$$S(p) = 2y_0(p)y_1(p) \neq 0.$$

□

Theorem 3.3.1 (Normalized separator as a quadratic constraint). *For $p \notin \text{Fix} \cup L_0$, the normalized separator*

$$\hat{S}(x) = \frac{S(x)}{S(p)}$$

satisfies

$$\hat{S}(p) = 1, \quad \hat{S}(x) = 0 \text{ for } x \in \text{Fix} \cup L_0.$$

In (y_0, y_1) -coordinates, the level set $\{x : \hat{S}(x) = 1\}$ corresponds to the hyperbola

$$2y_0y_1 = 1.$$

Hence the inequality $\hat{S}(x) < 1$ contains $\text{Fix} \cup L_0$, while p lies on the boundary

$$\{x : \hat{S}(x) = 1\}.$$

Proof. By Lemma 3.3.1, $S(x) = 0$ whenever $x \in \text{Fix} \cup L_0$, so $\hat{S}(x) = S(x)/S(p) = 0$ on those subspaces. By Lemma 3.3.2, $S(p) \neq 0$ for $p \notin \text{Fix} \cup L_0$, so $\hat{S}(p) = S(p)/S(p) = 1$. In coordinates $y = B^\top x$, we have $S(x) = 2y_0y_1$ from (3.19), hence $\hat{S}(x) = 1$ corresponds to $2y_0y_1 = 1$, which is a hyperbola in the (y_0, y_1) -plane. Since $\hat{S}(x) = 0$ for every $x \in \text{Fix} \cup L_0$, all points in the invariant subspaces satisfy the inequality $\hat{S}(x) < 1$. Moreover, $\hat{S}(p) = 1$, so p lies on the boundary $\{x : \hat{S}(x) = 1\}$. $\square$

Lemma 3.3.3 (Invariant subspaces as coordinate axes). *In the transformed coordinates $y = B^\top x$, the Layer 0 subspace coincides with the axis $y_0 = 0$, and the Fixed space coincides with the axis $y_1 = 0$.*

Proof. Write

$$B = [\hat{e}, \hat{f}, g_1, \dots, g_{n-2}],$$

so that

$$y_0 = \hat{e}^\top x, \quad y_1 = \hat{f}^\top x.$$

First consider Layer 0.

If $x \in L_0 = \ker(e^\top)$, then $e^\top x = 0$, hence

$$y_0 = \hat{e}^\top x = \frac{e^\top x}{\|e\|_2} = 0.$$

Conversely, if $y_0 = 0$, then by (3.20), $\hat{e}^\top x = 0$. Since $\hat{e} = e/\|e\|_2$, it follows that $e^\top x = 0$, and therefore $x \in L_0$.

Hence

$$L_0 = \{x \in \mathbb{R}^n : y_0 = 0\}.$$

Next consider the Fixed space.

If $x \in \text{Fix} = \text{span}\{e\}$, then $x = \alpha e$ for some α . Since $\hat{f} \in L_0 = \ker(e^\top)$, we have $\hat{f}^\top e = 0$, and therefore

$$y_1 = \hat{f}^\top x = \alpha \hat{f}^\top e = 0.$$

Conversely, if $y_1 = 0$, then by (3.20), $\hat{f}^\top x = 0$. In the compressed (y_0, y_1) coordinates used by the separator, only the first two coordinates are retained. Therefore, the condition $y_1 = 0$ corresponds to the y_0 -axis in the compressed coordinate plane. Since vectors in the Fixed space satisfy $y_1 = 0$, the image of the Fixed space in the compressed coordinates is represented by the axis $y_1 = 0$.

Therefore, in the transformed coordinates, Layer 0 coincides with the axis $y_0 = 0$ and the Fixed space coincides with the axis $y_1 = 0$. $\square$

Remark. [Geometric picture] In the (y_0, y_1) -plane, the level set $\{x : \hat{S}(x) = 1\}$ is the hyperbola $2y_0y_1 = 1$, as illustrated in Figure 3.2. The coordinate axes correspond to the invariant subspaces: $y_0 = 0$ represents Layer 0, and $y_1 = 0$ represents the Fixed space. Since $\hat{S}(x) = 0$ on these subspaces, they lie in the region $\hat{S}(x) < 1$. From the solver point of view, the hyperbola is the boundary between the kept region $\hat{S}(x) < 1$ and the cut-off region $\hat{S}(x) \geq 1$.

3.3.3 Visualization in transformed coordinates

In practice, we visualize the separator by plotting contours of $\hat{S}(x)$ in the (y_0, y_1) -plane. For these experiments, we use **Racket** [26] because its built-in **plot** module provides a convenient way to generate the contour visualizations in the (y_0, y_1) -plane. Recall that the compressed coordinates are $y = MB^\top x = (y_0, y_1)$ and $S(x) = 2y_0y_1$, so the level sets of $\hat{S}(x) = S(x)/S(p)$ are hyperbolas in (y_0, y_1) .

Small- n case study ($n = 3$). For the base case $n = 3$, the contour plot illustrates the essential geometry of the quadratic separator. The normalized form satisfies $\hat{S}(p) = 1$, so the point p lies exactly on the red contour. The invariant subspaces appear as the coordinate axes $y_0 = 0$ (Layer 0) and $y_1 = 0$ (Fixed space), on which the function $\hat{S}(x)$ is zero. As discussed in Subsection 3.1.4, these lines correspond to axis-orthogonal hyperplanes in the compressed coordinates; in this two-dimensional visualization they coincide with coordinate axes. The inner contour $\hat{S}(x) = 0.5$ demonstrates how nearby points are classified relative to p , lying strictly between p and the invariant subspaces.

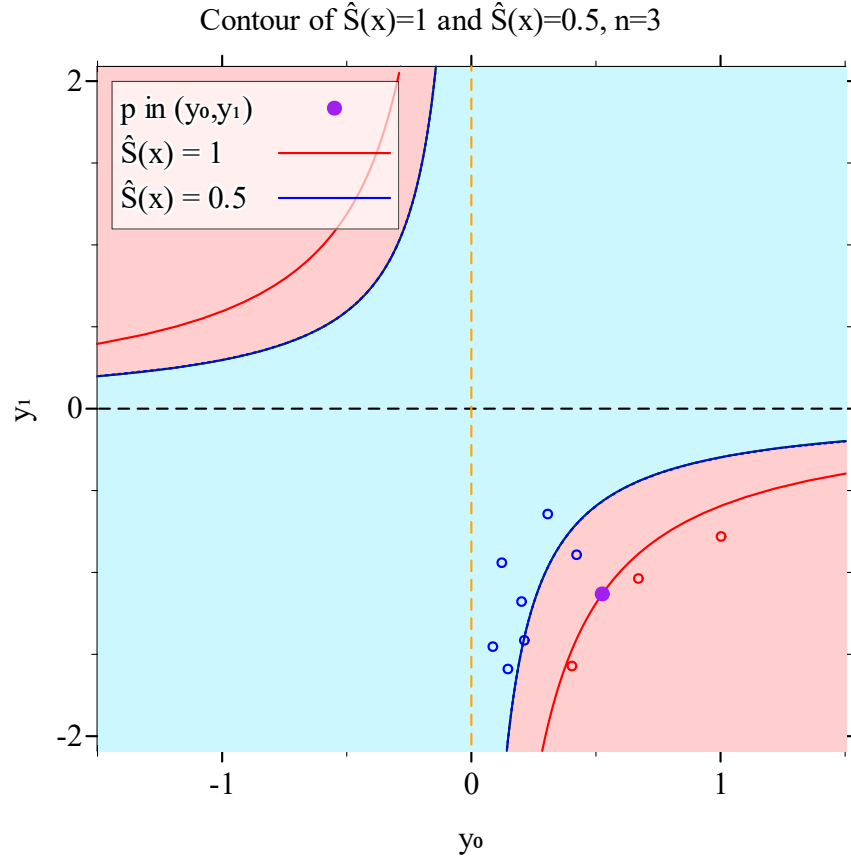


Figure 3.3: Contours of $\hat{S}(x) = 1$ and $\hat{S}(x) = 0.5$ in the (y_0, y_1) -plane for $n = 3$ (generated in `Racket`). The point p is plotted in compressed coordinates $y = MB^\top p = (y_0, y_1)$; the axes $y_0 = 0$ and $y_1 = 0$ correspond to Layer 0 and Fixed space.

Large- n case study ($n = 208$). To test scalability, we extended the construction to dimension $n = 208$. After building the orthonormal basis $B = [\hat{e}, \hat{f}, g_1, \dots, g_{206}]$ via the Gram–Schmidt process and extracting the compressed coordinates $y(p) = MB^\top p = (y_0, y_1)$ of the candidate point p , we evaluated and plotted the normalized separator $\hat{S}(x)$ in the (y_0, y_1) -plane. The results confirm the theoretical properties: the point p lies exactly on the contour $\hat{S}(x) = 1$, while $\hat{S}(x) = 0$ along the coordinate axes $y_0 = 0$ (Layer 0) and $y_1 = 0$ (Fixed space). Despite the high dimensionality, the contour plot retains the same hyperbolic structure observed in low dimensions, demonstrating that the quadratic separator scales cleanly without loss of interpretability.

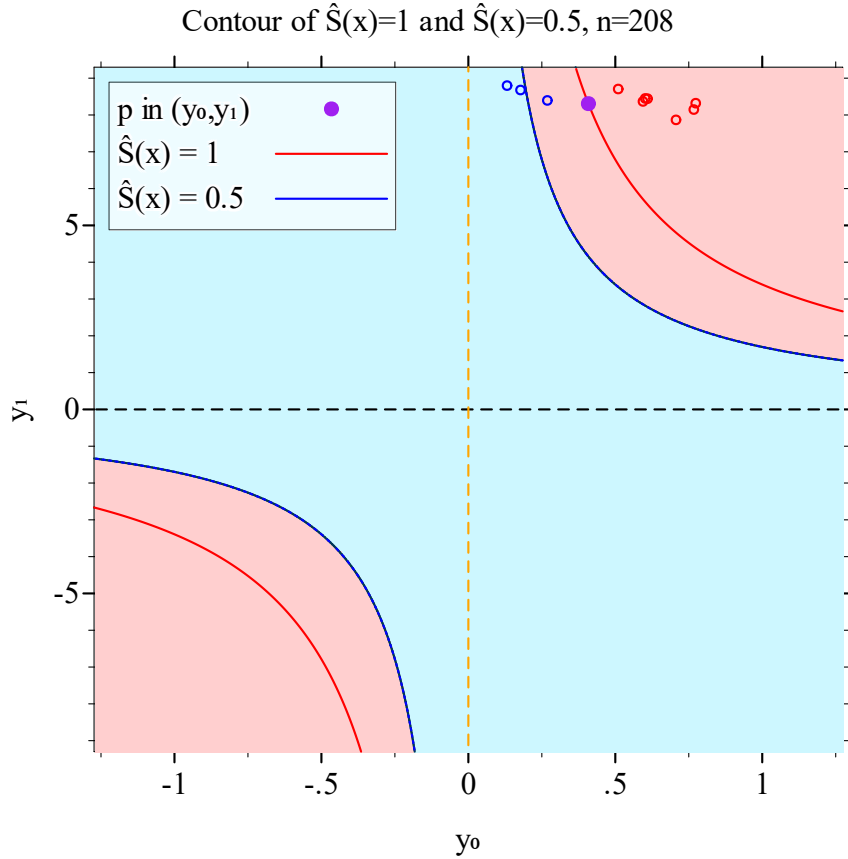


Figure 3.4: Contours of $\hat{S}(x) = 1$ and $\hat{S}(x) = 0.5$ in the (y_0, y_1) -plane for $n = 208$ (generated in **Racket**). The point p is plotted in compressed coordinates $y = MB^\top p = (y_0, y_1)$; the axes $y_0 = 0$ and $y_1 = 0$ correspond to Layer 0 and Fixed space.

Offset sampling and the meaning of the contour $\hat{S} = 0.5$

The plots in Figures 3.3 and 3.4 are generated by sampling nearby points around p in the (y_0, y_1) -plane and evaluating $\hat{S}$ to show which side of the separator they lie on.

Perturbations in transformed coordinates. To test the robustness of the separator, we analyze how the normalized form $\hat{S}(x) = \frac{S(x)}{S(p)}$ behaves under small perturbations of the reference point p . Recall that in the compressed coordinates $y = MB^\top x = (y_0, y_1)$, the separator has the simple expression

$$S(x) = 2 y_0 y_1.$$

We generate nearby trial points by perturbing only these two coordinates:

$$\tilde{y} = (y_0(p) + \Delta y_0, y_1(p) + \Delta y_1, 0, \dots, 0), \quad (\Delta y_0, \Delta y_1) \in [-\varepsilon, \varepsilon]^2,$$

with a small perturbation radius $\varepsilon > 0$. Each perturbed vector $\tilde{y}$ is mapped back to the ambient space by

$$\tilde{x} = B\tilde{y},$$

and then evaluated in the normalized form $\hat{S}(\tilde{x})$.

This procedure amounts to exploring a small square neighborhood around $y(p)$ in the (y_0, y_1) -plane, while fixing all higher coordinates to zero. Geometrically, points with $\hat{S}(\tilde{x}) > 1$ lie on the same side of the separator as p and are shown in red, while points with $\hat{S}(\tilde{x}) < 1$ lie on the opposite side and are shown in blue. The reference point p , for which $\hat{S}(p) = 1$, is shown in purple. The level curve $\hat{S}(x) = 0.5$, for example, gives a contour strictly separating p from the invariant subspaces $\text{Fix} \cup L_0$.

Closed form for the classification. Since $S(x) = 2y_0y_1$, the normalized form at a perturbed point is

$$\hat{S}(\tilde{x}) = \frac{S(\tilde{x})}{S(p)} = \frac{2\tilde{y}_0\tilde{y}_1}{2y_0(p)y_1(p)} = \frac{\tilde{y}_0\tilde{y}_1}{y_0(p)y_1(p)}.$$

Thus the sign and relative size of $\tilde{y}_0\tilde{y}_1$ with respect to $y_0(p)y_1(p)$ determine the classification: $\hat{S}(\tilde{x}) > 1$ (same side as p), $\hat{S}(\tilde{x}) < 1$ (opposite side), and $\hat{S}(\tilde{x}) = 1$ (on the separator).

The significance of the contour $\hat{S} = 0.5$. Level sets of $\hat{S}$ in (y_0, y_1) are hyperbolas

$$\hat{S}(\tilde{x}) = c \iff 2\tilde{y}_0\tilde{y}_1 = cS(p).$$

The branch $\hat{S} = 1$ passes through p because $2y_0(p)y_1(p) = S(p)$. Any c with $0 < c < 1$ defines a hyperbola *strictly closer* to the coordinate axes $y_0 = 0$ and $y_1 = 0$ (the Fixed and Layer 0 subspaces, on which $S(x) = 0$). Thus choosing $c = 0.5$ produces a contour that lies between the separator through p and the subspaces. This intuition can be stated as the following lemma:

Lemma 3.3.4 (Intermediate level sets lie between p and the subspaces). *Let $p \notin \text{Fix} \cup L_0$ so that $y_0(p) \neq 0$ and $y_1(p) \neq 0$. For any c with $0 < c < 1$, the level set $\{x : \hat{S}(x) = c\}$ lies strictly closer to the axes $y_0 = 0$ or $y_1 = 0$ than the level set $\{x : \hat{S}(x) = 1\}$. Consequently, every continuous path from p to either subspace must cross $\{x : \hat{S}(x) = c\}$.*

Proof. In (y_0, y_1) -coordinates, the normalized separator is

$$\hat{S}(x) = \frac{2y_0y_1}{S(p)}.$$

Hence the level set $\hat{S}(x) = c$ is given by

$$2y_0y_1 = cS(p).$$

The level set $\{x : \hat{S}(x) = 1\}$ passes through p , since $\hat{S}(p) = 1$. Now let $0 < c < 1$. Then $cS(p)$ has the same sign as $S(p)$, so the level set $\hat{S}(x) = c$ lies on the same branch family as $\hat{S}(x) = 1$. Also, because $|cS(p)| < |S(p)|$, the corresponding hyperbola lies strictly closer to the coordinate axes $y_0 = 0$ and $y_1 = 0$, which are exactly the sets where $\hat{S}(x) = 0$.

Thus the curve $\hat{S}(x) = c$ lies strictly between the axes and the curve $\hat{S}(x) = 1$. Therefore, any continuous path from the point p , which lies on $\{x : \hat{S}(x) = 1\}$, to either invariant subspace, on which $\hat{S}(x) = 0$, must cross the intermediate level set $\{x : \hat{S}(x) = c\}$. $\square$

3.3.4 Numerical Experiments

We now validate the quadratic separator empirically across a range of dimensions. The goal is to confirm that the theoretical properties established earlier are reflected in numerical practice: vanishing on the invariant subspaces, nonvanishing at the chosen candidate p , and a stable hyperbolic geometry in the transformed coordinates.

Choice of the reference point p

In the two-subspace setting studied in this section, the point p is chosen as a reference point outside $\text{Fix} \cup L_0$ in order to study the geometry of the quadratic separator. Choosing such a point allows us to normalize the separator at p , verify that it vanishes on the invariant subspaces, and illustrate how it distinguishes points that

do not belong to either subspace.

A crucial requirement is that p must lie outside the invariant subspaces:

$$p \notin \text{Fix} \cup L_0.$$

Equivalently, in transformed coordinates $y = B^\top p$, we require both $y_0(p)$ and $y_1(p)$ to be nonzero. If either vanishes, then p lies in the Fixed space ($y_1(p) = 0$) or in Layer 0 ($y_0(p) = 0$), and hence $S(p) = 0$.

To generate such a point in practice, we sampled each coordinate of p independently and uniformly from the interval $[-1, 1]$ and rejected degenerate samples. The sampling routine was implemented in `Racket` by repeatedly generating random vectors in $[-1, 1]^n$ and rejecting any sample that lay in Fix or L_0 . In the implementation, these checks were performed using exact equality tests.

- If all coordinates were equal ($p = \alpha e$), then $p \in \text{Fix}$, so the sample was discarded.
- If the coordinate sum vanished ($e^\top p = 0$), then $p \in L_0$, so the sample was discarded.

This procedure was repeated until a valid point p was obtained. By construction, the resulting point satisfies

$$S(p) = 2y_0(p)y_1(p) \neq 0,$$

see Lemma 3.3.2, which ensures nontrivial separation.

Experimental protocol

For each experiment, a candidate point $p \in \mathbb{R}^n$ was generated with $p \notin \text{Fix} \cup \text{L}_0$, ensuring $S(p) \neq 0$. The procedure was as follows:

1. Construct the orthonormal basis $B = [\hat{e}, \hat{f}, g_1, \dots, g_{n-2}]$ and verify $B^\top B \approx I$.
2. Form Q' as in (3.14), compute $S(p)$, and confirm that $S(e) \approx S(f) \approx 0$.
3. Normalize to obtain $\hat{S}(x) = S(x)/S(p)$, so that $\hat{S}(p) = 1$ by construction.
4. Plot contours of $\hat{S}(x)$ in the (y_0, y_1) -plane, marking p and the invariant axes $y_0 = 0$ (Layer 0) and $y_1 = 0$ (Fixed space).

Numerical results

Table 3.1 summarizes the outcomes for $n = 3, 4, 5, 207, 208, 209$. Across all dimensions, three consistent features were observed:

- $S(p)$ is nonzero, verifying separation of the candidate p .
- $S(e)$ and $S(f)$ vanish to numerical precision (10^{-14} or better), confirming invariance.

Table 3.1: Separator evaluations across tested dimensions. Entries for $S(e)$ and $S(f)$ vanish up to numerical precision, while $S(p) \neq 0$ in all cases. Therefore, the normalized separator satisfies $\hat{S}(p) = 1$ by construction.

Dim	$S(e)$	$S(f)$	$S(p)$	p in (y_0, y_1)
3	≈ 0	≈ 0	0.275	$(-0.130, -1.059)$
4	≈ 0	≈ 0	-0.888	$(-0.809, 0.549)$
5	≈ 0	≈ 0	-1.202	$(-0.600, 1.001)$
207	≈ 0	≈ 0	2.250	$(0.139, 8.100)$
208	≈ 0	≈ 0	-2.493	$(-0.148, 8.439)$
209	≈ 0	≈ 0	-13.735	$(-0.803, 8.555)$

The values in Table 3.1 show that $S(p)$ is not fixed across dimensions: it may be positive, negative, less than 1, or greater than 1. Therefore, the unnormalized inequality $S(x) \leq 1$ is not uniformly adapted to an arbitrary chosen point p , since p may or may not violate that fixed threshold. For this reason, the normalized separator $\hat{S}(x) = S(x)/S(p)$ is used: it preserves the same zero set while rescaling the quadratic form so that $\hat{S}(p) = 1$ by construction.

These results confirm that the quadratic separator behaves exactly as predicted: it vanishes on the invariant subspaces, normalizes to 1 at p , and preserves hyperbolic level sets across small and large dimensions alike. Importantly, scaling to $n = 208$ and $n = 209$ did not introduce numerical instability. The separator therefore extends smoothly from low-dimensional, easily visualized cases to high-dimensional settings that arise in practical instances.

3.4 Generalizing to Multiple Subspaces

The quadratic separator of Section 3.3 treats the case $k = 2$, corresponding to the Fixed space and Layer 0. We now extend the same geometric idea to the family of

invariant subspaces

$$V_j := \ker \Delta(j, n), \quad j = 0, \dots, k-1,$$

associated with the Fourier-based $\Delta(h, n)$ construction introduced in Chapter 2 and recalled earlier in this chapter.

Our goal is to construct a single separator that preserves the union

$$\bigcup_{j=0}^{k-1} V_j$$

while excluding the reference point p .

The strategy is the same as in the two-subspace case: we build a basis adapted to the chosen subspaces, compress the problem to a lower-dimensional coordinate system, and then design the separator in those compressed coordinates.

3.4.1 Multi-Subspace Setup and Compressed Coordinates

For each selected index $j = 0, \dots, k-1$, we work with the invariant kernel subspace

$$V_j = \ker \Delta(j, n).$$

These are the subspaces to be preserved by the separator. These kernel spaces should be distinguished from the Fourier mode spaces $V_j^\perp = \text{span}\{V(j, n), U(j, n)\}$ introduced earlier. For each j , let $P(j, n)$ denote the orthogonal projector onto V_j , and define

$$f_j := p - P(j, n)p,$$

which measures the component of the reference point p orthogonal to the subspace V_j . Stacking these vectors gives the matrix

$$F = [f_0 \ f_1 \ \cdots \ f_{k-1}].$$

We complete these columns to a basis of $\mathbb{R}^n$ by writing

$$B = [F \ G],$$

where the columns of G form an orthonormal basis of the nullspace of $F^\top$, thereby completing F to an invertible matrix B .

As in the two-subspace case, we retain only the first k coordinates in the B -representation. With

$$M = [I_k \ 0], \quad P := MB^{-1},$$

the compressed coordinates of a point $x \in \mathbb{R}^n$ are

$$y = Px \in \mathbb{R}^k.$$

This compressed map has the same geometric meaning as in Subsection 3.1.4: membership in a subspace V_j forces the j -th compressed coordinate to vanish. That is,

$$x \in V_j \implies y_j = 0.$$

Hence, for each $j = 0, \dots, k-1$, under the compressed map $x \mapsto y = Px$, every point of the invariant subspace V_j is sent to the coordinate hyperplane

$$\{y \in \mathbb{R}^k : y_j = 0\}.$$

Equivalently,

$$P(V_j) \subseteq \{y \in \mathbb{R}^k : y_j = 0\}.$$

Therefore, the union of invariant subspaces maps into the union of these coordinate hyperplanes:

$$P\left(\bigcup_{j=0}^{k-1} V_j\right) \subseteq \bigcup_{j=0}^{k-1} \{y \in \mathbb{R}^k : y_j = 0\}.$$

In the special case $k = 2$, these hyperplanes reduce to the two coordinate axes in the (y_0, y_1) -plane, recovering the earlier picture for Layer 0 and the Fixed space:

$$\text{Fix} \mapsto \{y_1 = 0\}, \quad L_0 \mapsto \{y_0 = 0\}.$$

In the general case, however, they are coordinate hyperplanes in $\mathbb{R}^k$, and this is the geometric setting in which the cone separator is constructed below.

3.4.2 Derivation of the Cone Constraint in k Dimensions

Before introducing the algebraic cone form used later in this chapter, we recall the geometric motivation that arises when separating a point from the union of several invariant subspaces. This derivation follows the symmetry-based argument used for circulant structures and explains why the cone constraint has the particular shape that will be employed in Subsection 3.4.3.

As noted in the multi-subspace construction, we may check separately whether the origin is feasible or optimal. Therefore it is permissible to design separators whose level set passes through the origin. Since the compressed map $x \mapsto y = Px$ (see Subsection 3.4.1) is linear, it preserves the origin: $x = 0$ maps to $y = 0$. Thus it is natural, in compressed coordinates, to consider a conic region with vertex at the origin.

Orthant geometry and the cone formulation

In y -space the positive orthant

$$\mathbb{R}_{\geq 0}^k = \{y \in \mathbb{R}^k : y_i \geq 0 \text{ for all } i\}$$

is a simplicial cone generated by the coordinate directions $e_1, \dots, e_k$. To motivate the sharp rotationally symmetric cone, observe that in the Layer 1 slice the positive orthant becomes a regular simplex. The inscribed sphere of this simplex touches each facet at its barycenter, and joining this sphere to the origin yields a natural cone contained in the positive orthant. For the facet obtained by omitting coordinate k , this barycenter is

$$b = \sum_{i=1}^{k-1} e_i.$$

The cosine of the angle between the all-ones direction e and b is

$$\cos(e, b) = \frac{e^\top b}{\|e\| \|b\|} = \frac{k-1}{\sqrt{k} \sqrt{k-1}} = \sqrt{\frac{k-1}{k}}.$$

As k grows, this quantity approaches 1. Geometrically, the angle between the central axis e and the facet barycenters tends to zero, reflecting the shrinking solid angle of each orthant.

In these compressed coordinates, the all-ones direction e is the central direction of the positive orthant. As shown later in Subsection 3.4.3, the basis construction gives $Pp = e$, so this is also the direction of the compressed candidate point.

To construct a separator that cuts off this direction while accepting all axis directions, we require the cone to exclude points y whose angle with e is too small; equivalently,

whose cosine with e is too large. Thus we impose

$$\cos(y, e) \leq \sqrt{\frac{k-1}{k}}.$$

Using

$$\cos(y, e) = \frac{\langle y, e \rangle}{\|y\| \sqrt{k}},$$

we obtain

$$\frac{\langle y, e \rangle}{\|y\| \sqrt{k}} \leq \sqrt{\frac{k-1}{k}} \quad \Longleftrightarrow \quad \langle y, e \rangle \leq \sqrt{k-1} \|y\|.$$

Squaring both sides yields the quadratic relation

$$(k-1) \|y\|^2 \geq \langle y, e \rangle^2. \quad (3.21)$$

Relaxation and the admissible range for c

In practice, enforcing the sharp cone (3.21) is numerically fragile, especially in large dimension. We therefore relax the cone by introducing a scalar parameter $c < 1$:

$$(k-1) \langle y, y \rangle - c \langle y, e \rangle^2 \geq 0. \quad (3.22)$$

However, the relaxation must preserve the essential requirement that the direction $y = e$ remains *infeasible*. Substituting $y = e$ into (3.22) gives

$$(k-1) \langle e, e \rangle - c \langle e, e \rangle^2 < 0$$

which ensures e lies *inside* the cone. Since $\langle e, e \rangle = k$, the inequality becomes

$$(k-1)k < ck^2, \quad \Longleftrightarrow \quad c > \frac{k-1}{k}.$$

Thus the relaxation parameter must satisfy

$$\frac{k-1}{k} < c < 1. \tag{3.23}$$

A convenient family of admissible choices is

$$c = 1 - \frac{1}{k^d}, \quad d > 1.$$

Since

$$\frac{1}{k^d} < \frac{1}{k},$$

every such choice satisfies

$$\frac{k-1}{k} < c < 1.$$

For example, the choice

$$c = 1 - \frac{1}{k^2}$$

lies safely within the admissible range and serves as a convenient representative of this family.

The parameter c governs the width of the relaxed cone. Values of c close to the lower

bound $(k-1)/k$ produce a narrower forbidden cone, cutting off fewer points around the central direction e . In contrast, values of c closer to 1 yield a wider cone that more closely approximates the unrelaxed separator corresponding to $c = 1$.

Within the family $c = 1 - 1/k^d$, larger values of d produce values of c closer to 1, improving the approximation of the unrelaxed cone while potentially increasing numerical sensitivity. The specific choice of d may therefore be guided by implementation and experimental considerations.

3.4.3 The Algebraic Cone Constraint

Inequality (3.22) defines the relaxed cone directly in the compressed coordinates $y = Px$. In these coordinates, each invariant subspace maps into a coordinate hyperplane, and the cone separates the distinguished direction e from the union of these hyperplanes.

The matrix formulation introduced below represents the same cone condition in quadratic form, making it suitable for pullback to the original x -coordinates and for implementation within the optimization model.

Quadratic form and pullback. Define the matrix

$$Q_{\text{cone}}(k, c) := (k-1)I_k - cee^{\top} \in \mathbb{R}^{k \times k}, \quad e = (1, \dots, 1)^{\top}.$$

Then the relaxed cone inequality (3.22) can be written as

$$y^{\top} Q_{\text{cone}}(k, c) y \geq 0.$$

With the compression map $y = Px = MB^{-1}x$ (Subsection 3.1.4), we have

$$\begin{aligned} y^\top Q_{\text{cone}}(k, c)y &= (MB^{-1}x)^\top Q_{\text{cone}}(k, c) (MB^{-1}x) \\ &= x^\top \underbrace{(B^{-\top} M^\top Q_{\text{cone}}(k, c) MB^{-1})}_{=: Q'_{\text{cone}}} x. \end{aligned}$$

Thus the cone appears in the original coordinates as the single quadratic inequality

$$x^\top Q'_{\text{cone}} x \geq 0, \quad Q'_{\text{cone}} = B^{-\top} M^\top Q_{\text{cone}}(k, c) MB^{-1}. \quad (3.24)$$

Why the candidate maps to e and is cut

For $j = 0, \dots, k-1$, let $V_j = \ker \Delta(j, n)$ be the subspace for frequency j , and let $P(j, n)$ denote the projector onto V_j . Define the directions

$$f_j := p - P(j, n)p,$$

which point from the subspace V_j up to p . Stacking these we form the basis

$$B = \begin{bmatrix} f_0 & f_1 & \cdots & f_{k-1} & G \end{bmatrix},$$

where the columns of G complete $\{f_j\}$ to a basis of $\mathbb{R}^n$ (e.g., G can be taken from the nullspace of $F^\top$ with $F = [f_0 \cdots f_{k-1}]$). By construction, we can write p in the B -basis

$$p = \sum_{j=0}^{k-1} f_j + 0 \cdot G,$$

(Lemma 3.1.2), so in the B -coordinates we have

$$y(p) = B^{-1}p = (1, 1, \dots, 1, 0, \dots, 0)^\top, \quad (3.25)$$

and by the selector $M = [I_k \ 0]$, the compression keeps only the first k coordinates via $P = MB^{-1}$, so (3.25) reduces to

$$Pp = MB^{-1}p = e. \quad (3.26)$$

Hence the candidate point p *always* appears as the vector e in y -space. By the admissible range derived in Subsection 3.4.2, namely

$$\frac{k-1}{k} < c < 1,$$

the point $y = e$ violates the relaxed cone inequality

$$y^\top Q_{\text{cone}}(k, c)y \geq 0.$$

Therefore the pulled-back cone constraint excludes the candidate point p , as intended.

The formal verification of these properties is given in Theorem 3.4.1.

Supporting hyperplanes of subspace images

For each j , the hyperplane $\{y \in \mathbb{R}^k : y_j = 0\}$ contains the image of the subspace $V_j = \ker \Delta(j, n)$ under the mapping $x \mapsto y = MB^{-1}x$:

$$MB^{-1}(\ker \Delta(j, n)) \subseteq \{y : y_j = 0\}.$$

When $\dim \ker \Delta(j, n) = k - 1$, this containment may be equality; in general (typical case $\dim \ker \Delta(j, n) = k - 2$) the compressed image lies strictly inside that hyperplane.

Geometrically, $y_j = 0$ is a supporting hyperplane containing

$$MB^{-1}(\ker \Delta(j, n)).$$

Our cone constraint is designed to separate $y = e$ from the union of these subspace images, each of which lies in one of these supporting hyperplanes of the form $y_j = 0$, rather than from the individual coordinate axes, so the construction remains valid even when the V_j are not mutually orthogonal. (In the special two-subspace case, Fixed space and Layer zero, the picture reduces to orthogonal axes.)

With the relaxed inequality $y^\top Q_{\text{cone}}(k, c)y \geq 0$,

$$\mathcal{F} := \{y : y^\top Q_{\text{cone}}(k, c)y \geq 0\} \quad \text{is the } \textit{feasible} \text{ side (outside the cone),}$$

and the strict violation set

$$\mathcal{I} := \{y : y^\top Q_{\text{cone}}(k, c)y < 0\} \quad \text{is the } \textit{infeasible} \text{ interior (containing } e\text{).}$$

At $y = 0$ we have equality $0 = 0$; inclusion of the origin in the overall model's feasible set is checked separately, since the cone is only used to cut points with too small an angle to e , not to certify feasibility of $y = 0$.

3.4.4 Acceptance and Rejection Properties of the Cone

We summarize precisely what the relaxed cone inequality keeps and what it cuts.

Theorem 3.4.1 (Acceptance–Rejection of the Cone Separator). *Let $k \geq 2$, $e = (1, \dots, 1)^\top \in \mathbb{R}^k$, and*

$$Q_{\text{cone}}(k, c) = (k-1)I_k - cee^\top, \quad \text{with } \frac{k-1}{k} < c < 1.$$

Let $P = MB^{-1}$ with $M = [I_k \ 0]$ and B constructed as in Section 3.1.3. Then:

1. (Acceptance of all invariant subspaces.) For every $j \in \{0, \dots, k-1\}$ and every $u \in V_j$, we have

$$(Pu)^\top Q_{\text{cone}}(k, c)(Pu) \geq 0.$$

Equivalently, the inequality $y^\top Q_{\text{cone}}(k, c)y \geq 0$ holds for all y in $PB(V_j)$, so the cone keeps the entire union $\bigcup_j V_j$.

2. (Rejection of the candidate point.) If p is the current candidate and $Pp = e$ (as shown in Subection 3.4.3), then

$$(Pp)^\top Q_{\text{cone}}(k, c)(Pp) = (k-1)k - ck^2 < 0.$$

Hence p violates the cone inequality and is cut by $x^\top Q'_{\text{cone}}x \geq 0$, where $Q'_{\text{cone}} = B^{-\top}M^\top Q_{\text{cone}}(k, c)MB^{-1}$.

Proof. (1) Let $u \in V_j$ and set $y := Pu \in \mathbb{R}^k$. By the multi-subspace hyperplane mapping property (Subsection 3.4.1), membership $u \in V_j$ forces the j -th compressed coordinate to vanish:

$$y_j = 0.$$

Hence

$$e^\top y = \sum_{i=0}^{k-1} y_i = \sum_{i \neq j} y_i,$$

because the j -th term is zero.

We now apply Cauchy–Schwarz inequality (see, e.g., [13]) to the $(k-1)$ numbers $\{y_i : i \neq j\}$:

$$\left(\sum_{i \neq j} y_i\right)^2 \leq \left(\sum_{i \neq j} 1^2\right) \left(\sum_{i \neq j} y_i^2\right) = (k-1) \sum_{i \neq j} y_i^2 \leq (k-1) \|y\|^2.$$

(Equivalently, the square of the sum of $k - 1$ numbers is at most $(k - 1)$ times the sum of their squares.)

Therefore,

$$\begin{aligned} y^\top Q_{\text{cone}}(k, c)y &= (k - 1)\|y\|^2 - c(e^\top y)^2 \\ &\geq (k - 1)\|y\|^2 - c(k - 1)\|y\|^2 \\ &= (k - 1)(1 - c)\|y\|^2 \geq 0. \end{aligned}$$

since $c < 1$, $k \geq 1$ (so $k - 1 \geq 0$), and $\|y\|^2 \geq 0$, it follows that $(Pu)^\top Q_{\text{cone}}(k, c)(Pu) \geq 0$ for every $u \in V_j$.

(2) Since $Pp = e$, it suffices to evaluate the cone quadratic form on the all-ones vector $e \in \mathbb{R}^k$. Recall that

$$Q_{\text{cone}}(k, c) = (k - 1)I_k - cee^\top.$$

Then

$$e^\top Q_{\text{cone}}(k, c)e = e^\top ((k - 1)I_k - cee^\top)e = (k - 1)e^\top I_k e - ce^\top(ee^\top)e.$$

For the first term, $I_k e = e$, hence $e^\top I_k e = e^\top e = k$ because e has k entries equal to

1. For the second term, note that

$$(ee^\top)e = e(e^\top e) = ke,$$

so $e^\top(ee^\top)e = e^\top(ke) = k(e^\top e) = k^2$. Substituting these identities yields

$$e^\top Q_{\text{cone}}(k, c)e = (k - 1)k - ck^2.$$

This value is negative precisely when $c > \frac{k-1}{k}$, which is the admissible range in (3.23). Finally, pulling back by $y = Px$ expresses the same inequality as a cut in the original x -coordinates.

□

Remark. [Feasible vs. infeasible sides] The *feasible side* is $\{y : y^\top Q_{\text{cone}}(k, c)y \geq 0\}$, which contains all $PB(V_j)$; the *infeasible interior* is $\{y : y^\top Q_{\text{cone}}(k, c)y < 0\}$ and contains $e = Pp$.

3.4.5 Two-Subspace Prototype vs. Multi-Subspace Cone

Our `Racket` prototype works with $k = 2$ (Fixed space and Layer 0) and the bilinear form $S(x) = 2y_0y_1$, which vanishes on the union $\{y_0 = 0\} \cup \{y_1 = 0\}$. The multi-subspace generalization replaces this 2×2 block by the relaxed cone matrix

$$Q_{\text{cone}}(k, c) = (k - 1)I_k - cee^\top, \quad c \in \left(\frac{k-1}{k}, 1\right),$$

and tests the single inequality $y^\top Q_{\text{cone}}(k, c)y \geq 0$ in y -space.

SCIP workflow in the next chapter (Chapter 4). For each candidate point exposed during a monitored solve slice, the separation routine:

1. builds $B(p)$ from the invariant subspace analysis (Section 3.1.3);
2. maps the current candidate p to $y = Px = MB^{-1}x$ and evaluates the cone score

$$S(p) := y^\top Q_{\text{cone}}(k, c)y.$$

By Theorem 3.4.1, $S(p) < 0$, so p violates the cone;

3. adds the quadratic cut in original coordinates

$$x^\top Q'_{\text{cone}} x \geq 0, \quad Q'_{\text{cone}} = B^{-\top} M^\top Q_{\text{cone}}(k, c) M B^{-1},$$

which cuts p while keeping the union of invariant subspaces.

Thus, the $k = 2$ **Racket** model corresponds to the two-subspace bilinear form $2y_0y_1$, while the SCIP implementation uses the general cone construction for $k \geq 2$.

This chapter developed the geometric framework underlying the proposed separator. Beginning with the two-subspace setting of the Fixed space and Layer 0, we constructed a quadratic separator and then extended the construction to the general multi-subspace setting associated with cyclic symmetry. This led to a cone-based quadratic inequality in compressed coordinates together with its pullback to the original variables. The next chapter integrates the multi-subspace cone separator into a cut-and-solve framework implemented in SCIP through PySCIPOpt and evaluates its computational behaviour.

Chapter 4

Geometric Cone Separators in a Cut-and-Solve Framework

4.1 Motivation and Overview

Chapter 3 developed the geometric and algebraic framework underlying our separator. In the multi-subspace setting associated with the Fourier decomposition, as shown in Subsection 3.1.4, the compressed coordinates

$$y = MB^{-1}x$$

map each invariant subspace V_j into a coordinate hyperplane of $\mathbb{R}^k$, rather than onto a single axis. That is,

$$MB^{-1}(V_j) \subseteq \{y \in \mathbb{R}^k : y_j = 0\}.$$

Consequently, in our construction the feasible side of the separator is chosen to contain the union of these coordinate hyperplanes

$$\bigcup_{j=0}^{k-1} \{ y \in \mathbb{R}^k : y_j = 0 \},$$

so that points coming from the preserved invariant subspaces are not cut off. At the same time, the compressed version of the current candidate point p is always

$$y(p) = Pp = e = (1, \dots, 1)^\top,$$

as established in Subsection 3.4.3. Thus the separator must *reject* the single direction e , while accepting all coordinate hyperplanes.

The relaxed cone inequality developed in Subsection 3.4.3 serves exactly this purpose:

$$y^\top Q_{\text{cone}}(k, c) y \geq 0, \quad Q_{\text{cone}}(k, c) := (k-1)I_k - cee^\top, \quad (4.1)$$

with parameter c chosen in the admissible range

$$\frac{k-1}{k} < c < 1.$$

Here the feasible side is

$$\{ y \in \mathbb{R}^k : y^\top Q_{\text{cone}}(k, c) y \geq 0 \},$$

while points with

$$y^\top Q_{\text{cone}}(k, c) y < 0$$

lie in the infeasible interior of the cone, which contains the direction e .

By Theorem 3.4.1, this inequality leaves every coordinate hyperplane $\{y : y_j = 0\}$

on the feasible side, and therefore also accepts the image of each invariant subspace V_j , while rejecting the direction $e = Pp$ whenever

$$c \in \left(\frac{k-1}{k}, 1 \right).$$

This characterization ensures that the cone captures the intended separation geometry in the multi-subspace setting.

It is important here to distinguish between the intended mathematical model and the temporary subproblems generated by SCIP during the search. At the level of the intended model, one may think of the problem as

$$Ax \leq b, \quad x \in \mathbb{Z}^n, \quad x \in \bigcup_{j=0}^{k-1} V_j,$$

where the final condition is implicit rather than given to the solver as an explicit family of constraints. The present implementation does not enforce the union-of-subspaces condition through an explicit disjunctive formulation. Instead, it uses the cone-based quadratic relaxation developed in Chapter 3, which provides a single quadratic constraint that accepts the preserved invariant subspaces while rejecting violating candidate directions. When the solver exposes a candidate point, the Python separation routine acts as a black box: the geometric module either returns no cut or produces a cone-based quadratic cut that excludes the violating candidate. Thus, the mathematical model remains fixed, while SCIP generates and solves many internal LP relaxations and branch-and-bound subproblems during the search.

The purpose of the present chapter is to integrate the cone inequality (4.1) into a cut-and-solve framework in SCIP. The current optimization model is optimized only for a short time slice, and, if a candidate point p is available, the geometric model

evaluates the cone score

$$S(p) = y(p)^\top Q_{\text{cone}} y(p).$$

This score indicates whether the compressed point lies on the accepted side of the relaxed cone inequality or strictly inside the forbidden cone around the direction e . In particular, $S(p) < 0$ means that the point violates the cone constraint, whereas $S(p) \geq 0$ means only that the point lies on or outside the cone. Since the cone is a relaxation, the condition $S(p) \geq 0$ does not imply that p lies in, or is necessarily close to, one of the coordinate hyperplanes. The detailed geometric test described in Section 4.2 also checks separately whether the point is already numerically close to one of the invariant subspaces.

Whenever the cone test is violated, the pullback of the cone inequality is added to the original variables:

$$x^\top Q'_{\text{cone}} x \geq 0, \quad Q'_{\text{cone}} = B^{-\top} M^\top Q_{\text{cone}} M B^{-1}.$$

The model is then rebuilt and optimized again for another short time slice. In each outer iteration, the process proceeds in three steps: a current candidate point is obtained from a partial solve, the geometric model evaluates the separation test at that point, and, whenever violating candidates are detected, the corresponding cone-based quadratic cuts are constructed and appended to the problem. This disciplined time-slice structure ensures that every new inequality has a clear geometric meaning. In this way, the global optimizer continues to operate as a black-box search engine, while the geometric model contributes analytically motivated quadratic constraints whenever violating candidates are detected and the corresponding cone cuts are prepared.

The present method is closely connected to the earlier outer-approximation frame-

work of Shahverdi, Banihashemi, and Bremner [3]. In Algorithm 1 of that work, the singular-circulant case is handled through the type $Q2$ subproblems, in which constraints (9) and (10) are imposed explicitly. The instances studied in this chapter arise from closely related reformulations of that same setting. The main difference is methodological: instead of enforcing those nonlinear constraints explicitly in advance, the present implementation monitors candidate points during the SCIP solve and, whenever the cone test is violated, uses a Python separation routine to add the corresponding cone-based quadratic cut. Thus, the current chapter should be read as a geometric cut-and-solve reformulation of that earlier outer-approximation framework.

The remainder of this chapter describes the algorithmic structure of the cone-based cut-and-solve procedure, its implementation within a Python framework using SCIP and PYSCIPOPT, and the experimental protocol used to evaluate its behaviour across several instance classes.

4.2 Cone-Based Cutting Algorithm

This section formalizes the cut-and-solve process used throughout the chapter. The central object is the cone inequality

$$y^T Q_{\text{cone}}(k, c) y \geq 0,$$

introduced in Section 4.1, where $y = MB^{-1}x$ denotes the compressed coordinate representation aligned with the Fourier invariant subspaces. The role of the algorithm is to monitor candidate points exposed during a short solve slice and, whenever the

cone test is violated, to add the corresponding pullback constraint

$$x^\top Q'_{\text{cone}} x \geq 0$$

to the optimization model.

Each outer iteration consists of the following three steps:

1. **Candidate-point exposure.** The current optimization model is solved for a short time slice in order to expose candidate points during the search. If the current model is found to be infeasible after the solve slice, the algorithm terminates.
2. **Geometric evaluation.** For a candidate point p exposed during Step 1, the vectors $f_h = p - P(h, n)p$ are first computed using the constructions of Chapter 3. Since $P(h, n)p$ is the orthogonal projection of p onto the invariant subspace V_h , the norm $\|f_h\|_2$ measures the distance from p to V_h . If one of these distances is below the chosen numerical tolerance, then p is treated as lying in the corresponding invariant subspace, and no cut is generated from that candidate. Otherwise, the matrix B is assembled and the compressed coordinates $y = MB^{-1}p$ are computed. The cone score

$$S(p) = y^\top Q_{\text{cone}}(k, c) y$$

is then evaluated. If $S(p) < 0$, the point violates the cone inequality and a candidate cut is prepared. If $S(p) \geq 0$, no cone cut is generated from that candidate. Since the cone constraint is a relaxation, this nonnegative score indicates only that p is accepted by the relaxed cone test; it does not by itself imply that p lies in, or is necessarily close to, one of the coordinate hyperplanes. In particular, the cone test depends on the position of $y = MB^{-1}p$ relative to

the cone, so a point farther from the origin may still be much farther from the corresponding hyperplane while remaining on the accepted side of the cone.

3. **Cut construction and reintegration.** When violating candidates are identified, the cone inequality is pulled back to the original coordinates through the transformation

$$Q'_{\text{cone}} = B^{-\top} M^{\top} Q_{\text{cone}}(k, c) M B^{-1}.$$

For each violating candidate for which the corresponding cone cut is prepared, the quadratic constraint $x^{\top} Q'_{\text{cone}} x \geq 0$ is appended to the problem. The updated model is then solved again for the next outer time slice.

This disciplined structure adds cone-based quadratic inequalities only when violating candidates are detected by the geometric test. In the current implementation, more than one such cut may be prepared within a single outer time slice if several violating candidates are collected during that slice. As a result, each added inequality still has a clear geometric meaning derived from the multi-subspace theory developed in Chapter 3. The solver itself remains a black box: it continues to explore the feasible region using its internal branching and cutting strategies, while the geometric module contributes analytically motivated quadratic constraints whenever violating candidates are identified.

4.2.1 Compressed coordinates

We now briefly recall the compressed-coordinate construction used by the implementation. The purpose of this subsection is to introduce the matrices and coordinate transformation needed for the cone score computation, while the rank and invertibility of this construction are discussed separately in Subsection 4.2.2 below.

The relevant Δ -subspaces were introduced in Subsection 2.7.1, and their orthogonal projectors were described in Subsection 2.7.3. Using the notation of Chapter 3, we write

$$V_j := \ker \Delta(j, n), \quad P_j := P(j, n), \quad j = 0, \dots, k-1.$$

Given a candidate point $p \in \mathbb{R}^n$, we form the vectors

$$f_j = p - P_j p, \quad j = 0, \dots, k-1,$$

exactly as in Subsection 3.4.1. These vectors are stacked into the matrix

$$F = [f_0 \ f_1 \ \cdots \ f_{k-1}].$$

They are then completed to a basis of $\mathbb{R}^n$ by adjoining columns from the nullspace of $F^\top$:

$$B = [F \ G], \quad G \subseteq \ker(F^\top).$$

The rank and invertibility of this construction are discussed in Subsection 4.2.2 below.

Once B has been formed, we use the same compressed map as in Subsection 3.1.4 and Subsection 3.4.1:

$$y = MB^{-1}x, \quad M = [I_k \ 0].$$

This map retains only the first k coordinates in the B -representation.

As established earlier in the multi-subspace construction (see Subsections 3.1.4 and 3.4.3), each invariant subspace V_j maps into the coordinate hyperplane

$$\{y \in \mathbb{R}^k : y_j = 0\},$$

and the current candidate point is sent to the distinguished direction

$$y(p) = MB^{-1}p = e = (1, \dots, 1)^\top.$$

Thus the compressed coordinates place the preserved invariant subspaces on coordinate hyperplanes and send the current point to the distinguished direction e , yielding exactly the geometric configuration needed for the cone separator.

4.2.2 Full-Rank Properties of the Basis Matrix

The cone separator requires the basis matrix $B = [F \ G]$ to be invertible. This subsection records the structural facts that justify this requirement and explains how the determinant analysis from Subsection 2.6.2 implies the full-rank property of F whenever the current point p does not lie in any invariant subspace.

Recall that

$$f_j = p - P_j p, \quad j = 0, \dots, k-1,$$

where $P_j = P(j, n)$ is the orthogonal projector onto

$$V_j = \ker \Delta(j, n).$$

If $p \notin V_j$ for all j , then each f_j is nonzero. The rank condition

$$\text{rank}(F) = k, \quad F = [f_0 \ \cdots \ f_{k-1}],$$

is therefore the essential algebraic requirement ensuring that F spans a k -dimensional subspace and that the completion $B = [F \ G]$ is invertible for any choice of $G \subseteq \ker(F^\top)$.

A key observation is that the rank of F reduces to the rank of a structured trigonometric matrix $R(n)$ that depends only on the Fourier geometry of the operators $\Delta(j, n)$, and not on the current point p . The matrix $R(n)$, together with its odd- and even-dimensional forms, was introduced and analyzed in Subsection 2.6.2. We therefore do not repeat that construction here. Instead, we use the consequence needed in the present chapter: any linear dependence among the columns of

$$F = [f_0 \ f_1 \ \cdots \ f_{k-1}]$$

would induce a corresponding dependence in the rows of $R(n)$. Hence

$$\text{rank}(R(n)) = n \implies \text{rank}(F) = k.$$

Scott's determinant. The determinant analysis for the odd-dimensional form of $R(n)$ was already developed in Subsection 2.6.2, using Scott's classical trigonometric determinant formula [12].

For odd dimensions $n = 2k + 1$, the relevant angle vector is

$$a = \left(0, \frac{2\pi}{n}, \frac{4\pi}{n}, \dots, \frac{2\pi(n-1)}{n}\right),$$

whose entries are pairwise distinct modulo 2π . As shown in Subsection 2.6.2, the corresponding Scott matrix coincides with $R(n)$ up to the ordering of rows, and therefore

$$\det R(n) \neq 0 \quad \text{for odd } n.$$

Hence $R(n)$ has full rank n in the odd case.

When n is even, certain cosine and sine rows exhibit cancellations. Before introducing the cancellation vector, it is useful to recall the structure of the even-dimensional

case. If we look at e.g. $R(6)$, we can see there is a Scott matrix (with a different a vector) in the top right, augmented with one more row and column.

Scott submatrix placement (example $n = 6$).

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & \frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} & 0 & -\frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ 1 & \frac{1}{2} & -\frac{1}{2} & -1 & -\frac{1}{2} & \frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} & 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ 1 & -\frac{1}{2} & -\frac{1}{2} & 1 & -\frac{1}{2} & -\frac{1}{2} \\ 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}$$

In the case $R(6)$, the top-right block is a Scott matrix formed with a suitable angle vector (different ordering than the odd case), and $R(6)$ is a Scott block by one extra row and column.

In general, when n is even, the trigonometric matrix $R(n)$ contains a Scott matrix in its top-right block, but with a different angle vector than in the odd case.

If we remove the first column and the last row of $R(n)$, the resulting $(n-1) \times (n-1)$ matrix

$$\hat{R}(n)$$

is itself a Scott matrix built from the evenly-spaced angles

$$\left(\frac{2\pi}{n}, \frac{4\pi}{n}, \dots, \frac{2\pi(n-1)}{n} \right)$$

i.e.,

$$\hat{R}(n) \equiv \text{Scott}(a), \quad a := \left(\frac{2\pi}{n}, \frac{4\pi}{n}, \dots, \frac{2\pi(n-1)}{n} \right),$$

Thus the odd-dimensional determinant formula continues to play a role here as well: $\widehat{R}(n)$ inherits the full-rank property of Scott matrices. The difficulty for $n = 2k$ arises from the additional cosine row in $R(n)$, which prevents $R(n)$ from being directly expressible as a single Scott matrix. To relate $\det R(n)$ to $\det \widehat{R}(n)$, we express $R(n)$ in block form and track how one extra row and one extra column modify the determinant.

We will prove that these are resolved by considering the system with the weight vector

$$L = (1, 0, 2, 0, 2, \dots, 0, 2, 1),$$

which encodes the cancellation identity of the Dirichlet kernel (see Subsection 2.6.2).

Decomposing $R(n)$ into block form

$$R(n) = \begin{bmatrix} B & \widehat{R}(n) \\ D & C \end{bmatrix},$$

Experimentally there is a simple relationship between the determinants of the smaller Scott matrix and even dimensional R matrix. The rest of the discussion is devoted to proving the following formula

$$\det R(n) = -n \det (\widehat{R}(n)).$$

One may perform an odd number of column swaps to obtain the related matrix

$$R'(n) = \begin{bmatrix} \widehat{R}(n) & B \\ C & D \end{bmatrix}.$$

The determinants are related by

$$\det R(n) = -\det R'(n). \quad (4.3)$$

Since $\widehat{R}(n)$ is invertible and D is a 1×1 matrix, applying the Schur complement (see Subsection 2.1.5) yields

$$\det R'(n) = \det(\widehat{R}(n)) (D - C \widehat{R}(n)^{-1} B). \quad (4.4)$$

To evaluate the scalar $D - C \widehat{R}(n)^{-1} B$ without explicitly inverting $\widehat{R}(n)$, set

$$x = C \widehat{R}(n)^{-1} B.$$

Then x is the unique vector satisfying

$$x \widehat{R}(n) = C,$$

and equivalently, there is a unique such linearity, the $(k+1)$ -dimensional vector, namely

$$\ell = (-x, 1)$$

which is a left-null vector (linearity) of the stacked matrix

$$\begin{bmatrix} \widehat{R}(n) \\ C \end{bmatrix}.$$

Experimentally, a direct trigonometric argument shows that this linearity is unique

up to scaling and equals

$$\ell = (1, 0, 2, 0, 2, \dots, 0, 2, 1),$$

i.e., weight 1 on the first and last cosine rows and weight 2 on all intermediate cosine rows. Uniqueness follows from dimension considerations and scaling of the last element. To verify that ℓ is indeed a linearity, consider the inner product of ℓ with column m of $\begin{bmatrix} \hat{R}(n) \\ C \end{bmatrix}$:

$$0 = (-1)^m + 2 \sum_{j=1}^{\frac{n}{2}-1} \cos\left(\frac{2\pi jm}{n}\right) + 1 \quad (4.5)$$

Before proceeding with the proof, we tested equation (4.5) for a few values of m and n . Since the matrix $\begin{bmatrix} \hat{R}(n) \\ C \end{bmatrix}$ constitutes the block of $R(n)$ corresponding to the non-zero column indices (as shown in $R(n) = \begin{bmatrix} B & \hat{R}(n) \\ D & C \end{bmatrix}$), we only need to verify this identity for columns indexed by $m > 0$ (i.e., non-zero column indices).

The output matrix, generated by evaluating linearity over the relevant range of indices ℓ (rows) and m (columns), confirms that the result is consistently zero for all tested values of m and n within the defined bounds. This experimental validation supports the theoretical claim that the inner product of the linearity vector ℓ with the columns of the matrix block $\begin{bmatrix} \hat{R}(n) \\ C \end{bmatrix}$ is zero, thereby confirming that ℓ is indeed a linearity.

The proof of this result is obtained by recognizing the summation (the two terms of the right-hand side of (4.5)) as a component of the Dirichlet kernel. The trigono-

metric sum is a Dirichlet kernel, and we will show that the identity

$$1 + 2 \sum_{k=1}^N \cos(k\theta) = \frac{\sin((N + \frac{1}{2})\theta)}{\sin(\frac{1}{2}\theta)}$$

reduces the expression to zero for all columns $m > 0$.

In this theoretical proof, we aim to show that the weighted sum

$$S(m) = (-1)^m + 2 \sum_{j=1}^{\frac{n}{2}-1} \cos\left(\frac{2\pi jm}{n}\right) + 1$$

which represents the inner product of ℓ with column m of $\begin{bmatrix} \widehat{R}(n) \\ C \end{bmatrix}$, vanishes for all relevant column indices $m > 0$.

The weighted sum is given by:

$$\begin{aligned} S(m) &= (-1)^m + 2 \sum_{j=1}^{\frac{n}{2}-1} \cos\left(\frac{2\pi jm}{n}\right) + 1 \\ &= \left(1 + 2 \sum_{j=1}^{\frac{n}{2}-1} \cos\left(\frac{2\pi jm}{n}\right)\right) + (-1)^m \end{aligned}$$

We first simplify the term in parentheses using the Dirichlet kernel identity:

$$1 + 2 \sum_{j=1}^N \cos(j\theta) = \frac{\sin((N + \frac{1}{2})\theta)}{\sin(\frac{\theta}{2})}$$

where we set $N = \frac{n}{2} - 1$ and $\theta = \frac{2\pi m}{n}$. The sum term becomes:

$$\begin{aligned}
1 + 2 \sum_{j=1}^{\frac{n}{2}-1} \cos\left(\frac{2\pi jm}{n}\right) &= \frac{\sin\left(\left(\left(\frac{n}{2} - 1\right) + \frac{1}{2}\right) \frac{2\pi m}{n}\right)}{\sin\left(\frac{1}{2} \cdot \frac{2\pi m}{n}\right)} \\
&= \frac{\sin\left(\left(\frac{n-1}{2}\right) \frac{2\pi m}{n}\right)}{\sin\left(\frac{\pi m}{n}\right)} \\
&= \frac{\sin\left(\pi m - \frac{\pi m}{n}\right)}{\sin\left(\frac{\pi m}{n}\right)}
\end{aligned}$$

Using the sine subtraction identity $\sin(\pi m - \beta) = -\cos(\pi m) \sin(\beta) = -(-1)^m \sin(\beta)$, the numerator simplifies. Substituting $\cos(\pi m) = (-1)^m$:

$$\begin{aligned}
1 + 2 \sum_{j=1}^{\frac{n}{2}-1} \cos\left(\frac{2\pi jm}{n}\right) &= \frac{-\cos(\pi m) \sin\left(\frac{\pi m}{n}\right)}{\sin\left(\frac{\pi m}{n}\right)} \\
&= -\cos(\pi m) \\
&= -(-1)^m
\end{aligned}$$

Note that the denominator $\sin\left(\frac{\pi m}{n}\right)$ is nonzero for all column indices $m = 1, 2, \dots, n-1$. The only column for which the Dirichlet-kernel division is invalid is the direct-current column $m = 0$, where $\sin(0) = 0$. However, due to the structure of the block matrix $R(n)$ (where $m = 0$ is handled by block $\frac{B}{D}$), and the fact that we consider only $m > 0$ columns of $\hat{R}_C^{(n)}$, the division is valid for all relevant columns.

Substituting this simplified result back into the expression for $S(m)$:

$$\begin{aligned}
S(m) &= \left(1 + 2 \sum_{j=1}^{\frac{n}{2}-1} \cos\left(\frac{2\pi jm}{n}\right)\right) + (-1)^m \\
&= -(-1)^m + (-1)^m \\
&= 0
\end{aligned}$$

Thus, for every column index $m > 0$ considered, the weighted sum vanishes. This confirms that ℓ is a linearity, which means that the linearity vector

$$\ell = (1, 0, 2, 0, 2, \dots, 0, 2, 1)$$

(with weights 1 on the first and last cosine rows, weights 2 on all interior cosine rows, and 0 on sine rows) annihilates each column of the matrix $\begin{bmatrix} \widehat{R}(n) \\ C \end{bmatrix}$, i.e.,

$$\ell \cdot \begin{bmatrix} \widehat{R}(n) \\ C \end{bmatrix} = \mathbf{0}. \quad (4.6)$$

This establishes that ℓ annihilates every column of $\begin{bmatrix} \widehat{R}(n) \\ C \end{bmatrix}$, which is the augmented block of $R(n)$ containing all columns except the first.

Equation (4.6) implies that the last row C of the augmented block is a linear combination of the rows of $\widehat{R}(n)$. Specifically, there exists a unique row vector x such that

$$x \widehat{R}(n) = C,$$

and therefore

$$x = C \widehat{R}(n)^{-1}. \quad (4.7)$$

It follows immediately that the row vector

$$\ell := (-x, 1)$$

is a left-null vector (linearity) of the stacked matrix

$$\begin{bmatrix} \widehat{R}(n) \\ C \end{bmatrix},$$

since

$$\ell \begin{bmatrix} \widehat{R}(n) \\ C \end{bmatrix} = (-x)\widehat{R}(n) + 1 \cdot C = -C + C = 0.$$

Thus ℓ encodes the cancellation relation between the rows of $\widehat{R}(n)$ and the additional row C .

While (4.6) shows perfect cancellation on columns arising from $\widehat{R}(n)$, applying the same linearity weights (from ℓ) to the extra augmentation column B yields a constant residual equal to the sum of the weights on the cosine rows. The sum of the weights of ℓ is:

$$1 + 2 \cdot \left(\frac{n}{2} - 1\right) + 1 = 1 + n - 2 + 1 = n. \quad (4.8)$$

This demonstrates that when the cancellation vector is applied to B , the resulting scalar is forced to equal n . In Schur complement form, this translates to the required identity:

$$D - C\widehat{R}(n)^{-1}B = n. \quad (4.9)$$

Substituting (4.9) into the block determinant formula for $R(n)$ then allows for the final determinant calculation:

$$\det R'(n) = \det(\widehat{R}(n))(D - C\widehat{R}(n)^{-1}B) = n \det(\widehat{R}(n)),$$

and

$$\det R(n) = -\det R'(n),$$

yields

$$\det R(n) = -n \det (\widehat{R}(n)).$$

Consequently,

$$\det R(n) \neq 0.$$

Therefore $\text{rank}(R(n)) = n$ also in the even case.

Consequently, for any point p satisfying $p \notin \bigcup_{h=0}^{k-1} V_h$, the matrix F has full column rank k , the completion $B = [F \ G]$ is invertible, and the compressed representation $y = MB^{-1}p$ used by the cone separator is well defined. These results justify the invertibility checks in Section 4.2, which function as numerical safeguards rather than structural requirements in theory.

4.2.3 Cone score and pulled-back quadratic cut

We briefly recall the cone construction from Chapter 3, using the notation needed for the present algorithm.

In Chapter 3, we introduced the admissible family

$$c = 1 - \frac{1}{k^d}, \quad d > 1,$$

all of whose members satisfy the validity condition

$$c \in \left(\frac{k-1}{k}, 1 \right)$$

from Theorem 3.4.1 (Acceptance–Rejection of the Cone Separator).

In the current implementation and experiments of Chapter 4, we use the specific

choice

$$c = 1 - \frac{1}{k^3}.$$

This value satisfies the same validity condition and corresponds to a value of c closer to 1 than the illustrative example $c = 1 - \frac{1}{k^2}$ discussed in Subsection 3.4.2. Since the cone score is

$$S(y) = (k - 1)\|y\|^2 - c(e^\top y)^2,$$

choosing a larger value of c increases the influence of the term $(e^\top y)^2$, which measures how closely the compressed point y is aligned with the distinguished direction e . Consequently, points whose direction is closer in angle to the cone axis e are more likely to satisfy $S(y) < 0$. In geometric terms, this corresponds to a wider forbidden cone around the axis e , while the same acceptance–rejection theorem continues to apply. Accordingly, the computational logs and cuts reported in this chapter correspond to the implementation choice

$$c = 1 - \frac{1}{k^3}.$$

Thus, in the present implementation, the cone matrix is

$$Q_{\text{cone}} = (k - 1)I_k - c e e^\top.$$

Given a candidate point $p \in \mathbb{R}^n$, its compressed coordinates are $y = MB^{-1}p$, and the *cone score* is defined by

$$S(p) = y^\top Q_{\text{cone}} y.$$

Theorem 3.4.1 characterizes the meaning of this score:

$$S(p) < 0 \iff MB^{-1}p \text{ lies strictly inside the forbidden region of the cone.}$$

In this case the point p is certified as violating the present cone test and is therefore separated by adding the pulled-back quadratic cut in the original coordinates. Conversely, $S(p) \geq 0$ indicates only that the candidate is not cut off by this relaxed cone test. It does not by itself prove that p lies in one of the invariant subspaces V_j . In the implementation, this cone-score test is preceded by a separate geometric check for whether p is already numerically close to one of the invariant subspaces, and it is also accompanied by numerical safeguards for cases in which the matrix B cannot be used safely. Thus, the outcome “no cut generated” may arise either because p is treated as already lying in an invariant subspace, because the candidate passes the relaxed cone test, or because no safe pullback cut can be formed from that candidate. See Algorithm 4.3.

We now recall the pullback construction from Subsection 3.4.3 and apply it in the present algorithmic setting. The cone test is formulated in the compressed coordinates

$$y = MB^{-1}x,$$

where the feasible side is given by

$$y^\top Q_{\text{cone}} y \geq 0.$$

Therefore the corresponding quadratic constraint in the original variables is

$$x^\top Q'_{\text{cone}} x \geq 0, \quad Q'_{\text{cone}} = B^{-\top} M^\top Q_{\text{cone}} M B^{-1}.$$

Indeed, for every $x \in \mathbb{R}^n$,

$$x^\top Q'_{\text{cone}} x = (MB^{-1}x)^\top Q_{\text{cone}}(MB^{-1}x),$$

so the quadratic inequality in the original coordinates is exactly the pullback of the cone inequality from the compressed space.

By Theorem 3.4.1 (Acceptance–Rejection of the Cone Separator), the cone inequality leaves every coordinate hyperplane $\{y : y_j = 0\}$ on the feasible side, and hence preserves the images of the invariant subspaces under the compression map. Accordingly, in the cut-and-solve procedure, whenever the cone score satisfies $S(p) < 0$, we add the pulled-back inequality

$$x^\top Q'_{\text{cone}} x \geq 0$$

to exclude the current violating candidate point while preserving the intended multi-subspace feasible structure.

4.3 Pseudocode Description

The separation routine implemented in our SCIP/PYSCIP OPT framework proceeds through a sequence of partial optimization runs. During each solve slice, it monitors candidate points exposed by the solver and evaluates the cone score at those points. After the slice ends, it constructs quadratic cone cuts from the safely identified violating candidates and then re-solves the model with those new cuts added. In the experiments reported in Section 4.3.1, this behavior appears mainly as a repeated one-by-one pattern: one violating candidate followed by one new cut.

Implementation parameters and notation. In Algorithm 4.3, T in $t = 1, \dots, T$ denotes the maximum number of outer iterations, namely, the maximum number of solve–monitor–separate–resolve cycles carried out by the implementation. The parameter $\tau \geq 0$ denotes the cone-score tolerance used in the test $S(p) < -\tau$; in the code this corresponds to the parameter `S_TOL`. We also use a small geometric tolerance $\varepsilon_f > 0$ to detect whether a point already lies in one of the target invariant subspaces; in the present implementation this acceptance tolerance is set to 10^{-8} . For a candidate point p , we form the matrices F and B following the construction in Section 3.1. In particular, F collects the residual vectors $f_h = p - P(h, n)p$, and $B = [F \ G]$ is obtained by completing F with a basis of $\ker(F^\top)$.

In the implementation, this completion is computed numerically by constructing a basis of $\ker(F^\top)$ and orthonormalizing it via QR factorization [13]. For $k = \lfloor n/2 \rfloor + 1$, the cone matrix is taken as

$$Q_{\text{cone}} = (k - 1)I_k - c e e^\top, \quad c = 1 - \frac{1}{k^3}.$$

Algorithm 4.3. Cone-Based Separation Loop in a SCIP/PySCIPOpt Framework.

1. Read the original model M_0 and initialize the stored-cut list $\mathcal{Q} \leftarrow \emptyset$.
2. Set up the SCIP event-based collection mechanism and separator fallback used to collect candidate points exposed during the solve slice.
3. For $t = 1, \dots, T$:
 - (a) Clear the current collection of recorded violating candidates and solve the current cut-augmented model for one time slice.

- (b) During this solve, whenever SCIP exposes a candidate point p , perform the following geometric test:

- i. Build the matrix

$$F = [f_0 \cdots f_{k-1}], \quad f_h = p - P(h, n)p.$$

- ii. Form

$$B = [F \ G],$$

where the columns of G span $\ker(F^\top)$.

- iii. If $\|f_h\| \leq \varepsilon_f$ for some h , treat p as lying in the corresponding invariant subspace and generate no cut from this candidate.

- iv. Otherwise, if B is not invertible, do not generate a cut from this candidate.

- v. Otherwise compute

$$y = MB^{-1}p$$

and the cone score

$$S(p) = y^\top Q_{\text{cone}} y.$$

- vi. If $S(p) < -\tau$, compute the pulled-back quadratic form

$$Q'_p = B^{-\top} M^\top Q_{\text{cone}} M B^{-1},$$

and store Q'_p as a new candidate cut associated with p .

- vii. Otherwise, regard p as nonviolating for the current cone test and generate no cone cut from this candidate.

- (c) After the solve slice ends, stop if the model with the cuts added so far has become infeasible.

- (d) After the solve slice ends, examine all violating candidates recorded during that slice and keep the cone-cut matrices Q'_p that can be constructed safely.
- (e) If the current slice produces no new safe cone cut, terminate the outer loop.
- (f) Leave the current transformed solving stage,¹ add all new safely constructed quadratic cone cuts

$$x^\top Q'_p x \geq 0$$

to the current model, and store their matrices in $\mathcal{Q}$.

- (g) If the current slice ended by time limit, enlarge the slice length up to the prescribed maximum and continue.

Because the quadratic cone separators used here are added as nonlinear constraints, the implementation is organized as an outer solve–monitor–separate–resolve loop rather than as a single-call branch-and-cut routine. Here, the outer loop means the repeated cycle in which the current model is solved for one time slice, candidate points are monitored during that slice, the violating ones are processed afterward, any new safe cone cuts are added, and the updated model is then solved again.

Why the implementation is not a standard cut-callback scheme. The final implementation does not use the usual “generate a cut immediately inside a callback” pattern. The reason is that our separator produces nonlinear quadratic constraints of the form $x^\top Q'x \geq 0$, and these constraints cannot be incorporated as ordinary linear cuts within the standard callback-based separation mechanism. During each solve

¹In the implementation, this is done by calling `freeTransform()` before adding new nonlinear quadratic constraints. This is not a full rebuild from scratch; it exits the current transformed solving stage so that the newly constructed cone cuts can be added safely to the model.

slice, candidate points exposed by SCIP are evaluated by the geometric module, and any violating candidates are recorded for later cut generation. After the solve slice ends, the algorithm exits the current transformed solving stage and only then adds the newly constructed quadratic constraints before resolving. This outer-loop step is not a full rebuild from scratch of the optimization model; rather, the implementation leaves the current transformed solving stage, adds the new nonlinear constraints, and then continues optimization on the updated model.

SCIP interfaces used in the implementation. The implementation uses several SCIP/PySCIPOpt interfaces, each for a different purpose. The model interface is used to read the instance, solve the current formulation, adjust the time limit for each outer slice, leave the transformed solving stage by `freeTransform()`, and add the newly constructed quadratic constraints. The event handler, and separator fallback are used mainly to access candidate points during the solve. In particular, the event handler monitors `BESTSOLFOUND` and `LPSOLVED`,² while the separator fallback inspects the current LP solution through the separator interface and records the point when it violates the cone test.

Why an event handler is used. The event handler is used to provide more opportunities to observe candidate points during a solve slice. In particular, it allows the implementation to inspect newly found incumbents and relevant solver states while the search is still in progress, rather than waiting only for the end of the solve. An initial implementation explored a custom constraint-handler approach for collecting candidate points. However, due to technical difficulties with the built-in constraint handler, the final implementation relied primarily on the event handler,

²`BESTSOLFOUND` is triggered when SCIP finds a new incumbent solution, while `LPSOLVED` is triggered after an LP relaxation has been solved. In our implementation, these two events provide opportunities to inspect incumbent and LP-based candidate points during the search.

together with the separator fallback, to collect candidate points for later separation.

Conservative acceptance rules. The implementation also includes two conservative acceptance mechanisms. First, if some residual $f_h = p - P(h, n)p$ is numerically zero, then p is accepted immediately and treated as lying near the corresponding invariant subspace, i.e., close to the associated supporting hyperplane. Second, if the matrix B is not invertible, then no cut is generated from that candidate. This is not an “error stop” in the sense of algorithmic failure; rather, it is a deliberate decision to avoid constructing a separator from a point for which the pulled-back quadratic form would not be numerically trustworthy.

Practical implementation lessons. Several practical lessons emerged during the implementation. First, for nonlinear separation it is useful to distinguish clearly between three tasks: monitoring candidate points, performing the geometric test, and inserting newly generated constraints. Early attempts tended to merge these tasks into a single callback-stage action, but the final implementation became much more stable once they were separated. Second, different SCIP interfaces can play complementary roles. In our implementation, candidate points are exposed primarily through the event handler and the separator fallback, even though the actual insertion of nonlinear cone constraints occurs later in the outer loop. An initial constraint-handler approach was also explored, but the final implementation relied on the event-based mechanism for collecting candidate points. Third, conservative numerical safeguards are important. In particular, when a point is detected as lying near one of the target invariant subspaces, or when the matrix B cannot be inverted safely, the implementation prefers to generate no cut rather than risk inserting an invalid separator.

4.3.1 Explicit Reformulation versus Cone Method

In this subsection, we compare two closely related approaches and then extend that comparison through additional cyclic instances and follow-up parameter studies. In the earlier framework of Shahverdi, Banihashemi, and Bremner [3], the relevant symmetry-based restrictions are imposed in the corresponding `*-binary.lp` reformulation. In the present implementation, we instead start from the matching original plain `.lp` model and enforce related restrictions implicitly during the solve by monitoring candidate points and adding cone-based quadratic cuts when needed.

For this reason, the comparison should not be read as a direct solver ranking on exactly the same input formulation. Rather, it compares an explicit reformulation, represented by the `*-binary.lp` instances, with an implicit cut-and-solve version based on the corresponding original `.lp` instances. This is the closest comparison to the earlier framework of [3], where the related restrictions are built into the model in advance.

This is also important for the labels “feasible” and “infeasible.” These labels are inherited from the original benchmark family and describe whether the underlying instance is feasible overall before additional symmetry-based restrictions are imposed. Accordingly, the original `.lp` instances and the corresponding `*-binary.lp` reformulations inherit the same family label, even though they represent different formulations. Consequently, an instance labelled “feasible” in the original dataset may still become infeasible after repeated cone cuts are added. To avoid confusion, we therefore refer to such cases as *instances labelled feasible in the original dataset*.

Computational environment. All computational experiments were performed on a machine equipped with an Intel Core i9-13900H processor (2.60 GHz) and 16 GB of RAM, running Windows 11. The implementation was carried out in Python using

PySCIPOpt (version 5.4.1), interfaced with the SCIP optimization solver (version 9.0.0).

For the explicit baseline, plain SCIP was run on the corresponding `*-binary.lp` instances. For the cone-based method, the cut-and-solve implementation was run on the matching original `.lp` instances. The default runs use `time-slice=2.0` and `max-iters=10`. When a slice ends by time limit, the time slice may be increased in later outer iterations. In addition to the main comparison instances, further experiments were carried out on additional cyclic instances and on reruns with modified parameter settings in order to clarify the stopping behavior of the cone method and to obtain a cleaner explicit-reformulation baseline for the larger binary cases.

To make the comparison reproducible, Table 4.1 summarizes the main run settings used for the explicit-reformulation baseline and for the cone method.

Table 4.1: Run settings used in the comparison and follow-up experiments.

Method	Default setting	Notes
Explicit reformulation baseline	plain SCIP on the corresponding <code>*-binary.lp</code> instance	Most binary baseline runs solved directly. For the larger binary cases examined in the follow-up experiments, plain SCIP was also run with a 1-hour time limit.
Cone-based cut-and-solve method	original plain <code>.lp</code> instance, <code>time-slice=2.0</code> , <code>max-iters=10</code>	Event-based outer-loop runs. Each outer slice is solved under the SCIP time limit corresponding to that slice.
Tuned cone run	larger <code>max-iters</code>	Used for CF10. The default run stopped at the outer-iteration cap, while increasing <code>max-iters</code> to 20 produced infeasibility. A 40-iteration run gave the same final status and is not reported separately.
Additional cone studies	larger <code>time-slice</code> and/or larger <code>max-iters</code>	Used for the additional cyclic instances in order to distinguish outer-iteration limits from time-limited slices and to test whether longer slices changed the observed behavior.

For the explicit baseline, plain SCIP was run on the corresponding `*-binary.lp` reformulation. For the cone method, the code was run on the matching original `.lp`

instance using the default slice length and iteration budget unless otherwise noted. The main comparison is based on the instances CF5, CI5, P1, CF10, CF15, and CI15, while the later tables in this subsection extend the picture to additional cyclic instances and follow-up runs.

Since the comparison is between two related but different formulations, it is also useful to record the model size of each formulation. Table 4.2 therefore gives the basic model characteristics for the original plain instances and for the corresponding explicit `*-binary.lp` reformulations.

Table 4.2: Model characteristics of the original instances and the corresponding explicit reformulations.

Instance	Original plain <code>.lp</code>					Explicit reformulation <code>*-binary.lp</code>				
	Total vars	Binary	General int.	Linear cons.	Quad. cons.	Total vars	Binary	General int.	Linear cons.	Quad. cons.
CF5	5	0	5	8	0	8	3	5	11	2
CI5	5	0	5	11	0	8	3	5	14	2
P1	5	0	5	6	0	8	3	5	9	2
CF10	10	0	10	13	0	16	6	10	18	4
CF15	15	0	15	18	0	23	8	15	21	7
CI15	15	0	15	31	0	23	8	15	34	7

Instance codes: CF5 = `cyclic_feasible_5_0`, CI5 = `cyclic_infeasible_5_0`, P1 = P1, CF10 = `cyclic_feasible_10_0`, CF15 = `cyclic_feasible_15_0`, CI15 = `cyclic_infeasible_15_0`. For each family, the explicit reformulation is the corresponding `*-binary.lp` instance.

Table 4.2 shows clearly that the original plain instances and the explicit reformulations are not the same optimization models. The original family consists of general-integer linear models only, whereas the `*-binary.lp` reformulations introduce additional binary variables and quadratic constraints. This difference is exactly why the computational results in this subsection should not be read as a direct solver ranking on one common formulation. Rather, they compare an explicit reformulation with the present cut-and-solve implementation applied to the matching original instance.

With this distinction in place, Table 4.3 reports the main comparison: plain SCIP on the explicit reformulation and the cone method on the corresponding original plain

instance.

Table 4.3: Main comparison between the explicit reformulation baseline and the cone method.

Instance	Plain SCIP on <code>*-binary.lp</code>	Nodes	Cone run setting	Cone run status	Integer inc.	Cuts added
CF5	infeasible	17	default	infeasible	yes	5
CI5	infeasible	17	default	infeasible	no	0
P1	infeasible	2324	default	timelimit	no	0
CF10	infeasible	1483	tuned (<code>max-iters=20</code>)	infeasible	yes	10
CF15	timelimit	47292	default	timelimit	yes	6
CI15	timelimit	116496	default	timelimit	no	0

Notes: “Nodes” denotes the number of branch-and-bound subproblems explored by SCIP during the solve. “Integer inc.” means whether the cone run produced an integer incumbent that could be tested by the cone separator. “Default” means `time-slice=2.0` and `max-iters=10`. For CF10, the tuned run with `max-iters=20` is reported because the default run stopped at the outer-iteration cap; infeasibility was proved only after increasing `max-iters`. For CF15 and CI15, the plain SCIP baseline entries shown here are the 1-hour reruns.

Table 4.3 should be interpreted as a comparison between two related but different ways of handling the same underlying symmetry-based restriction. In the explicit baseline, that restriction is already built into the `*-binary.lp` reformulation. In the present method, the solver starts from the original plain `.lp` model, and related restrictions are introduced only when violating candidate points are detected during the event-based outer loop.

Table 4.3 shows that the cone method is most informative when the solver actually produces integer incumbents. In those cases, the solver events expose candidate incumbents, the cone score is evaluated at those points, and a quadratic cone cut is added whenever a violating incumbent is found. Thus, the logs should be read as a repeated cycle of the form

incumbent $\longrightarrow$ cone test $\longrightarrow$ violating candidate $\longrightarrow$ new cut $\longrightarrow$ re-solve.

When no integer incumbent is produced, the cone-based separator remains inactive, because no violating incumbent becomes available for the cone test.

The instances labelled feasible in the original dataset show this mechanism most clearly. The logs do not suggest that SCIP visits incumbents in a fixed circular order. Rather, they show that the same sparse value pattern reappears at different cyclic positions. In other words, the nonzero values stay the same, while their locations among the variables shift. The cone separator is therefore not reacting to unrelated incumbents; it is repeatedly cutting shifted representatives of the same basic pattern.

The 5-variable case gives the clearest small example.

Consider the instance `cyclic_feasible_5.0.1p`. It is governed by one cyclic family of five inequalities together with the fixed-sum condition

$$x_1 + x_2 + x_3 + x_4 + x_5 = 4.$$

Since the objective is also $x_1 + \dots + x_5$, every feasible solution has objective value 4. Thus, in this instance, the main point of the logs is not variation in the objective value, but the repeated appearance of cyclic shifts of the same sparse feasible pattern. In the first outer iteration, SCIP finds the incumbent

$$p = (0, 0, 0, 1, 3)^\top$$

with objective value 4, and the cone test returns

$$S(p) = -2.666667 < 0.$$

So this incumbent is recorded as violating and one cone cut is prepared. Later in the run, SCIP finds another incumbent

$$p = (3, 0, 0, 0, 1)^\top,$$

again with objective value 4 and again with the same negative cone score. The two incumbents therefore have the same value pattern $(1, 3)$, but at different cyclic positions. After five cone cuts have been added, the next rebuild produces no incumbent, and the run proves infeasibility. Thus, in CF5, the shifted incumbent family is cut to completion.

The 10-variable case shows the same mechanism on a larger shifted family. In the initial slice, SCIP finds

$$p = (0, 0, 0, 0, 0, 0, 0, 0, 1, 8)^\top$$

with objective value 9, and again the cone score is

$$S(p) = -5.833333 < 0.$$

In the default run, the process continues until the outer-iteration cap is reached. At that point, the final incumbent is

$$p = (0, 1, 8, 0, 0, 0, 0, 0, 0, 0)^\top,$$

which has the same nonzero values $(1, 8)$ at different cyclic positions. Thus, the default run stops because the iteration budget is exhausted, not because the shifted family has necessarily been exhausted. When `max-iterations` is increased to 20, the longer run proves infeasibility after 10 cuts. This supports the practical observation that the outer-iteration budget should be at least comparable to the number of variables in these cyclic instances.

The 15-variable case shows the same shifted-family behavior in its early phase, but it also illustrates the larger-instance behavior of the method. In the initial slice, SCIP finds

$$p = (13, 0, 0, \dots, 0, 1)^\top$$

with objective value 14, and again the cone score is negative:

$$S(p) = -7.875 < 0.$$

So the incumbent is violating and a cone cut is added. However, unlike the 5-variable and 10-variable cases, the process does not run to completion within the present run settings. Instead, later slices end by time limit, and the run stops with cuts already added but without proving infeasibility. So, in this larger instance, the method still reveals the symmetry-related incumbent family clearly, but it stops earlier than in the smaller feasible cases.

By contrast, when no integer incumbent is produced, the cone separator remains inactive. This is what happens in CI5, P1, and CI15. These cases therefore show that the present implementation is most informative when the solver actually produces incumbents for the cone test.

The same general pattern also appears on additional cyclic instances from the same benchmark family. Table 4.4 summarizes the default cone runs on those additional original plain instances.

Table 4.4: Default cone runs on additional cyclic instances.

Instance	Final status	Outer iters	Integer incumbents	Cuts added
CF11	unknown	10	yes	10
CF12	timelimit	4	yes	3
CF13	unknown	10	yes	10
CF14	timelimit	4	yes	3
CI10	infeasible	1	no	0
CI11	timelimit	1	no	0
CI12	timelimit	1	no	0
CI13	timelimit	1	no	0
CI14	timelimit	1	no	0

Table 4.4 shows that the same overall behavior continues on the additional cyclic instances. Among the instances labelled feasible in the original dataset, CF11 and CF13 reached the outer-iteration cap after ten successful cut rounds, while CF12

and CF14 each added three cuts and then stopped when the next slice ended by time limit with no violating candidate. Among the instances labelled infeasible in the original dataset, CI10 was proved infeasible in the first slice, whereas CI11–CI14 all stopped in the first slice by time limit, with no recorded incumbent and no cone cut.

To clarify whether these stopping patterns were caused mainly by the outer-iteration cap or by limited time per slice, selected follow-up reruns were carried out with modified parameter settings. The corresponding outcomes are summarized in Table 4.5.

Table 4.5: Effect of parameter changes on selected cone runs.

Instance	Changed setting	Observed effect
CF11	larger max-iters	changed from an outer-iteration stop to infeasible after 11 cuts
CF12	larger time-slice	no change in operational outcome
CF13	larger max-iters only	no improvement beyond the shorter run
CF13	larger time-slice together with larger max-iters	improved to 12 successful cut rounds, then plateaued
CF14	larger time-slice	no change in operational outcome
CI11–CI14	larger time-slice	no change in operational outcome

The follow-up experiments show that the effect of parameter changes is strongly instance-dependent. Increasing **max-iters** changed the outcome of CF11 from an outer-iteration stop to infeasibility after 11 cuts. By contrast, increasing only the time slice did not change the observed behavior of CF12 or CF14. The most informative case is CF13: increasing **max-iters** alone did not extend the run beyond eight cuts, whereas increasing **time-slice** together with **max-iters** increased the run to twelve successful cut rounds; however, further increases in the time slice did not push the run beyond that point. For CI11–CI14, increasing the time slice from 10 to 30 did not change the observed outcome: the first slice still ended by time limit in every case, with no violating candidate and no cut added.

A particularly notable outcome occurred for CF11. Increasing **max-iters** allowed

the cone-based approach to continue generating cuts beyond the shorter run and eventually reach an infeasible conclusion after 11 cuts. By contrast, the corresponding explicit reformulation did not reach an infeasible conclusion in the reported plain-SCIP experiments. Although this is only a single instance, it illustrates that additional cone cuts can sometimes provide useful information beyond what is obtained in the shorter runs.

The explicit-reformulation baseline was also extended on the larger binary cases. Earlier plain-SCIP runs on several of these binary reformulations had terminated with insufficient-memory errors. To obtain a cleaner and more uniform baseline interpretation, the unresolved binary cases were rerun under a 1-hour plain-SCIP policy. Table 4.6 records the resulting outcomes.

Table 4.6: Plain SCIP on binary reformulations under a 1-hour time limit.

Instance	Final status	Solutions	Dual bound	Nodes
CF11-binary	timelimit	0	10	7147057
CF12-binary	timelimit	0	11	1165336
CF13-binary	timelimit	0	12	195428
CF14-binary	timelimit	0	13	140149
CF15-binary	timelimit	0	14	47292
CI11-binary	timelimit	0	10	4169512
CI12-binary	timelimit	0	11	830387
CI13-binary	timelimit	0	12	1365327
CI14-binary	timelimit	0	13	254864
CI15-binary	timelimit	0	14	116496

Notes: Here, *nodes* denotes the number of branch-and-bound subproblems explored by SCIP during the solve, and *dual bound* denotes the best objective bound proved by SCIP during the branch-and-bound search.

Table 4.6 shows that, in contrast to the earlier runs that ended with insufficient-memory errors, these reruns terminated cleanly by time limit. In all cases shown here, SCIP reported zero feasible solutions, so no incumbent was found within the 1-hour window. For completeness, `cyclic_infeasible_10_0-binary.lp` had already been solved earlier by plain SCIP and proved infeasible, so it is not included among the 1-hour time-limited reruns.

Overall, the experiments indicate that the cone method is most useful as a structural cutting procedure. Its logs do not merely show isolated incumbents; rather, on the cyclic instances they reveal repeated shifted versions of the same sparse incumbent pattern. This allows the method to generate cone cuts that remove symmetry-related representatives one by one. The additional cyclic instances and follow-up reruns reinforce the same interpretation: on the original plain instances, the method is most informative when the solver produces incumbents that can be tested and cut, and the effect of tuning depends strongly on the instance. On the explicit reformulations, the revised 1-hour plain-SCIP policy gives a cleaner baseline interpretation for the larger binary cases: rather than terminating with memory errors, these runs can now be reported uniformly as time-limited. In this sense, the experiments support the cone method as a way of extracting symmetry-related information from incumbent solutions and converting it into additional cuts during the search, while also providing a more stable computational picture of the larger cyclic instances.

This chapter presented the implementation of the cone-based separator within a cut-and-solve framework using SCIP and PySCIPOpt. The computational experiments illustrated how the proposed geometric cuts interact with the optimization process across several cyclic-symmetry instances and highlighted the situations in which the method is most informative. The final chapter summarizes the main contributions of the thesis and discusses directions for future work.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This thesis developed a geometric quadratic separator for integer programs with cyclic symmetry and integrated it into a cut-and-solve framework in SCIP through PySCIPOpt. The construction begins with the geometry of invariant subspaces, first in the two-subspace setting and then in the more general multi-subspace framework, and leads to a cone-based quadratic inequality that can be pulled back to the original coordinates and added as a valid nonlinear cut. In this way, the thesis connects the mathematical structure of symmetry-invariant subspaces with an implementable separation procedure inside an optimization solver.

The computational study in Chapter 4 shows that the cone method is most informative when the solver produces integer incumbents during the search. In those cases, the procedure repeatedly exposes candidate incumbents, evaluates the cone score, records violating points, adds quadratic cone cuts, and then resolves the updated model. Across the cyclic instances, the logs show that the method is reacting to repeated shifted versions of the same sparse incumbent pattern rather than to un-

related incumbents. The additional cyclic instances and follow-up reruns support the same interpretation: the method is most effective when incumbents are exposed, while the effect of tuning is instance-dependent and becomes more important on the larger cyclic cases. In one follow-up experiment (CF11), increasing the number of outer iterations allowed the cone-based approach to continue generating cuts and eventually conclude infeasibility. By contrast, the corresponding explicit reformulation was not solved within the one-hour time limit. This illustrates that additional cone cuts can sometimes provide useful information beyond the shorter runs.

The experiments also show an important feature of the present implementation. When no integer incumbent is produced, the cone separator remains inactive, so the method is currently most effective as a structural cutting procedure in runs where the solver exposes incumbents that can be tested against the cone.

Overall, this thesis contributes both a theoretical and a computational framework for integer programs with cyclic symmetry. On the theoretical side, it clarifies the Fourier-based invariant-subspace structure associated with cyclic symmetry, including the role of the Fourier mode spaces in providing an orthogonal direct-sum decomposition of the ambient space and the relationship between these mode spaces and the invariant subspaces preserved by the separator. Building on this structure, the thesis develops a compressed-coordinate representation in which the images of the preserved invariant subspaces lie in coordinate hyperplanes and extends the two-subspace separator for the Fixed space and Layer 0 to a cone-based separator for multiple invariant subspaces.

On the computational side, the results support the cone method as a geometric cutting procedure based on symmetry-derived invariant subspaces. Its main computational contribution is not simply to replace plain SCIP on every instance, but to make orbit-like structure visible during the search and to turn that structure into

additional quadratic cuts. In this sense, the method provides both a geometric interpretation of symmetry-based separation and a practical framework for integrating that interpretation into integer-programming search.

5.2 Future Work: Cubic and Logarithmic barrier-based separators

The results of Chapter 4 suggest several natural directions for future work. First, it would be useful to strengthen the separation framework for cases in which no integer incumbent is produced, so that the method can draw more information from LP points and other intermediate solver states. Second, the experiments indicate that the outer-loop settings can affect the final behaviour of the method, especially on larger cyclic instances, so a more systematic study of time-slice growth, stopping conditions, and cut management would be valuable. Third, a natural next step is a more systematic computational comparison between the implicit cut-and-solve approach used here, the earlier explicit reformulation framework based on binary-controlled constraints, and a direct disjunctive formulation in SCIP. In the present implementation, the union-of-subspaces condition is not imposed as an explicit disjunction; instead, it is approximated through the cone-based quadratic separator. Since SCIP supports disjunctive constraints, future work could evaluate whether directly modeling the alternatives

$$x \in V_0 \vee x \in V_1 \vee \cdots \vee x \in V_{k-1}$$

performs better or worse than the single cone-based quadratic relaxation used in this thesis. Finally, the quadratic cone separator developed in this thesis naturally motivates the study of richer separator families, including cubic and logarithmic

barrier-based constructions in the compressed coordinates y , which may capture more detailed geometric interactions in that transformed space. In particular, one may consider the constraint

$$\log\left(\prod_i y_i\right) \leq -\varepsilon,$$

or equivalently

$$\sum_i \log(y_i) \leq -\varepsilon,$$

for some positive ε . Since in the compressed coordinates the reference point p is mapped to e , this type of constraint would separate p from the coordinate hyperplanes. It may therefore provide another way to encode the intended exclusion region in the transformed space. At present, however, it is not clear how effectively a sum-of-logs constraint can be handled within SCIP, so this remains an interesting direction for further study. These directions continue the main theme of the thesis: using the geometry of symmetry-invariant subspaces to guide the design of effective separation procedures.

5.2.1 Cubic Separators

Cubic Separators in Compressed Coordinates Quadratic separators of the form $S(x) = x^\top Q'x$ have already proven effective at distinguishing the reference point p from structured subspaces such as the fixed space and Layer 0 (see Section 3.3). However, the quadratic construction captures only *pairwise* interactions. When the geometry involves *three* coordinates interacting simultaneously, a natural generalization is a cubic separator, which vanishes whenever any relevant coordinate equals zero.

Although both constructions are motivated by the same multi-subspace geometry in the compressed coordinates, they are algebraically different. The quadratic cone

separator is the implemented method of this thesis, while the cubic and degree- k polynomial cases are proposed only as future extensions.

As in the quadratic case, we first project to a low-dimensional coordinate system.

Let $B \in \mathbb{R}^{n \times n}$ be the basis constructed from subspace projections, and define

$$y = MB^\top x, \quad M = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \end{bmatrix}.$$

Thus every $x \in \mathbb{R}^n$ is mapped to $y = (y_0, y_1, y_2) \in \mathbb{R}^3$. In this space, each invariant subspace corresponds to a coordinate hyperplane: points in V_0 map into $\{y : y_0 = 0\}$, points in V_1 map into $\{y : y_1 = 0\}$, and points in V_2 map into $\{y : y_2 = 0\}$.

The unnormalized cubic form

$$\tilde{S}(y) = y_0 y_1 y_2$$

vanishes whenever one coordinate is zero, hence it vanishes on the union of the coordinate hyperplanes $\{y : y_0 = 0\}$, $\{y : y_1 = 0\}$, and $\{y : y_2 = 0\}$. To normalize at the reference point p , with $y(p) = (y_0(p), y_1(p), y_2(p))$, define

$$\hat{S}(y) = \frac{y_0 y_1 y_2}{y_0(p) y_1(p) y_2(p)}.$$

This guarantees $\hat{S}(p) = 1$. Geometrically, the level set $\hat{S}(y) = 1$ passes through p and the cubic form vanishes on the coordinate hyperplanes associated with the invariant subspaces.

Constraint Form and Generalization A cubic separator can be used as a feasibility constraint of the form

$$\widehat{S}(x) \leq \delta, \quad 0 < \delta < 1.$$

This ensures that feasible solutions satisfy the separator inequality in the compressed coordinates; in particular, the invariant-subspace hyperplanes remain feasible (since $\widehat{S} = 0$ there), while the reference point p is excluded for any $0 < \delta < 1$ (since $\widehat{S}(p) = 1$).

More generally, a natural generalization is a degree- k polynomial

$$S(y) = \prod_{i=0}^{k-1} y_i,$$

which vanishes whenever any coordinate equals zero (i.e., on the union of the coordinate hyperplanes $y_i = 0$). The cubic case ($k = 3$) is the simplest nontrivial example.

Integration with optimization (SCIP). A natural next step is to study how cubic constraints could be handled in SCIP. This remains a topic for future work and may require reformulation or approximation, for example by introducing auxiliary variables to rewrite the cubic expression or by replacing the cubic constraint with a simpler one.

5.2.2 Open Questions for Further Exploration

The cubic and logarithmic barrier-based constructions discussed above suggest several natural questions for future study. For cubic separators, an important question is how such constraints should be represented within SCIP. It is also natural to ask how

cubic separators would behave computationally in comparison with the quadratic cone separator, both in terms of the cuts they produce and in terms of overall solver effort. Since higher-degree expressions are generally more sensitive to perturbations, questions of numerical stability would also need to be studied carefully.

For logarithmic barrier-based separators, the main open question is how well non-convex constraints such as

$$\sum_i \log(y_i) \leq -\varepsilon$$

can be supported in practice within SCIP. If such constraints can be handled effectively, it would then be important to study whether they provide useful separation behaviour in the same compressed-coordinate setting as the polynomial separators.

These questions point to a broader direction for future work: understanding how different geometric separator families can be represented, solved, and compared within practical optimization frameworks.

Bibliography

- [1] N. Shahverdi. *Outer Approximations of Core Points for Integer Programming*. PhD thesis, University of New Brunswick, 2021. Available at: <https://unbscholar.lib.unb.ca/handle/1882/13165>.
- [2] S. Banihashemi. *Symmetric Integer Linear Programming with a Core Point Approach*. Master’s thesis, University of New Brunswick, 2024. Available at: <https://unbscholar.lib.unb.ca/handle/1882/38227>.
- [3] N. Shahverdi, S. Banihashemi, and D. Bremner. *Outer Approximations of Core Points for Integer Programming*. In: *Data Science and Optimization*, Springer, 2026, pp. 175–207.
- [4] F. Margot, *Exploiting orbits in symmetric ILP*, *Mathematical Programming*, **98**(1–3):3–23, 2003. DOI: 10.1007/s10107-003-0427-7.
- [5] R. Bödi, K. Herr, and M. Joswig. *Algorithms for Highly Symmetric Linear and Integer Programs*. *Mathematical Programming*, 137(1–2):65–90, 2013. doi:10.1007/s10107-011-0487-6.
- [6] K. Herr, T. Rehn, and A. Schürmann. Exploiting Symmetry in Integer Convex Optimization Using Core Points. *Operations Research Letters*, 41(3):298–304, 2013.

- [7] R. K. Clancy, *Constructing Realizations of Abstract Regular Polytopes from Character Tables*, MSc thesis, UNB, 2005.
- [8] J. Kronqvist, R. Misener, and C. Tsay, *P-split formulations: a class of intermediate formulations between big-M and convex hull for disjunctive constraints*, Mathematical Programming, 2025. doi:10.1007/s10107-025-02232-1.
- [9] E. Balas and A. M. Kazachkov, *V-polyhedral disjunctive cuts*, Mathematical Programming, 2025. doi:10.1007/s10107-025-02227-y.
- [10] D. C. Lay, S. R. Lay, J. J. McDonald, *Linear Algebra and Its Applications*, 5th ed., Pearson, 2016.
- [11] E.M. Stein and R. Shakarchi, *Fourier Analysis: An Introduction*, Princeton University Press, 2003.
- [12] R.F. Scott, *A Treatise on the Theory of Determinants and Their Applications in Analysis and Geometry*, Cambridge University Press, 1880.
- [13] G. Strang. *Linear Algebra and Its Applications*. 5th ed., Wellesley-Cambridge Press, 2016.
- [14] R. A. Horn and C. R. Johnson. *Matrix Analysis*. 2nd ed., Cambridge University Press, 2012.
- [15] G. M. Ziegler. *Lectures on Polytopes*. Springer-Verlag, New York, 1995.
- [16] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
- [17] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1986.
- [18] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley–Interscience, 1988.

- [19] J. J. Rotman. *An Introduction to the Theory of Groups*. 4th ed., Springer, 1995.
- [20] J.-P. Serre. *Linear Representations of Finite Groups*. Springer, 1977.
- [21] P. J. Davis. *Circulant Matrices*. 2nd ed., AMS Chelsea Publishing, 1994.
- [22] R. M. Gray. Toeplitz and Circulant Matrices: A Review. *Foundations and Trends in Communications and Information Theory*, 2(3):155–239, 2006.
- [23] T. S. Shores. *Applied Linear Algebra and Matrix Analysis*. Undergraduate Texts in Mathematics, Springer, New York, 2007.
- [24] T. Rehn. *Exploring core points for fun and profit — a study of lattice-free orbit polytopes*. PhD thesis, Universität Rostock, 2014.
- [25] PySCIPOpt Developers. *Non-Linear Expressions*. PySCIPOpt Documentation (stable), Zuse Institute Berlin, 2024. Available at <https://pyscipopt.readthedocs.io/en/stable/tutorials/expressions.html>.
- [26] Racket Language Documentation. *Racket Documentation*. Available at: <https://docs.racket-lang.org/>.
- [27] K. Bestuzheva, A. Gleixner, and T. Achterberg. *Efficient Separation of RLT Cuts for Implicit and Explicit Bilinear Products*. In A. Del Pia and V. Kaibel (eds.), *Integer Programming and Combinatorial Optimization (IPCO 2023)*, Lecture Notes in Computer Science, vol. 13904, Springer, Cham, 2023, pp. 14–28. doi:10.1007/978-3-031-32726-1_2.
- [28] K. Bestuzheva, A. Chmiela, B. Müller, F. Serrano, S. Vigerske, and F. Wegscheider. *Global optimization of mixed-integer nonlinear programs with SCIP 8*. *Journal of Global Optimization*, 91:287–310, 2025. doi:10.1007/s10898-023-01345-1.
- [29] S. Axler. *Linear Algebra Done Right*. 4th ed., Springer Nature Switzerland AG, Cham, 2024. DOI: 10.1007/978-3-031-41026-0.

- [30] A. Jeffrey and H. H. Dai. *Handbook of Mathematical Formulas and Integrals*.
4th ed., Academic Press, 2008.

Curriculum Vitae

Candidate's full name: Seyedeh Maryam Abbasnia

Education:

1. M.S., Applied Mathematics-Numerical Analysis, University of Guilan, Iran
(Thesis-based, (2015–2018))
2. B.S., Mathematics and Applications, University of Guilan, Iran (2010–2014)

Publications:

13th International Conference on Information Technology, Computer & Telecommunication, Tbilisi, Georgia, 2021.

An efficient implementation of the fourth-order compact finite difference program to solve the Poisson equation with Dirichlet boundary conditions.

Work Experience:

1. Digital Marketing Intern, Access Drive; Summer 2026
2. Research Assistant, University of New Brunswick; Jan 2023–June 2026
3. Teaching Assistant, University of New Brunswick; Jan 2023–April 2026

Awards:

Second Prize (Growth Category), RBC Student Pitch Competition, J. Herbert Smith Centre, University of New Brunswick; Mar 2026.

Presentations:

1. “Geometric Quadratic Separators for Symmetric Integer Programs,” 3-Minute Thesis Presentation (3MT), Faculty of Computer Science Research Expo, University of New Brunswick, April 2026.
2. “OptimaTech Fleet Intelligence: An Explainable Anomaly Detection Prototype for GPS and Logistics Data,” Graywolf Dual-Use Hackathon, University of New Brunswick, Fredericton, NB, May 2026.

Volunteer Experience:

1. Graduate Student Representative (Volunteer), Senate Standing Committee on Teaching Excellence & Policy, UNB; 2025–2026
2. Volunteer Mathematics Teacher, Fredericton High School; Oct 2025–Present

Courses, Workshops, and Certifications:

1. Digital Marketing Course for Beginners, IBM SkillsBuild / Udemy Business; 2026
2. AI in Digital Marketing, IBM SkillsBuild / Udemy Business; 2026
3. Power BI Masterclass Completion, Access Drive Innovation & McKenna Institute; 2025
4. Machine Learning, Stanford University (Online Course); 2021
5. Python, University of Michigan (Online Course); 2021
6. Algorithm Workshop, University of Guilan; 2021