# Numerical Range Over Finite Fields

BY
Timothy Lund

Submitted in partial fulfillment of the requirement for
Major Honors in Mathematics

Houghton University, Houghton, NY
May 2023

Honors Committee

Dr. Rebekah Yates, Project Advisor

Dr. Katrina Koehler, Reader 1

Dr. Brandon Bate, Reader 2

# Contents

# 1 Introduction

In this honors project, we examine the numerical ranges of square matrices with entries from finite fields, specifically $\mathbb{Z}_p[i]$. The classical numerical range over the complex numbers was first defined by Toeplitz [1] in 1918, and further developed with Hausdorff [2]. Then in 1951, Kippenhahn defined the boundary generating curve and classified the numerical ranges of all $3 \times 3$ matrices over the complex numbers [3] [4]. However, very little investigation has been done on numerical ranges over finite fields, with the first of formalization of the finite field numerical range proposed by Coons, et al. in 2016 [5]. This project will examine an emerging area of math research, drawing from established mathematical disciplines such as linear algebra, abstract algebra, and matrix analysis.

Following the work done by Coons, et al., we consider the numerical range of matrices over finite fields $\mathbb{Z}_p[i]$. A majority of our focus will be dedicated to matrices with nonzero eigenvectors $\boldsymbol{x} \in \mathbb{Z}_p[i]^n$ with the property that $\boldsymbol{x}^*\boldsymbol{x} = 0$. Such vectors, termed self-orthogonal vectors in [6], cannot exist over the complex numbers and have only recently garnered interest in the research on finite field numerical ranges. Prior to [7], matrices with self-orthogonal eigenvectors were explicitly excluded from consideration. We seek to describe the structures of numerical ranges of certain matrices with self-orthogonal eigenvectors and note additional properties of self-orthogonal vectors encountered in our investigation.

## 1.1 Key Terms

To understand the concept of finite field numerical range, we must recall the following definitions.

**Definition 1.1.** *A* field $\mathbb{F}$ *is a set of numbers with multiplication* $(\cdot)$ *and addition* $(+)$ *defined on the set such that for all* $a, b, c \in \mathbb{F}$*, the following properties hold:*

- *Closure:* $a + b \in \mathbb{F}$*, and* $a \cdot b \in \mathbb{F}$*,*
- *Associativity:* $(a + b) + c = a + (b + c)$*, and* $(a \cdot b) \cdot c = a \cdot (b \cdot c)$*,*
- *Commutativity:* $a + b = b + a$*, and* $a \cdot b = b \cdot a$*,*
- *Existence of identity elements: there exist* $0, 1 \in \mathbb{F}$ *such that* $a + 0 = a$ *and* $a \cdot 1 = a$*,*
- *Existence of additive inverses: there exists* $-a \in \mathbb{F}$ *such that* $a + (-a) = 0$
- *Existence of multiplicative inverses: for* $a \neq 0$*, there exists* $a^{-1} \in \mathbb{F}$ *such that* $a \cdot a^{-1} = 1$*, and*
- *Distributivity:* $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$*.*

The set of $n \times n$ matrices with elements drawn from the field $\mathbb{F}$ is denoted $M_n(\mathbb{F})$. As a specific example of a field, we consider $\mathbb{Z}_p[i]$ defined below.

**Definition 1.2.** *The set* $\mathbb{Z}_p[i]$ *is given by*

$$\mathbb{Z}_p[i] = \{a + bi : a, b \in \mathbb{Z}_p, i = \sqrt{-1}\},$$

*where* $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$ *is the set of integers modulo* $p$*.*

For the remainder of this paper, we will consider $\mathbb{Z}_p[i]$ with $p$ a prime congruent to 3 modulo 4 in order to ensure that $i = \sqrt{-1} \notin \mathbb{Z}_p$, which is true by the law of quadratic reciprocity.

This then brings us to the definition of the finite field numerical range given in [5].

**Definition 1.3.** *Let* $p$ *denote a prime congruent to 3 modulo 4 and let* $M \in M_n(\mathbb{Z}_p[i])$*. We define the* finite field numerical range *of* $M$ *to be*

$$W(M) = \{\boldsymbol{x}^* M \boldsymbol{x} : \boldsymbol{x} \in \mathbb{Z}_p[i]^n, \boldsymbol{x}^* \boldsymbol{x} = 1\},$$

*where* $\boldsymbol{x}^*$ *is the conjugate transpose of* $\boldsymbol{x}$*.*

As demonstrated by Ballico [8], this definition for the finite field numerical range can be extended to the field $\mathbb{F}_{q^2}$ (where $q = p^s$ for some prime $p$ and positive integer $s$), which is the degree two Galois extension of $\mathbb{F}_q$, the finite field with $q$ elements. $\mathbb{Z}_p[i]$ is a degree two Galois extension of $\mathbb{Z}_p$ for $p \equiv 3 \bmod 4$.

## 1.2 Self-Orthogonality

To understand self-orthogonal vectors, we must recall the following definition of a norm in a vector space.

**Definition 1.4.** *A* norm *is a scalar-valued function from a vector space to the real numbers such that for all vectors* $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *and any scalar* $\alpha$*, the following properties hold:*

- *Positive-definiteness:* $\|\boldsymbol{x}\| \geq 0$ *and* $\|\boldsymbol{x}\| = 0$ *if and only if* $\boldsymbol{x} = \boldsymbol{0}$*,*
- *Triangle inequality:* $\|\boldsymbol{x} + \boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$
- *Homogeneity:* $\|\alpha \boldsymbol{x}\| = |\alpha| \|\boldsymbol{x}\|$

For a vector $\boldsymbol{x}$ over the complex numbers, we compute the norm of $\boldsymbol{x}$ by

$$\|\boldsymbol{x}\| = \sqrt{\boldsymbol{x}^* \boldsymbol{x}}.$$

Note that this function fulfills all of the properties listed in definition 1.4. However, the above formulation fails to extend to vectors over finite fields, as seen in the following example.

**Example 1.5.** *Let* $\boldsymbol{x} = \begin{bmatrix} i \\ 2 + 3i \end{bmatrix}$ *be a vector in* $\mathbb{Z}_7[i]^2$*. Though* $\boldsymbol{x} \neq \boldsymbol{0}$*, we note that*

$$\boldsymbol{x}^* \boldsymbol{x} = 1 + 4 + 9 \equiv 0 \bmod 7.$$

This leads us to the following definition.

**Definition 1.6.** *A vector* $\boldsymbol{x} \in \mathbb{F}_{q^2}^n$ *is* self-orthogonal *if* $\boldsymbol{x}^*\boldsymbol{x} = 0$ *and* $\boldsymbol{x} \neq \mathbf{0}$.

Because of the existence of self-orthogonal vectors, the function $\|x\| = \sqrt{x^*x}$ fails to be positive-definite over finite fields and is thus not a norm. However, we may still take advantage of the useful properties of $\boldsymbol{x}^*\boldsymbol{x}$ in finite fields, as in the definition of $W(M)$ which uses the set of $\boldsymbol{x}$ such that $\boldsymbol{x}^*\boldsymbol{x} = 1$.

## 1.3 Eigenvalues & Eigenvectors

To understand matrices with self-orthogonal eigenvectors and the challenge they pose in our research of finite field numerical ranges, we must recall the following definitions.

**Definition 1.7.** *A scalar* $\lambda$ *is an* eigenvalue *of matrix* $M$ *if there exists some vector* $\boldsymbol{x}$ *such that* $M\boldsymbol{x} = \lambda\boldsymbol{x}$. *The corresponding vector* $\boldsymbol{x}$ *is called an eigenvector of* $M$.

**Definition 1.8.** *We define the* spectrum *of an* $n \times n$ *matrix* $M$ *to be the set*

$$\sigma(M) = \{\lambda_1, \lambda_2, \ldots, \lambda_k\},$$

*where* $\lambda_i$ *are the eigenvalues of* $M$ *and* $1 \leq k \leq n$.

In examining the spectra of matrices over $\mathbb{F}_{q^2}$, Basha showed that the existence of nonzero eigenvectors $\boldsymbol{x}$ for every eigenvalue ensures that the spectrum of a matrix is a subset of its numerical range, so long as $\boldsymbol{x}$ is not self-orthogonal [7]. This result is formalized in the following theorem.

**Theorem 1.9.** *Let* $M \in M_n(\mathbb{F}_{q^2})$ *with* $\sigma(M) = \{\lambda_1, \lambda_2, \ldots, \lambda_k\}$. *If for all* $\lambda_i \in \sigma(M)$ *there exists a nonzero eigenvector* $\boldsymbol{x_i}$ *such that* $\boldsymbol{x_i}^*\boldsymbol{x_i} \neq 0$, *then* $\sigma(M) \subseteq W(M)$.

However, Basha noted that in finite fields it is possible for a matrix to have self-orthogonal eigenvectors. In these cases, the spectrum will not necessarily be a subset of the numerical range.

**Example 1.10.** *Let* $M$ *be the following matrix in* $M_2(\mathbb{Z}_7[i])$:

$$M = \begin{bmatrix} 6+6i & 4+i \\ 3+6i & 3+4i \end{bmatrix}.$$

$M$ *has eigenvalues* $6+i$ *and* $3+2i$, *corresponding to eigenvectors*

$$\begin{bmatrix} i \\ 2+3i \end{bmatrix} \quad \textit{and} \quad \begin{bmatrix} i \\ 2+4i \end{bmatrix},$$

*both of which are self-orthogonal. Computing* $W(M)$, *we find that*

$$W(M) = \{2i, 1+5i, 2+i, 3+4i, 4, 5+3i, 6+6i\}.$$

*Note that our eigenvalues* $6+i, 3+2i \notin W(M)$ *(see Figure 1).*

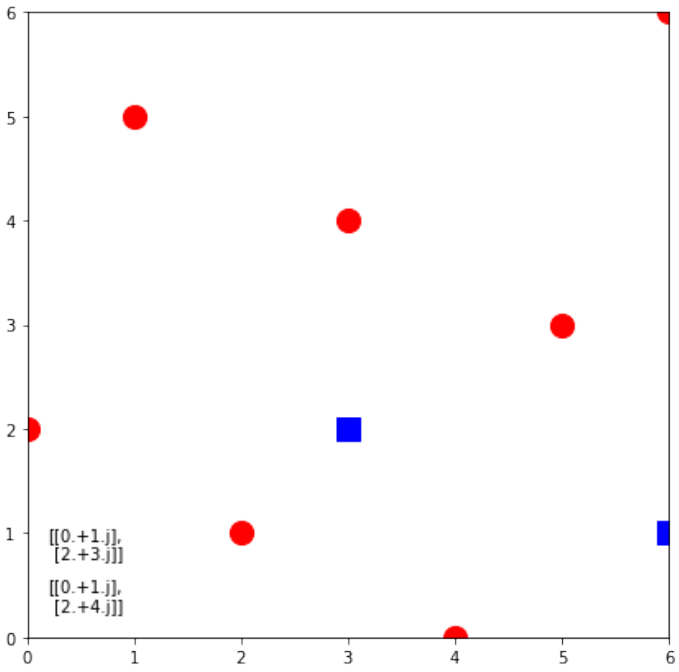


Figure 1: Numerical range (red circles) and eigenvalues (blue squares) for Example 1.10 (eigenvectors listed in the lower left corner of the plot)

To better understand the numerical ranges of matrices with self-orthogonal eigenvectors, it is necessary to construct examples of such matrices. This leads us to the following theorem.

**Theorem 1.11.** *If $M \in M_n(\mathbb{F}_{q^2})$ is a matrix with $n$ linearly independent, self-orthogonal eigenvectors then $M = PDP^{-1}$, where $P$ is a matrix with linearly independent, self-orthogonal columns and $D$ is a diagonal matrix. The eigenvectors of $M$ are given by the columns of $P$, and the spectrum of $M$ is given by the values along the diagonal of $D$.*

*Proof.* Let $M \in M_n(\mathbb{F}_{q^2})$ be a matrix with $n$ linearly independent, self-orthogonal eigenvectors. By [6], the eigenvectors form a self-orthogonal basis for our vector space. The final result follows from [9]. □

The process of constructing a matrix with linearly independent, self-orthogonal eigenvectors is demonstrated in the following example.

**Example 1.12.** *Suppose we wish to construct a matrix $M \in M_2(\mathbb{Z}_7[i])$ with eigenvalues $\lambda_1 = 5i$ and $\lambda_2 = 2 + 5i$ corresponding to the eigenvectors*

$$\boldsymbol{x_1} = \begin{bmatrix} 1+3i \\ 4+3i \end{bmatrix} \quad \text{and} \quad \boldsymbol{x_2} = \begin{bmatrix} 3i \\ 1+2i \end{bmatrix}.$$

*Since $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ are not multiples of each other, they are linearly independent. Furthermore, we observe that*

$$\boldsymbol{x_1}^* \boldsymbol{x_1} = 1 + 9 + 16 + 9 \equiv 0 \bmod 7$$

*and*

$$\boldsymbol{x_2}^*\boldsymbol{x_2} = 9 + 1 + 4 \equiv 0 \bmod 7.$$

*Thus* $\boldsymbol{x_1}$ *and* $\boldsymbol{x_2}$ *are self-orthogonal. Let* $P$ *be the matrix with* $\boldsymbol{x_1}$ *and* $\boldsymbol{x_2}$ *as columns.*

$$P = \begin{bmatrix} 1+3i & 3i \\ 4+3i & 1+2i \end{bmatrix}.$$

*Since* $\boldsymbol{x_1}$ *and* $\boldsymbol{x_2}$ *are linearly independent,* $P$ *is invertible. We can calculate* $P^{-1}$ *by augmenting* $P$ *with the identity matrix and row reducing.*

$$\left[\begin{array}{cc|cc} 1+3i & 3i & 1 & 0 \\ 4+3i & 1+2i & 0 & 1 \end{array}\right] \xrightarrow{R_1 \div (1+3i) \to R_1} \left[\begin{array}{cc|cc} 1 & 3+i & 5+6i & 0 \\ 4+3i & 1+2i & 0 & 1 \end{array}\right]$$
$$\xrightarrow{R_2 - (4+3i)R_1 \to R_2} \left[\begin{array}{cc|cc} 1 & 3+i & 5+6i & 0 \\ 0 & 6+3i & 5+3i & 1 \end{array}\right]$$
$$\xrightarrow{R_2 \div (6+3i) \to R_2} \left[\begin{array}{cc|cc} 1 & 3+i & 5+6i & 0 \\ 0 & 1 & 6+i & 2+6i \end{array}\right]$$
$$\xrightarrow{R_1 - (3+i)R_2 \to R_1} \left[\begin{array}{cc|cc} 1 & 0 & 2+4i & i \\ 0 & 1 & 6+i & 2+6i \end{array}\right].$$

*Therefore*

$$P^{-1} = \begin{bmatrix} 2+4i & i \\ 6+i & 2+6i \end{bmatrix}.$$

*Now let* $D$ *be the diagonal matrix with* $\lambda_1$ *and* $\lambda_2$ *in the diagonal entries.*

$$D = \begin{bmatrix} 5i & 0 \\ 0 & 2+5i \end{bmatrix}.$$

*Calculating* $M = PDP^{-1}$ *yields*

$$M = \begin{bmatrix} 1+6i & 6+5i \\ 1+5i & 1+4i \end{bmatrix}.$$

*We observe that*

$$\begin{aligned} M\boldsymbol{x_1} &= \begin{bmatrix} 1+6i & 6+5i \\ 1+5i & 1+4i \end{bmatrix} \begin{bmatrix} 1+3i \\ 4+3i \end{bmatrix} \\ &= \begin{bmatrix} 6+5i \\ 6+6i \end{bmatrix} \\ &= 5i \begin{bmatrix} 1+3i \\ 4+3i \end{bmatrix} \\ &= \lambda_1 \boldsymbol{x_1} \end{aligned}$$

*and*

$$\begin{aligned} M\boldsymbol{x_2} &= \begin{bmatrix} 1+6i & 6+5i \\ 1+5i & 1+4i \end{bmatrix} \begin{bmatrix} 3i \\ 1+2i \end{bmatrix} \\ &= \begin{bmatrix} 6+6i \\ 6+2i \end{bmatrix} \\ &= (2+5i) \begin{bmatrix} 3i \\ 1+2i \end{bmatrix} \\ &= \lambda_2 \boldsymbol{x_2}. \end{aligned}$$

*Thus, we have verified that $M$ has eigenvalues $\lambda_1$ and $\lambda_2$ corresponding to eigenvectors $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$.*

## 1.4 Unitary Matrices

Unitary matrices are an important class of matrices that interact in useful ways with both the numerical range and self-orthogonal vectors.

**Definition 1.13.** *A matrix $U$ is* unitary *if $U^*U = I$.*

**Definition 1.14.** *Two matrices $A$ and $B$ are* unitarily similar *if there exists a unitary matrix $U$ such that $A = U^*BU$.*

A central property of unitary matrices that we will use in our investigation is the fact that the numerical range is invariant under unitary similarity (Theorem 1.16). The proof of this theorem requires the result of the following lemma.

**Lemma 1.15.** *If $\boldsymbol{x} \in \mathbb{F}_{q^2}^n$ such that $\boldsymbol{x}^*\boldsymbol{x} = 1$ and $U \in M_n(\mathbb{F}_{q^2})$ is unitary, then $(U\boldsymbol{x})^*(U\boldsymbol{x}) = 1$.*

*Proof.* Let $\boldsymbol{x} \in \mathbb{F}_{q^2}^n$ such that $\boldsymbol{x}^*\boldsymbol{x} = 1$ and let $U \in M_n(\mathbb{F}_{q^2})$ be unitary. Since $(AB)^* = B^*A^*$ for any two matrices $A$ and $B$ for which the product $AB$ is defined, we have

$$(U\boldsymbol{x})^*(U\boldsymbol{x}) = \boldsymbol{x}^*U^*U\boldsymbol{x} = \boldsymbol{x}^*\boldsymbol{x} = 1.$$

□

This leads us to the following theorem.

**Theorem 1.16.** *The finite field numerical range is invariant under unitary similarity; that is, for $M, U \in M_n(\mathbb{F}_{q^2})$ with unitary $U$,*

$$W(M) = W(U^*MU).$$

*Proof.* Let $M \in M_n(\mathbb{F}_{q^2})$ and let $U \in M_n(\mathbb{F}_{q^2})$ be unitary. Let $a \in W(U^*MU)$. There exists $\boldsymbol{x} \in \mathbb{F}_{q^2}^n$ with $\boldsymbol{x}^*\boldsymbol{x} = 1$ such that $a = \boldsymbol{x}^*U^*MU\boldsymbol{x} = (U\boldsymbol{x})^*MU\boldsymbol{x}$. Define $\boldsymbol{y} = U\boldsymbol{x}$. By Lemma 1.15, $\boldsymbol{y}^*\boldsymbol{y} = 1$, so $a = \boldsymbol{y}^*M\boldsymbol{y} \in W(M)$. Therefore $W(U^*MU) \subseteq W(M)$. The reverse inclusion follows similarly. Thus $W(M) = W(U^*MU)$. □

The next useful property of unitary matrices follows from the fact that $U^*U = I$.

**Lemma 1.17.** *If $U$ is unitary and $\overline{U}$ is the matrix containing the conjugates of the entries in $U$, then $U^T\overline{U} = I$ and $\overline{U}U^T = I$.*

*Proof.* We note first that, since $(AB)^T = B^T A^T$,

$$U^T \overline{U} = (U^* U)^T = I^T = I.$$

Similarly, we observe that

$$\overline{U} U^T = (U U^*)^T = I^T = I.$$

□

Since our investigation centers on self-orthogonal vectors, it is also helpful to know the following property of multiplication of unitary matrices and self-orthogonal vectors.

**Theorem 1.18.** *If $U \in M_n(\mathbb{F}_{q^2})$ is unitary and $\boldsymbol{v} \in \mathbb{F}_{q^2}^n$ is self-orthogonal, then $U\boldsymbol{v}$ is self-orthogonal.*

*Proof.* Let $\boldsymbol{v} \in \mathbb{F}_{q^2}^n$ be self-orthogonal, so $\boldsymbol{v}^* \boldsymbol{v} = 0$. We then compute

$$\begin{aligned}(U\boldsymbol{v})^*(U\boldsymbol{v}) &= \boldsymbol{v}^* U^* U \boldsymbol{v} \\ &= \boldsymbol{v}^* \boldsymbol{v} \\ &= 0\end{aligned}$$

Therefore $U\boldsymbol{v}$ is self-orthogonal. □

In addition, it is helpful to know properties of multiplication of the transpose of a unitary matrix with a self-orthogonal vector. In Theorem 1.20 we use the form $(\boldsymbol{v}^T U)^T = U^T \boldsymbol{v}$ for a self-orthogonal vector $\boldsymbol{v}$ and unitary matrix $U$, since this is the form we use in later proofs. The proof of Theorem 1.20 relies on the following lemma.

**Lemma 1.19.** *If $\boldsymbol{v} \in \mathbb{F}_{q^2}^n$ is self-orthogonal, then $\boldsymbol{v}^T \overline{\boldsymbol{v}} = 0$.*

*Proof.* Let $\boldsymbol{v} \in \mathbb{F}_{q^2}^n$ be self-orthogonal. We observe that

$$\boldsymbol{v}^T \overline{\boldsymbol{v}} = (\boldsymbol{v}^* \boldsymbol{v})^T = 0^T = 0.$$

□

**Theorem 1.20.** *If $U \in M_n(\mathbb{F}_{q^2})$ is unitary and $\boldsymbol{v} \in \mathbb{F}_{q^2}^n$ is self-orthogonal, then $\left(\boldsymbol{v}^T U\right)^T$ is self-orthogonal.*

*Proof.* Let $U \in M_n(\mathbb{F}_{q^2})$ be unitary and let $\boldsymbol{v} \in \mathbb{F}_{q^2}^n$ be self-orthogonal. Then

$$\begin{aligned}\left(\left(\boldsymbol{v}^T U\right)^T\right)^* \left(\boldsymbol{v}^T U\right)^T &= \overline{\left(\boldsymbol{v}^T U\right)} \left(\boldsymbol{v}^T U\right)^T \\ &= \left(\boldsymbol{v}^* \overline{U}\right) \left(\boldsymbol{v}^T U\right)^T \\ &= \boldsymbol{v}^* \overline{U} U^T \boldsymbol{v} \\ &= \boldsymbol{v}^* \boldsymbol{v} \\ &= 0.\end{aligned}$$

□

### 1.5 Approach

In our investigation, we primarily consider $2 \times 2$ matrices over the field $\mathbb{Z}_7[i]$. We construct matrices with two self-orthogonal eigenvectors and arbitrary eigenvalues via the diagonalization method described in Theorem 1.11, and then we calculate and graph the numerical ranges of these matrices. We examine the structures of these numerical ranges and propose a rule to describe them.

Simultaneous with our investigation, we also develop a set of computational tools to perform calculations and generate examples. These tools are implemented in Python and created in response to challenges that arise in the research process.

## 2 Computational Tools

Our computational tools for this investigation are packaged in the Python modules `linalg_Zpi.py` and `numrange_Zpi.py`, as well as Jupyter notebooks `gen_norm0.ipynb` and `gen_norm1.ipynb`. The `linalg_Zpi` module handles basic linear algebra over finite fields, `numrange_Zpi` allows us to calculate and graph numerical ranges, and `gen_norm0` and `gen_norm1` generate all self-orthogonal and unit vectors (respectively) in a given finite field.

### 2.1 Finite Field Linear Algebra

The `linalg_Zpi` module contains a single class `linalg_Zpi()`, which is initialized with the following parameters.

- `n`: The dimension of the matrices to be handled, and
- `p`: The prime $p$ congruent to 3 modulo 4 that determines the finite field $\mathbb{Z}_p[i]$.

The `linalg_Zpi()` class is designed to be extendable over arbitrarily large matrices over $\mathbb{Z}_p[i]$ for any prime $p$ congruent to 3 modulo 4. However, a vast majority of our research is restricted to the case where `n` $= 2$ and `p` $= 7$, and there may still be unseen limitations to the generalizability of the class methods. The `linalg_Zpi()` class depends on `numpy` and `itertools`. For more details about this module, see selected code listings in Appendix I.

#### 2.1.1 Scalar Division

Division in finite fields tends to be counterintuitive, as illustrated in Example 2.1. Furthermore, the `%` operator in Python, defined such that `a % p` yields the remainder of `a` when divided by `p`, is not sufficient to ensure an output in $\mathbb{Z}_p$ when `a` $\notin \mathbb{Z}_p$, as demonstrated in Example 2.2. Therefore, additional methods must be implemented in `linalg_Zpi()` to handle more complex calculations.

**Example 2.1.** *We know that* $8 \equiv 1 \mod 7$*; therefore* $1/2 \equiv 8/2 \equiv 4 \mod 7$*. Alternatively, we can think of* $1/2$ *as the number that we multiply by 2 to get 1. Since* $4 \cdot 2 = 8 \equiv 1 \mod 7$*, it follows that* $4 \equiv 1/2 \mod 7$*. This means that* $1 \div 2 = 4$ *in* $\mathbb{Z}_7$*.*

**Example 2.2.** *As shown in Example 2.1,* $1 \div 2 = 4$ *in* $\mathbb{Z}_7$*. However, in base Python, we observe the following.*

```
>>> 1/2 % 7
0.5
```

*This creates a problem, since* $0.5 \notin \mathbb{Z}_7$*. Therefore the* `%` *operator alone is insufficient for our purposes in this investigation.*

To ensure that $a \div b$ yields a result in $\mathbb{Z}_p$ for all $a, b \in \mathbb{Z}_p$, we created a "quotient table" that allows us to access the quotient of any two numbers in $\mathbb{Z}_p$. This is stored in a `numpy` array as the class variable `self.div_tbl` (see Listing 2). As an example, we consider the quotient table for $\mathbb{Z}_7$ below.

```
>>> from linalg_Zpi import linalg_Zpi
>>> linalg = linalg_Zpi(n=2, p=7)
>>> linalg.div_tbl
array([[0, 0, 0, 0, 0, 0, 0],
       [0, 1, 2, 3, 4, 5, 6],
       [0, 4, 1, 5, 2, 6, 3],
       [0, 5, 3, 1, 6, 4, 2],
       [0, 2, 4, 6, 1, 3, 5],
       [0, 3, 6, 2, 5, 1, 4],
       [0, 6, 5, 4, 3, 2, 1]])
```

In the quotient table, we indicate the dividend by the row and the divisor by the column. For example, we can compute $1 \div 2$ in $\mathbb{Z}_7$ by accessing the entry at `self.div_tbl[2,1]`, yielding the correct value of 4. Note that each entry in the 0th row of the table indicates the quotient of a number divided by 0. As a result of how the quotient table is generated (see Listing 3 in the appendix), the 0th row is (rather unintuitively) filled with 0s rather than undefined values, but we incorporate additional safeguards to raise an exception in the case of division by 0.

The quotient table is accessed directly via the method `self.div_real()`, which takes in arguments `a` and `b` as the dividend and divisor, respectively. The method `self.div_complex()` computes quotients of complex numbers and calls `self.div_real()`.

Many other methods in the `linalg_Zpi()` class rely on the quotient table, but several of them were implemented before our division methods arrived at their current robust forms, and thus take approaches that work around the need for certain actions. The `self.mult()` method described in the next section is one such example. These methods may later be streamlined to use the current division methods.

### 2.1.2 $\mathbb{Z}_p[i]$ Multiples of Vectors

Alongside performing division in finite fields, it is also extremely important for our generative methods (section 2.2) to be able to determine if two vectors are $\mathbb{Z}_p[i]$ multiples of each other. To this end, we developed the method `self.mult()`, which takes in vectors `v1` and `v2` as arguments. If `v1` and `v2` are $\mathbb{Z}_p[i]$ multiples such that `v1` $= (u + vi)$ `v2` for some $u, v \in \mathbb{Z}_p$, then the method returns the number $u + vi$. If `v1` and `v2` are not $\mathbb{Z}_p[i]$ multiples, then the method returns `None`.

To perform calculations, the `self.mult()` method employs a helper function `solve_mult()`, which calculates the quotient of each pair of corresponding elements in `v1` and `v2`. For a pair of corresponding elements $a + bi$ and $c + di$ from `v1` and `v2` respectively, we seek to find the values $u, v \in \mathbb{Z}_p$ such that

$$\begin{aligned} c + di &= (a + bi)(u + vi) \\ &= (au - bv) + (av + bu)i. \end{aligned}$$

We express this problem as an augmented matrix, which we manipulate to reduced row echelon form via the `solve_mult()` function (see Listing 4).

$$\left[\begin{array}{cc|c} a & -b & c \\ b & a & d \end{array}\right] \xrightarrow{\text{rref}} \left[\begin{array}{cc|c} 1 & 0 & u \\ 0 & 1 & v \end{array}\right]$$

The outputs of `solve_mult()` are checked for each pair of corresponding elements in `v1` and `v2`; if they all match, the value $u + vi$ is returned.

### 2.1.3 Matrix Inverses

The `linalg_Zpi()` also contains a method `self.inv()`, which takes in an invertible matrix `M` and returns its inverse. The inverse of `M` is calculated through a brute-force row reduction process. We augment `M` with the identity matrix and reduce the augmented matrix to reduced row echelon form. While there exist simpler methods to invert a matrix (such as the determinant method described in [9]), at the time of implementing `self.inv()` we had not yet found a proof verifying that such methods were valid for matrices over finite fields.

Our row reduction approach uses nested loops (see Listing 5). At the beginning of the outer loop, we rearrange the rows of our matrix so that the $i$th element of the $i$th row is nonzero; the entire $i$th row is then divided by this $i$th element. Multiples of the $i$th row are then subtracted from each $j$th row for $j > i$ so that the $i$th element of the $i$th row is the last nonzero element in its column. After using this process to arrive at row echelon form, a second loop from the last row of the matrix subtracts multiples of each row $i$ from every row $k$, $k < i$, to finally arrive at reduced row echelon form.

It is important to note that if we try to call `self.inv(M)` for a non-invertible matrix `M`, the program will inevitably encounter a zero-division error and throw an exception.

## 2.2 Generative Methods

To calculate the numerical range of a matrix of a given dimension over a given field, it is necessary to have a list of all vectors $\boldsymbol{x}$ such that $\boldsymbol{x}^*\boldsymbol{x} = 1$ (for the sake of brevity, we will refer to these as unit vectors). Additionally, to construct matrices with self-orthogonal eigenvectors, we generate a list of all self-orthogonal vectors of a given dimension over a given field. For this purpose, we develop the code in the Jupyter notebooks `gen_norm1.ipynb` and `gen_norm0.ipynb`, which handle the exhaustive generation of unit and self-orthogonal vectors, respectively. For more details about these notebooks, see selected code listing in Appendix II.

### 2.2.1 Unit and Self-Orthogonal Vectors

Our code for exhaustively generating unit and self-orthogonal vectors in $\mathbb{Z}_p[i]^n$ is found in the Jupyter notebooks `gen_norm1.ipynb` and `gen_norm0.ipynb`. The two notebooks each contain a single class `gen_norm1()` and `gen_norm0()`, respectively, both of which are initialized with the following parameters.

- `mod`: The prime $p$ congruent to 3 modulo 4 that determines the finite field $\mathbb{Z}_p[i]$ (equivalent to `p` in `linalg_Zpi()`), and
- `size`: The dimension of the vectors to be generated (equivalent to `n` in `linalg_Zpi()`).

The classes both depend on `numpy` and `itertools` imports, as well as the `linalg_Zpi()` module.

Both `gen_norm1()` and `gen_norm0()` generate vectors $\boldsymbol{x}$ in much the same way. Each entry of each vector, being complex, is defined by two values $a, b \in \mathbb{Z}_p$. Thus, each vector can be represented by a sequence of $2n$ values in the form $(a_1, b_1, \ldots, a_n, b_n)$; we generate all such sequences using `itertools`. For each sequence, we calculate the squared sum $a_1^2 + b_1^2 + \ldots + a_n^2 + b_n^2$ modulo $p$, which is equal to $\boldsymbol{x}^*\boldsymbol{x}$. For `gen_norm1()` we keep all sequences with a square sum of 1, and for `gen_norm0()` we keep all sequences with a square sum of 0 (omitting the sequence of all 0s, as this represents the zero vector). All sequences with the correct square sum are rearranged into `numpy` vectors.

When generating matrices with self-orthogonal eigenvectors, we only consider eigenvectors that are linearly independent in order to use the diagonalization method described in Theorem 1.11. We therefore employ the `linalg_Zpi.mult()` method to remove any $\mathbb{Z}_p[i]$ multiples present in the list generated from `gen_norm0()`. Similarly with our unit vectors from `gen_norm1()`, the existence of multiple units in a finite field may result in unit multiples of a vector that can be removed from consideration.

For each class, we export the final list of vectors as a `.npz` file. These files are referenced whenever we require unit or self-orthogonal vectors of a given dimension over a given field.

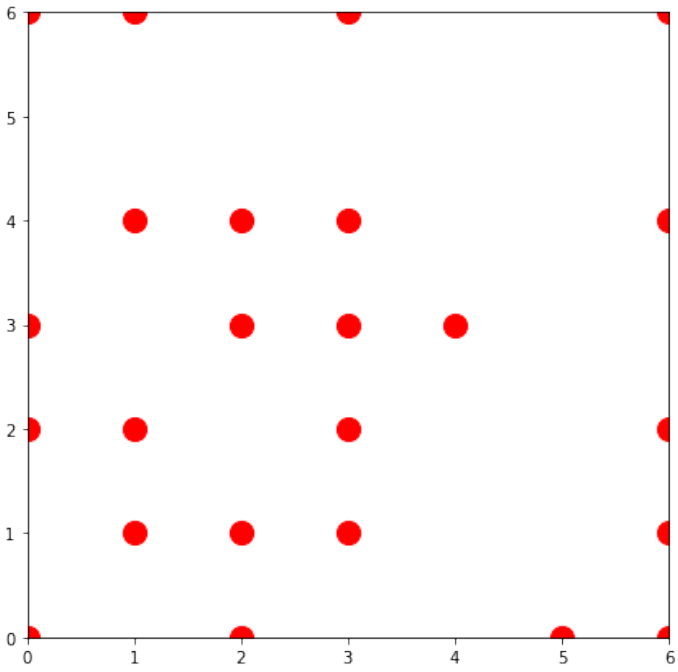


Figure 2: The plotted output of `numrange_Zpi.W1()` for a random matrix over $\mathbb{Z}_7[i]$.

### 2.3 Numerical Range Solver

Our tool for calculating and plotting the numerical range of a matrix can be found in the module `numrange_Zpi.py`. This module contains a single class `numrange_Zpi()`, which takes in the following parameters.

- `n`: The dimension of the matrix to be handled,
- `p`: The prime $p$ congruent to 3 modulo 4 that determines the finite field $\mathbb{Z}_p[i]$, and
- `M`: The matrix for which to compute the numerical range.

The `numrange_Zpi()` class contains the method `self.W1()`. This method `self.W1()` computes the finite field numerical range of `M`, referencing the relevant set of unit vectors generated by `gen_norm1()`. The method takes in a single Boolean parameter `plot`; if `plot` is set to `True`, the program will generate a plot of the numerical range. For more details about this module, see selected code listing in Appendix III.

# 3 Investigating $2 \times 2$ Matrices Over $\mathbb{Z}_7[i]$

## 3.1 Preliminary Observations

Using our computational modules, we generate pseudo-random combinations of two eigenvalues and two linearly independent, self-orthogonal eigenvectors and construct matrices from them using the diagonalization method in Theorem 1.11. We proceed to plot the numerical range of our random matrices and look for patterns. We make the key observation, evident in Figure 3, that the eigenvalues of our matrices are not contained in the numerical ranges for any of the generated examples. This leads us to the following conjecture.

**Conjecture 3.1.** *If $M \in M_2(\mathbb{Z}_7[i])$ has unique eigenvalues $\lambda_1$ and $\lambda_2$ corresponding to two linearly independent, self-orthogonal eigenvectors, then $\lambda_1, \lambda_2 \notin W(M)$.*

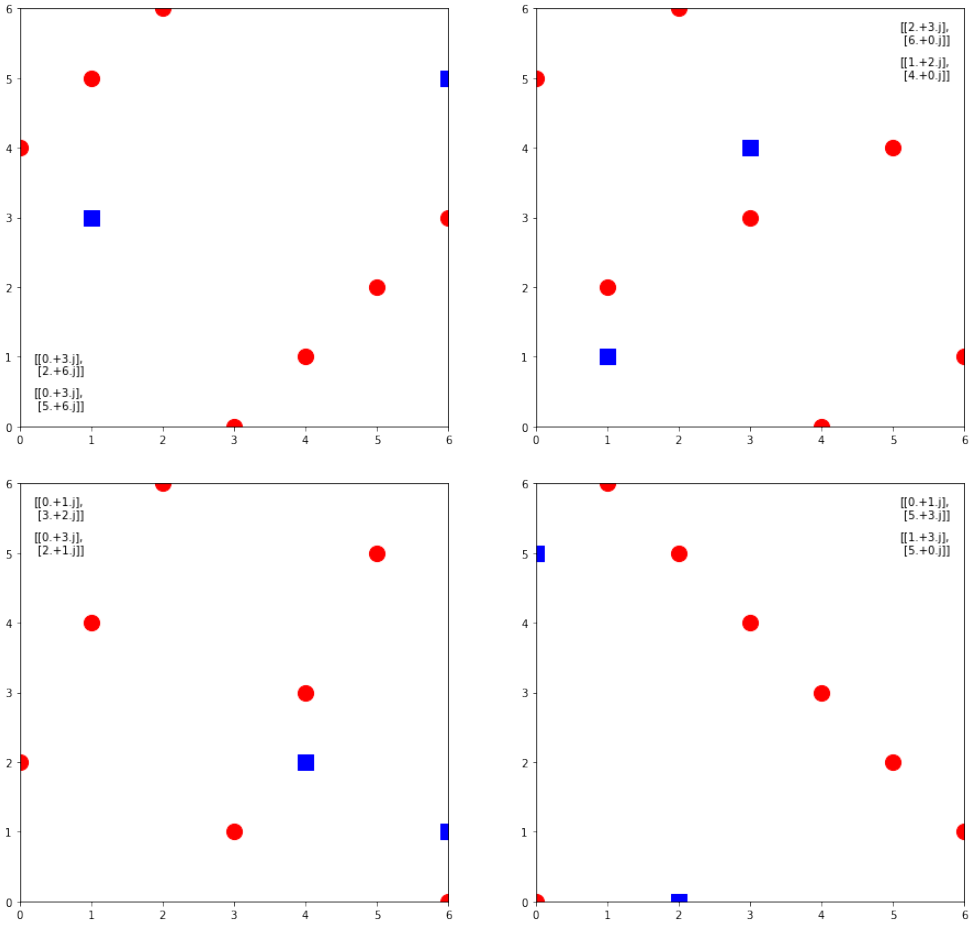


Figure 3: Numerical ranges (red circles) and eigenvalues (blue squares) for randomly-generated matrices with self-orthogonal eigenvectors. Eigenvectors are listed in the corners of each subplot.

For the sake of brevity, we introduce the term Lund matrices to refer to the class of matrices considered in Conjecture 3.1.

**Definition 3.2.** *A matrix $M \in M_2(\mathbb{Z}_7[i])$ is a* Lund matrix *if it has two unique eigenvalues corresponding to two linearly independent, self-orthogonal eigenvectors.*

In Conjecture 3.1, we do not consider the the trivial case where our matrix has one eigenvalue of multiplicity two, since this is equivalent to a multiple of the identity matrix. Coons et al. demonstrated that $W(\lambda I) = \{\lambda\}$ for some $\lambda \in \mathbb{Z}_7[i]$, and we prove the same result using our diagonalization method in the following theorem.

**Theorem 3.3.** *If $M = \lambda I \in M_2(\mathbb{Z}_7[i])$ for some $\lambda \in \mathbb{Z}_7[i]$, then $W(M) = \{\lambda\}$.*

*Proof.* Let $M = \lambda I \in M_2(\mathbb{Z}_7[i])$ for some $\lambda \in \mathbb{Z}_7[i]$. Since all vectors in $\mathbb{Z}_7[i]^2$ are eigenvectors of $M$, we pick two linearly independent, self-orthogonal vectors $\boldsymbol{z_1}$ and $\boldsymbol{z_2}$ to be eigenvectors. By Theorem 1.11,

$$M = \begin{bmatrix} \boldsymbol{z_1} & \boldsymbol{z_2} \end{bmatrix} \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \begin{bmatrix} \boldsymbol{z_1} & \boldsymbol{z_2} \end{bmatrix}^{-1}.$$

Each element of $W(M)$ can be expressed as $\boldsymbol{x}^* M \boldsymbol{x}$ for some unit vector $\boldsymbol{x}$. We observe that

$$\begin{aligned} \boldsymbol{x}^* M \boldsymbol{x} &= \boldsymbol{x}^* \begin{bmatrix} \boldsymbol{z_1} & \boldsymbol{z_2} \end{bmatrix} \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \begin{bmatrix} \boldsymbol{z_1} & \boldsymbol{z_2} \end{bmatrix}^{-1} \boldsymbol{x} \\ &= \lambda \boldsymbol{x}^* \begin{bmatrix} \boldsymbol{z_1} & \boldsymbol{z_2} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{z_1} & \boldsymbol{z_2} \end{bmatrix}^{-1} \boldsymbol{x} \\ &= \lambda \boldsymbol{x}^* \begin{bmatrix} \boldsymbol{z_1} & \boldsymbol{z_2} \end{bmatrix} \begin{bmatrix} \boldsymbol{z_1} & \boldsymbol{z_2} \end{bmatrix}^{-1} \boldsymbol{x} \\ &= \lambda \boldsymbol{x}^* \boldsymbol{x} \\ &= \lambda. \end{aligned}$$

Therefore $W(M) = \{\lambda\}$. □

Another interesting observation we see in the examples of Figure 4 is that Lund matrices with the same eigenvalues but different eigenvectors appear to have the same numerical range regardless of the specific eigenvectors.

**Conjecture 3.4.** *Let $A, B \in M_2(\mathbb{Z}_7[i])$ be Lund matrices that share the eigenvalues $\lambda_1$ and $\lambda_2$. Let $A$ have eigenvectors $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$, and let $B$ have eigenvectors $\boldsymbol{y_1}$ and $\boldsymbol{y_2}$. Then $W(A) = W(B)$.*

## 3.2 Structure of the Numerical Range

Examining our randomly-generated examples, we observe that the numerical range in each example is, in fact, a line in $\mathbb{Z}_7[i]$.

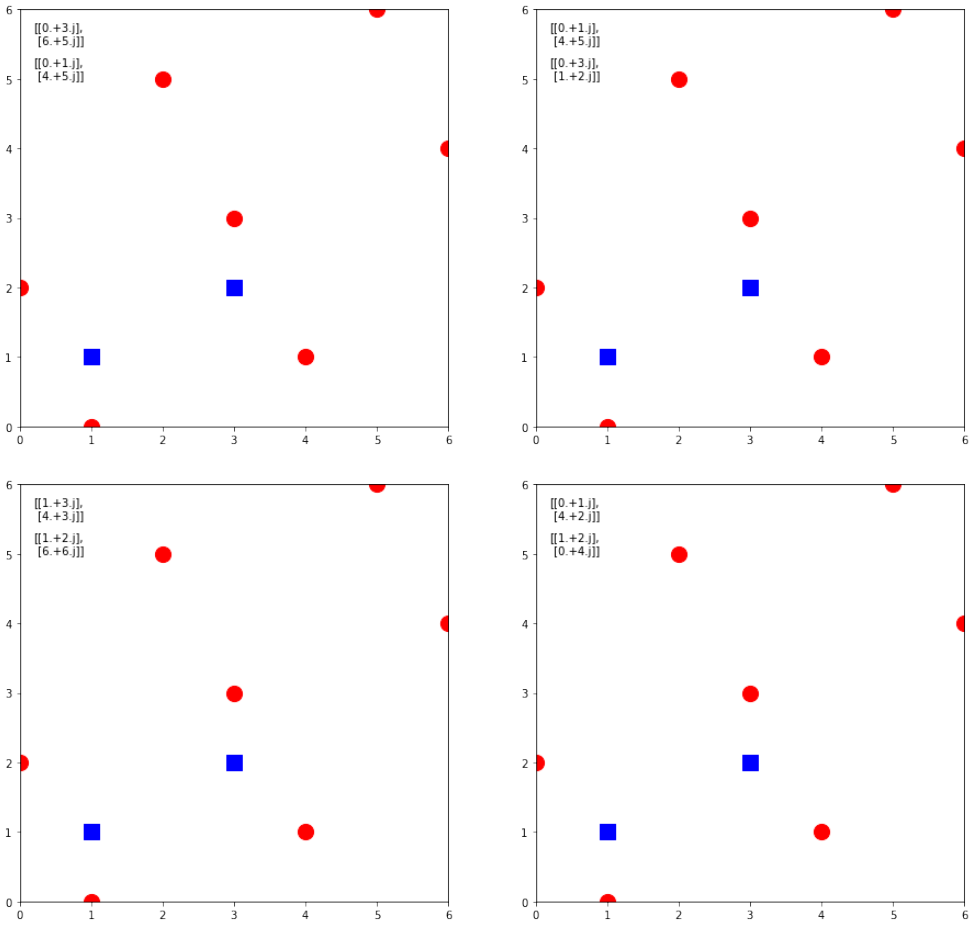


Figure 4: Numerical ranges for matrices with the same eigenvalues and different self-orthogonal eigenvectors.

**Example 3.5.** *Consider the upper-left subplot in Figure 4, which shows the numerical range for the matrix*

$$M = \begin{bmatrix} 5+6i & 6i \\ i & 4+5i \end{bmatrix}.$$

*The set of points in the numerical range is*

$$W(M) = \{2i, 1, 2+5i, 3+3i, 4+i, 5+6i, 6+4i\}.$$

*Taking any pair of these points, we can calculate the slope of this line in $\mathbb{Z}_7$. For example, taking $2i$ and $1$:*

$$\frac{0-2}{1-0} \equiv -2 \equiv 5 \bmod 7;$$

*taking $2i$ and $2+5i$:*

$$\frac{5-2}{2-0} \equiv \frac{3}{2} \equiv \frac{10}{2} \equiv 5 \bmod 7;$$

*and taking $5+6i$ and $3+3i$:*

$$\frac{3-6}{3-5} \equiv \frac{3}{2} \equiv 5 \bmod 7.$$

*It therefore becomes evident that the slope of this line in $\mathbb{Z}_7$ is 5. Since $2i$ is the only point in the numerical range that lies on the imaginary axis, the $y$-intercept of this line is 2. Therefore, for any point $a+bi \in W(M)$,*

$$b = 5a + 2.$$

Comparing the slope of the numerical range to the slope of the line determined by our eigenvectors further suggests that the two lines are perpendicular.

**Example 3.6.** *We again consider the upper-left subplot in Figure 4, which plots the numerical range of the matrix $M$ in Example 3.5 as well as its eigenvalues $1+i$ and $3+2i$. Recall that the slope of $W(M)$ was determined to be 5. Note that the slope of the line determined by $1+i$ and $3+2i$ is*

$$\frac{2-1}{3-1} \equiv \frac{1}{2} \equiv 4 \bmod 7.$$

*Finally note that 5 and 4 are negative reciprocals in $\mathbb{Z}_7$:*

$$\frac{-1}{5} \equiv \frac{6}{12} \equiv \frac{1}{2} \equiv 4 \bmod 7.$$

*Therefore the line of $W(M)$ is perpendicular to the line determined by the eigenvalues of $M$.*

As it appears that the eigenvalues of our matrix determine the slope of its numerical range, this naturally leads us to wonder whether the eigenvalues also determine a single point in the numerical range, which together with the slope will determine a line. We conjecture that the line of the numerical range intersects the midpoint of our eigenvalues.

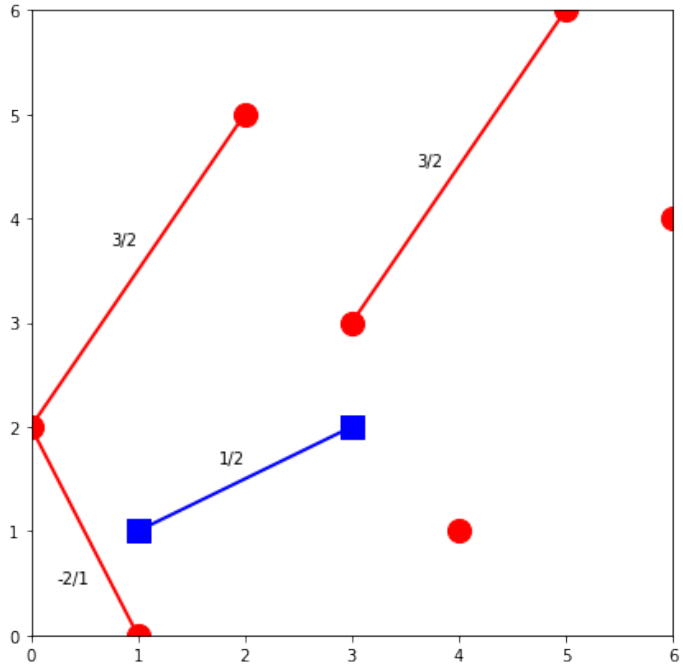


Figure 5: Calculating slopes in Examples 3.5 and 3.6.

**Example 3.7.** *Once again, consider the upper-left subplot in Figure 4, which plots the numerical range of the matrix $M$ in Example 3.5 as well as its eigenvalues $1+i$ and $3+2i$. First, we can calculate the midpoint of $1+i$ and $3+2i$ as*

$$\frac{(1+i)+(3+2i)}{2} = \frac{4+3i}{2} = 2+\frac{3}{2}i \equiv 2+5i \bmod 7.$$

*Let $m$ denote the line of $W(M)$; recall that this line is given by $b = 5a+2$. Now let $\ell$ denote the line determined by our eigenvalues, which we can represent in point-slope form by $(b-1) = 4(a-1)$; this simplifies to $b = 4a+4$.*

*The values for $a$ and $b$ that satisfy the equations of both $m$ and $\ell$ are $a = 2$ and $b = 5$. Therefore the point of intersection between $m$ and $\ell$ is $2+5i$, which is the midpoint of our eigenvalues.*

Figure 6 shows four other randomly chosen examples. In each, the point of intersection between lines $m$ and $\ell$ is clearly incident with the midpoint of the eigenvalues. This, along with significantly more examples and an inability to find any counterexamples, leads us to the following conjecture regarding the structure of the numerical range for Lund matrices.

**Conjecture 3.8.** *If $M \in M_2(\mathbb{Z}_7[i])$ is a Lund matrix with eigenvalues $\lambda_1$ and $\lambda_2$, then $W(M)$ is the line perpendicular to $\overleftrightarrow{\lambda_1\lambda_2}$ that passes through $\frac{\lambda_1+\lambda_2}{2}$.*

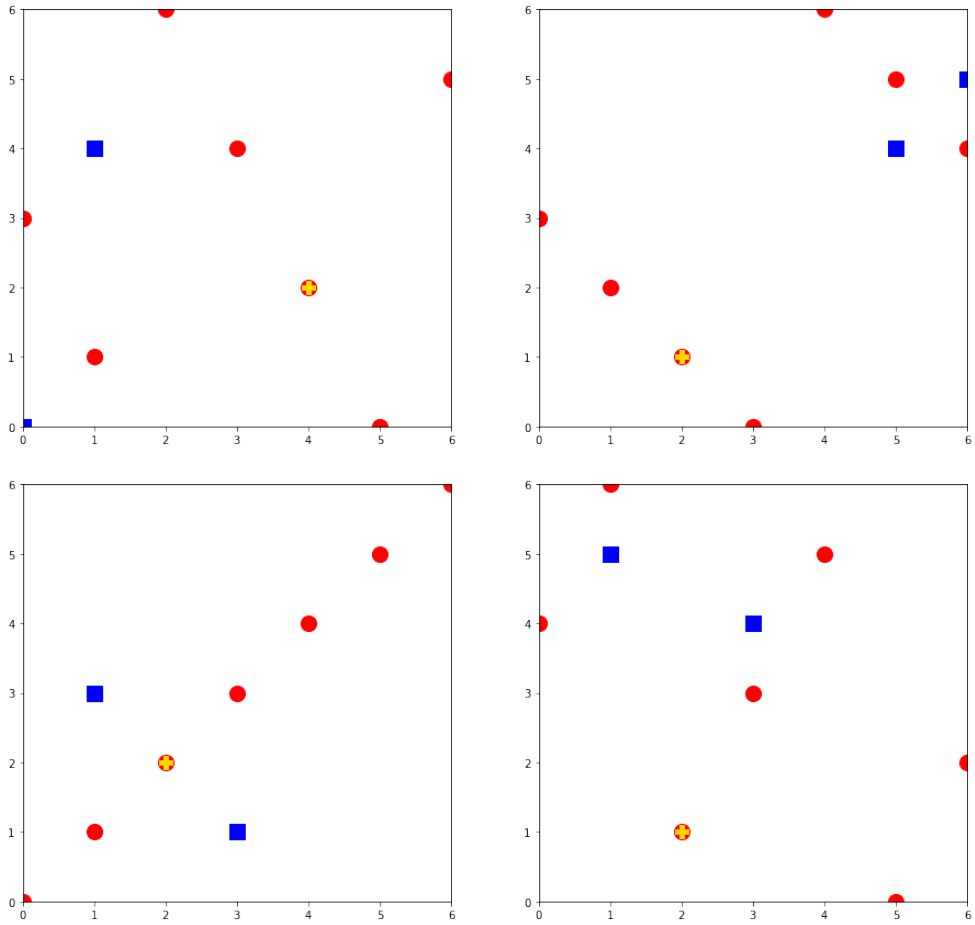

Figure 6: The midpoints (yellow crosses) of randomly-selected eigenvalues intersect with the numerical range of Lund matrices with those eigenvalues.

## 3.3 Approaches

Being able to prove Conjecture 3.8 is central to our goal of understanding the structure of the numerical range for matrices with self-orthogonal eigenvectors. However, a rigorous proof of this conjecture has been elusive. We consider two different approaches, namely analyzing the algebraic definition of the numerical range and investigating unitary similarity of matrices with self-orthogonal vectors.

### 3.3.1 Algebra of the Numerical Range

We investigate the algebraic definition of the numerical range for a Lund matrix $M \in M_2(\mathbb{Z}_7[i])$ with eigenvalues $\lambda_1, \lambda_2 \in \mathbb{Z}_7[i]$ corresponding to eigenvectors $\boldsymbol{z_1}, \boldsymbol{z_2} \in \mathbb{Z}_7[i]^2$. Using the diagonalization method, we begin by expressing an arbitrary element of $a \in W(M)$ in the form

$$a = \boldsymbol{x}^* \begin{bmatrix} \boldsymbol{z_1} & \boldsymbol{z_2} \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{z_1} & \boldsymbol{z_2} \end{bmatrix}^{-1} \boldsymbol{x},$$

where $\boldsymbol{x}$ is a unit vector. Clearly, the matrix

$$\begin{bmatrix} \boldsymbol{z_1} & \boldsymbol{z_2} \end{bmatrix}^{-1}$$

poses some challenges to algebraically manipulating the above expression, so we employ the result of the following lemma.

**Lemma 3.9.** *If $\boldsymbol{z_1}, \boldsymbol{z_2} \in \mathbb{Z}_p[i]^2$ are linearly-independent self-orthogonal vectors and*

$$\begin{bmatrix} \boldsymbol{a_1} & \boldsymbol{a_2} \end{bmatrix} = \begin{bmatrix} \boldsymbol{z_1}^T \\ \boldsymbol{z_2}^T \end{bmatrix}^{-1},$$

*then $\boldsymbol{a_1}$ and $\boldsymbol{a_2}$ are self-orthogonal.*

*Proof.* Let $\boldsymbol{z_1}, \boldsymbol{z_2} \in \mathbb{Z}_p[i]^2$ be linearly-independent self-orthogonal vectors and let

$$\begin{bmatrix} \boldsymbol{a_1} & \boldsymbol{a_2} \end{bmatrix} = \begin{bmatrix} \boldsymbol{z_1}^T \\ \boldsymbol{z_2}^T \end{bmatrix}^{-1}.$$

Since $\boldsymbol{z_1}$ and $\boldsymbol{z_2}$ are linearly independent, they form a self-orthogonal basis for $\mathbb{Z}_p[i]^2$ by [6]. Therefore the matrix

$$\begin{bmatrix} \boldsymbol{z_1}^T \\ \boldsymbol{z_2}^T \end{bmatrix}$$

is invertible. Thus there exist linearly independent vectors $\boldsymbol{a_1}, \boldsymbol{a_2} \in \mathbb{Z}_p[i]^2$ such that

$$\begin{bmatrix} \boldsymbol{a_1} & \boldsymbol{a_2} \end{bmatrix} = \begin{bmatrix} \boldsymbol{z_1}^T \\ \boldsymbol{z_2}^T \end{bmatrix}^{-1}.$$

We observe that

$$\begin{aligned}\begin{bmatrix}\boldsymbol{a_1}^*\boldsymbol{a_1} & \boldsymbol{a_1}^*\boldsymbol{a_2}\\ \boldsymbol{a_2}^*\boldsymbol{a_1} & \boldsymbol{a_2}^*\boldsymbol{a_2}\end{bmatrix} &= \begin{bmatrix}\boldsymbol{a_1}^*\\ \boldsymbol{a_2}^*\end{bmatrix}\begin{bmatrix}\boldsymbol{a_1} & \boldsymbol{a_2}\end{bmatrix}\\ &= \begin{bmatrix}\boldsymbol{a_1} & \boldsymbol{a_2}\end{bmatrix}^*\begin{bmatrix}\boldsymbol{a_1} & \boldsymbol{a_2}\end{bmatrix}\\ &= \left(\begin{bmatrix}\boldsymbol{z_1}^T\\ \boldsymbol{z_2}^T\end{bmatrix}^{-1}\right)^*\begin{bmatrix}\boldsymbol{z_1}^T\\ \boldsymbol{z_2}^T\end{bmatrix}^{-1}.\end{aligned}$$

Since $(A^*)^{-1} = \left(A^{-1}\right)^*$, we have

$$\begin{aligned}\left(\begin{bmatrix}\boldsymbol{z_1}^T\\ \boldsymbol{z_2}^T\end{bmatrix}^{-1}\right)^*\begin{bmatrix}\boldsymbol{z_1}^T\\ \boldsymbol{z_2}^T\end{bmatrix}^{-1} &= \left(\begin{bmatrix}\boldsymbol{z_1}^T\\ \boldsymbol{z_2}^T\end{bmatrix}^*\right)^{-1}\begin{bmatrix}\boldsymbol{z_1}^T\\ \boldsymbol{z_2}^T\end{bmatrix}^{-1}\\ &= \left(\begin{bmatrix}\boldsymbol{z_1}^T\\ \boldsymbol{z_2}^T\end{bmatrix}\begin{bmatrix}\boldsymbol{z_1}^T\\ \boldsymbol{z_2}^T\end{bmatrix}^*\right)^{-1}\\ &= \left(\begin{bmatrix}\boldsymbol{z_1}^T\\ \boldsymbol{z_2}^T\end{bmatrix}\begin{bmatrix}\overline{\boldsymbol{z_1}} & \overline{\boldsymbol{z_2}}\end{bmatrix}\right)^{-1}\\ &= \begin{bmatrix}\boldsymbol{z_1}^T\overline{\boldsymbol{z_1}} & \boldsymbol{z_1}^T\overline{\boldsymbol{z_2}}\\ \boldsymbol{z_2}^T\overline{\boldsymbol{z_1}} & \boldsymbol{z_2}^T\overline{\boldsymbol{z_2}}\end{bmatrix}^{-1}\\ &= \begin{bmatrix}0 & \boldsymbol{z_1}^T\overline{\boldsymbol{z_2}}\\ \boldsymbol{z_2}^T\overline{\boldsymbol{z_1}} & 0\end{bmatrix}^{-1}.\end{aligned}$$

Note that since $\boldsymbol{z_1}$ and $\boldsymbol{z_2}$ form a self-orthogonal basis for $\mathbb{Z}_p[i]^2$, they must not be orthogonal [6]. Therefore $\boldsymbol{z_1}^T\overline{\boldsymbol{z_2}} \neq 0$ and $\boldsymbol{z_2}^T\overline{\boldsymbol{z_1}} \neq 0$, so the matrix

$$\begin{bmatrix}0 & \boldsymbol{z_1}^T\overline{\boldsymbol{z_2}}\\ \boldsymbol{z_2}^T\overline{\boldsymbol{z_1}} & 0\end{bmatrix}$$

is invertible. By [9] we have

$$\begin{aligned}\begin{bmatrix}0 & \boldsymbol{z_1}^T\overline{\boldsymbol{z_2}}\\ \boldsymbol{z_2}^T\overline{\boldsymbol{z_1}} & 0\end{bmatrix}^{-1} &= \begin{bmatrix}0 & -\boldsymbol{z_1}^T\overline{\boldsymbol{z_2}}\\ -\boldsymbol{z_2}^T\overline{\boldsymbol{z_1}} & 0\end{bmatrix}\frac{-1}{\boldsymbol{z_1}^T\overline{\boldsymbol{z_2}}\boldsymbol{z_2}^T\overline{\boldsymbol{z_1}}}\\ &= \begin{bmatrix}0 & \frac{1}{\boldsymbol{z_2}^T\overline{\boldsymbol{z_1}}}\\ \frac{1}{\boldsymbol{z_1}^T\overline{\boldsymbol{z_2}}} & 0\end{bmatrix}.\end{aligned}$$

It therefore follows that

$$\begin{bmatrix}\boldsymbol{a_1}^*\boldsymbol{a_1} & \boldsymbol{a_1}^*\boldsymbol{a_2}\\ \boldsymbol{a_2}^*\boldsymbol{a_1} & \boldsymbol{a_2}^*\boldsymbol{a_2}\end{bmatrix} = \begin{bmatrix}0 & \frac{1}{\boldsymbol{z_2}^T\overline{\boldsymbol{z_1}}}\\ \frac{1}{\boldsymbol{z_1}^T\overline{\boldsymbol{z_2}}} & 0\end{bmatrix},$$

so $\boldsymbol{a_1}^*\boldsymbol{a_1} = 0$ and $\boldsymbol{a_2}^*\boldsymbol{a_2} = 0$. Thus $\boldsymbol{a_1}$ and $\boldsymbol{a_2}$ are self-orthogonal. □

Additionally, we are able to derive an expression for the matrix $\begin{bmatrix}\boldsymbol{z_1} & \boldsymbol{z_2}\end{bmatrix}^{-1}$ in terms of the vectors $\boldsymbol{z_1}$ and $\boldsymbol{z_2}$.

**Lemma 3.10.** *If $\boldsymbol{z_1}, \boldsymbol{z_2} \in \mathbb{Z}_p[i]^2$ are linearly-independent self-orthogonal vectors, then*

$$\begin{bmatrix}\boldsymbol{z_1} & \boldsymbol{z_2}\end{bmatrix}^{-1} = \begin{bmatrix}\frac{1}{\boldsymbol{z_2}^*\boldsymbol{z_1}} & 0\\ 0 & \frac{1}{\boldsymbol{z_1}^*\boldsymbol{z_2}}\end{bmatrix}\begin{bmatrix}\boldsymbol{z_2}^*\\ \boldsymbol{z_1}^*\end{bmatrix}.$$

*Proof.* Let $\boldsymbol{z_1}, \boldsymbol{z_2} \in \mathbb{Z}_p[i]^2$ be linearly-independent self-orthogonal vectors. We observe that

$$\begin{aligned}
\begin{bmatrix} \frac{1}{\boldsymbol{z_2}^* \boldsymbol{z_1}} & 0 \\ 0 & \frac{1}{\boldsymbol{z_1}^* \boldsymbol{z_2}} \end{bmatrix} \begin{bmatrix} \boldsymbol{z_2}^* \\ \boldsymbol{z_1}^* \end{bmatrix} \begin{bmatrix} \boldsymbol{z_1} & \boldsymbol{z_2} \end{bmatrix} &= \begin{bmatrix} \frac{1}{\boldsymbol{z_2}^* \boldsymbol{z_1}} & 0 \\ 0 & \frac{1}{\boldsymbol{z_1}^* \boldsymbol{z_2}} \end{bmatrix} \begin{bmatrix} \boldsymbol{z_2}^* \boldsymbol{z_1} & \boldsymbol{z_2}^* \boldsymbol{z_2} \\ \boldsymbol{z_1}^* \boldsymbol{z_1} & \boldsymbol{z_1}^* \boldsymbol{z_2} \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{\boldsymbol{z_2}^* \boldsymbol{z_1}} & 0 \\ 0 & \frac{1}{\boldsymbol{z_1}^* \boldsymbol{z_2}} \end{bmatrix} \begin{bmatrix} \boldsymbol{z_2}^* \boldsymbol{z_1} & 0 \\ 0 & \boldsymbol{z_1}^* \boldsymbol{z_2} \end{bmatrix} \\
&= \begin{bmatrix} \frac{\boldsymbol{z_2}^* \boldsymbol{z_1}}{\boldsymbol{z_2}^* \boldsymbol{z_1}} & 0 \\ 0 & \frac{\boldsymbol{z_1}^* \boldsymbol{z_2}}{\boldsymbol{z_1}^* \boldsymbol{z_2}} \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.
\end{aligned}$$

Since

$$\begin{bmatrix} \frac{1}{\boldsymbol{z_2}^* \boldsymbol{z_1}} & 0 \\ 0 & \frac{1}{\boldsymbol{z_1}^* \boldsymbol{z_2}} \end{bmatrix} \begin{bmatrix} \boldsymbol{z_2}^* \\ \boldsymbol{z_1}^* \end{bmatrix}$$

is a left inverse for $\begin{bmatrix} \boldsymbol{z_1} & \boldsymbol{z_2} \end{bmatrix}$, it follows by a corollary to the Invertible Matrix Theorem that it must also be a right inverse and therefore the inverse of $\begin{bmatrix} \boldsymbol{z_1} & \boldsymbol{z_2} \end{bmatrix}$. □

Using the results from Lemma 3.10, we can derive an expression for an arbitrary element of $W(M)$.

**Theorem 3.11.** *If $M \in M_2(\mathbb{Z}_7[i])$ is a Lund matrix with eigenvalues $\lambda_1, \lambda_2 \in \mathbb{Z}_7[i]$ corresponding to eigenvectors $\boldsymbol{z_1}, \boldsymbol{z_2} \in \mathbb{Z}_7[i]^2$, then an arbitrary element $a \in W(M)$ can be expressed as*

$$a = \frac{\lambda_1}{\boldsymbol{z_2}^* \boldsymbol{z_1}} \boldsymbol{x}^* \boldsymbol{z_1} \boldsymbol{z_2}^* \boldsymbol{x} + \frac{\lambda_2}{\boldsymbol{z_2}^* \boldsymbol{z_1}} \boldsymbol{x}^* \boldsymbol{z_2} \boldsymbol{z_1}^* \boldsymbol{x},$$

*where $\boldsymbol{x}$ is a unit vector.*

*Proof.* Let $M \in M_2(\mathbb{Z}_7[i])$ be Lund matrix with eigenvalues $\lambda_1, \lambda_2 \in \mathbb{Z}_7[i]$ corresponding to eigenvectors $\boldsymbol{z_1}, \boldsymbol{z_2} \in \mathbb{Z}_7[i]^2$. Using the diagonalization method, we can express an arbitrary element $a \in W(M)$ in the form

$$a = \boldsymbol{x}^* \begin{bmatrix} \boldsymbol{z_1} & \boldsymbol{z_2} \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{z_1} & \boldsymbol{z_2} \end{bmatrix}^{-1} \boldsymbol{x}$$

for some unit vector $\boldsymbol{x}$. By Lemma 3.10, we have

$$\begin{aligned}
\boldsymbol{x}^* \begin{bmatrix} \boldsymbol{z_1} & \boldsymbol{z_2} \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{z_1} & \boldsymbol{z_2} \end{bmatrix}^{-1} \boldsymbol{x} &= \boldsymbol{x}^* \begin{bmatrix} \boldsymbol{z_1} & \boldsymbol{z_2} \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \frac{1}{\boldsymbol{z_2}^* \boldsymbol{z_1}} & 0 \\ 0 & \frac{1}{\boldsymbol{z_1}^* \boldsymbol{z_2}} \end{bmatrix} \begin{bmatrix} \boldsymbol{z_2}^* \\ \boldsymbol{z_1}^* \end{bmatrix} \boldsymbol{x} \\
&= \begin{bmatrix} \boldsymbol{x}^* \boldsymbol{z_1} & \boldsymbol{x}^* \boldsymbol{z_2} \end{bmatrix} \begin{bmatrix} \frac{\lambda_1}{\boldsymbol{z_2}^* \boldsymbol{z_1}} & 0 \\ 0 & \frac{\lambda_2}{\boldsymbol{z_1}^* \boldsymbol{z_2}} \end{bmatrix} \begin{bmatrix} \boldsymbol{z_2}^* \boldsymbol{x} \\ \boldsymbol{z_1}^* \boldsymbol{x} \end{bmatrix} \\
&= \begin{bmatrix} \frac{\lambda_1}{\boldsymbol{z_2}^* \boldsymbol{z_1}} \boldsymbol{x}^* \boldsymbol{z_1} & \frac{\lambda_2}{\boldsymbol{z_1}^* \boldsymbol{z_2}} \boldsymbol{x}^* \boldsymbol{z_2} \end{bmatrix} \begin{bmatrix} \boldsymbol{z_2}^* \boldsymbol{x} \\ \boldsymbol{z_1}^* \boldsymbol{x} \end{bmatrix} \\
&= \frac{\lambda_1}{\boldsymbol{z_2}^* \boldsymbol{z_1}} \boldsymbol{x}^* \boldsymbol{z_1} \boldsymbol{z_2}^* \boldsymbol{x} + \frac{\lambda_2}{\boldsymbol{z_2}^* \boldsymbol{z_1}} \boldsymbol{x}^* \boldsymbol{z_2} \boldsymbol{z_1}^* \boldsymbol{x}.
\end{aligned}$$

□

While this result does not yield a conclusive result for Conjecture 3.8, we hope that it may shed some light on how a proof of the conjecture may be approached.

### 3.3.2 Unitary Similarity of Lund Matrices

We recall the observation in Conjecture 3.4 that Lund matrices with the same eigenvalues have the same numerical range regardless of the eigenvectors. Considering the fact that the finite field numerical range is invariant under unitary similarity, this leads us to conjecture that all Lund matrices with the same eigenvalues are unitarily similar to each other.

**Conjecture 3.12.** *If $M, N \in M_2(\mathbb{Z}_7[i])$ are Lund matrices with the same eigenvalues, then there exists a unitary matrix $U \in M_2(\mathbb{Z}_7[i])$ such that $M = U^*NU$.*

While we have not been able to prove this conjecture, we do have the following interesting result.

**Theorem 3.13.** *If $M \in M_2(\mathbb{Z}_7[i])$ is a Lund matrix and $U \in M_2(\mathbb{Z}_7[i])$ is unitary, then $U^*MU$ is a Lund matrix with the same eigenvalues.*

*Proof.* Let $M \in M_2(\mathbb{Z}_7[i])$ be a Lund matrix and let $U \in M_2(\mathbb{Z}_7[i])$ be unitary. By Lemma 3.10, $M$ can be expressed as

$$M = \begin{bmatrix} \boldsymbol{z_1} & \boldsymbol{z_2} \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \frac{1}{\boldsymbol{z_2}^*\boldsymbol{z_1}} & 0 \\ 0 & \frac{1}{\boldsymbol{z_1}^*\boldsymbol{z_2}} \end{bmatrix} \begin{bmatrix} \boldsymbol{z_2}^* \\ \boldsymbol{z_1}^* \end{bmatrix},$$

where $\lambda_1$ and $\lambda_2$ are the eigenvalues of $M$ corresponding to eigenvectors $\boldsymbol{z_1}$ and $\boldsymbol{z_2}$. Thus we have

$$\begin{aligned} U^*MU &= U^* \begin{bmatrix} \boldsymbol{z_1} & \boldsymbol{z_2} \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \frac{1}{\boldsymbol{z_2}^*\boldsymbol{z_1}} & 0 \\ 0 & \frac{1}{\boldsymbol{z_1}^*\boldsymbol{z_2}} \end{bmatrix} \begin{bmatrix} \boldsymbol{z_2}^* \\ \boldsymbol{z_1}^* \end{bmatrix} U \\ &= \begin{bmatrix} U^*\boldsymbol{z_1} & U^*\boldsymbol{z_2} \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \frac{1}{\boldsymbol{z_2}^*\boldsymbol{z_1}} & 0 \\ 0 & \frac{1}{\boldsymbol{z_1}^*\boldsymbol{z_2}} \end{bmatrix} \begin{bmatrix} \boldsymbol{z_2}^*U \\ \boldsymbol{z_1}^*U \end{bmatrix}. \end{aligned}$$

Let $\boldsymbol{a_1} = U^*\boldsymbol{z_1}$ and $\boldsymbol{a_2} = U^*\boldsymbol{z_2}$. Note that

$$\boldsymbol{a_2}^*\boldsymbol{a_1} = (U^*\boldsymbol{z_2})^* U^*\boldsymbol{z_1} = \boldsymbol{z_2}^*UU^*\boldsymbol{z_1} = \boldsymbol{z_2}^*\boldsymbol{z_1}$$

and

$$\boldsymbol{a_1}^*\boldsymbol{a_2} = (U^*\boldsymbol{z_1})^* U^*\boldsymbol{z_2} = \boldsymbol{z_1}^*UU^*\boldsymbol{z_2} = \boldsymbol{z_1}^*\boldsymbol{z_2}.$$

Therefore

$$U^*MU = \begin{bmatrix} \boldsymbol{a_1} & \boldsymbol{a_2} \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \frac{1}{\boldsymbol{a_2}^*\boldsymbol{a_1}} & 0 \\ 0 & \frac{1}{\boldsymbol{a_1}^*\boldsymbol{a_2}} \end{bmatrix} \begin{bmatrix} \boldsymbol{a_2}^* \\ \boldsymbol{a_1}^* \end{bmatrix}.$$

By Theorem 1.18, $\boldsymbol{a_1}$ and $\boldsymbol{a_2}$ are self-orthogonal. Therefore $U^*MU$ is a Lund matrix with the same eigenvalues as $M$. □

We still understand very little about Conjecture 3.12 but hope the results of Theorem 3.13 may be the first step towards proving this conjecture.

# 4 Conclusion

In this project, we have examined the finite field numerical range, specifically focusing on $2\times2$ matrices over the finite field $\mathbb{Z}_7[i]$. We devoted special attention to matrices with unique eigenvalues corresponding to linearly independent, self-orthogonal eigenvectors, which we term Lund matrices.

Section 3 yields numerous interesting results about Lund matrices that we hope can be explored further, with the specific goal of proving Conjecture 3.8. Both the approaches in sections 3.3.1 and 3.3.2 show great promise towards proving this conjecture, and either may be pursued as a future work. As an additional direction, we suggest approaching the proof of Conjecture 3.8 using Kippenhahn's boundary-generating curve, which we have not considered in this project but which has been useful in previous work on finite field numerical ranges. If we are able to prove Conjecture 3.8 for the $2 \times 2$ case over $\mathbb{Z}_7[i]$, this could open up even further investigation of Lund matrices over other finite fields and eventually in the $3 \times 3$ case.

While much more work remains to be done on the numerical range over finite fields, this project presents some very interesting preliminary results and raises even more questions about the structure of the numerical range and the behavior of self-orthogonal vectors. Hopefully our contributions can serve as a starting point for much more research to come in this developing area of mathematics.

# References


[1] O. Toeplitz. Das algebraische analogen zu einem satze von fejár. *Mathematische Zeitschrift*, 2:187–197, 1918.

[2] F. Hausdorff. Der wertvorrat einer bilinearform. *Mathematische Zeitschrift*, 3:314–316, 1919.

[3] Rudolf Kippenhahn. Über den wertevorrat einer matrix. *Mathematische Nachrichten*, 6(3-4):193–228, 1951.

[4] Translated by Paul F. Zachlin and Michiel E. Hochstenbach. On the numerical range of a matrix. *Linear and Multilinear Algebra*, 56(1-2):185–225, 2008.

[5] Jane Ivy Coons, Jack Jenkins, Douglas Knowles, Rayanne Luke, and Patrick Rault. Numerical ranges over finite fields. *Linear Algebra and its Applications*, 501:37–47, 07 2016.

[6] Aishah Ibraheam Basha and Judi J. McDonald. Orthogonality over finite fields. *Linear and Multilinear Algebra*, 70(22):7277–7289, 2022.

[7] Aishah Ibraheam Basha. *Linear Algebra over Finite Fields*. PhD thesis, Washington State University, 2020.

[8] E. Ballico. On the numerical range of matrices over a finite field. *Linear Algebra and its Applications*, 512:162–171, 2017.

[9] S.H. Friedberg, A.J. Insel, and L.E. Spence. *Linear Algebra*. Pearson Education, 2014.

# 5 Appendix I: Selected Code Listings for `linalg_Zpi()`

```
import numpy as np
import itertools
from itertools import permutations

class linalg_Zpi():
    # When initializing the tool, input the dimension n and prime p
    def __init__(self, n, p):
        self.size = n
        self.mod = p
```

Listing 1: Initializing the class `linalg_Zpi()`

```
    # Divide numbers in Zp
    def div_real(self, a, b):
        if b%self.mod==0:
            raise Exception("FREAK OUT")

        return self.div_tbl[b%self.mod, a%self.mod]

    # Divide numbers in Zp[i]
    def div_complex(self, z1, z2):
        num = z1 * np.conj(z2)
        den = z2.real**2 + z2.imag**2

        num = self.sim_s(num)
        den = int(den%self.mod)

        a = self.div_real(int(num.real), den)
        b = self.div_real(int(num.imag), den)

        return complex(a,b)
```

Listing 2: Division methods

```
        # Create table of all possible fractions in Zp
        div_tbl = np.zeros((self.mod, self.mod), dtype=int)

        for i in range(1,self.mod):
            for j in range(1,self.mod):
                for elem in range(1,self.mod):
                    if (elem*i)%self.mod == j:
                        div_tbl[i,j] = elem
                        break

        self.div_tbl = div_tbl
```

Listing 3: Generating the quotient table

```
    # Check if two column vectors are Zp[i] multiples of each other
    def mult(self, v1, v2):
        # This helper function determines whether two elements of Zp[i] are
        # Zp[i] multiples of each other by row-reducing a matrix
        def solve_mult(M):
            if M[0,0] == 0:
                temp = np.array(M[0])
                M[0] = M[1]
                M[1] = temp

            M[0] = self.div_real(np.array(M[0]), M[0,0])
            M[1] -= M[0]*M[1,0]
            M[1]%self.mod

            M[1] = self.div_real(np.array(M[1]), M[1,1])
            M[0] -= M[1]*M[0,1]
            M[0]%self.mod

            # Returns the multiplier
            return complex(M[0,2], M[1,2])

        store = np.empty((self.size, 1), dtype=np.csingle)

        inds = np.arange(0, self.size)

        for i in range(self.size):
            real1 = int(v1[i].real.item())
            imag1 = int(v1[i].imag.item())

            real2 = int(v2[i].real.item())
            imag2 = int(v2[i].imag.item())

            M = np.array([[real1, -imag1, real2],
                          [imag1,  real1, imag2]])

            # Check if the first number is 0 to avoid 0-division errors
            if real1==0 and imag1==0:
                if real2==0 and imag2==0:
                    inds[i]=-1
                    store[i]=-1
                else:
                    return None
            else:
                store[i] = self.sim_s(solve_mult(M))
```

```
    # Check that all of the multipliers are the same
    if np.all(np.take(store, inds[inds>=0]) == np.take(store, inds[inds
>=0])[0]):
        return np.take(store, inds[inds>=0])[0].item()
    else:
        return None
```

Listing 4: The `self.mult()` method

```
# Calculate the inverse of an invertible matrix and freak out if non-
invertible
def inv(self, M):
    M = np.concatenate((M, np.identity(self.size, dtype=int)), axis=1)

    for i in range(self.size-1):
        if M[i,i] == 0:
            for j in range(i+1, self.size):
                if M[j, i] != 0:
                    temp = np.array(M[i])
                    M[i] = M[j]
                    M[j] = temp

        temp = M[i,i]t
        for x in range(2*self.size):
            M[i,x] = self.div_complex(M[i,x], temp)

        for k in range(i+1, self.size):
            M[k] -= M[i]*M[k,i]
            for x in range(2*self.size):
                M[k,x] = self.sim_s(M[k,x])

    temp = M[-1,self.size-1]
    for x in range(2*self.size):
        M[-1,x] = self.div_complex(M[-1,x], temp)

    for l in range(self.size-1,0,-1):
        for n in range(l-1,-1,-1):
            M[n] -= M[l]*M[n,l]
            for x in range(2*self.size):
                M[n,x] = self.sim_s(M[n,x])

    return self.sim_a(M[:,-self.size:])
```

Listing 5: The `self.inv()` method

# 6 Appendix II: Selected Code Listing for gen_norm1()

```
import numpy as np
import itertools
from itertools import permutations
from itertools import product
from linalg_Zpi import linalg_Zpi

class gen_norm1():
    def __init__(self, mod, size):
        inst = linalg_Zpi(p=mod, n=size)
        field = np.arange(0, mod)

        # Generate all vectors x such that x*x=1
        def gen_x():
            combine = list(itertools.combinations_with_replacement(\
                field, 2*size))
            store = np.expand_dims(np.zeros(2*size, dtype=int), axis=0)

            for row in combine[1:]:
                sq_sum = 0
                for elem in row:
                    sq_sum += elem**2
                if sq_sum % mod == 1:
                    array = np.array(row)
                    expanded = np.expand_dims(array, axis=0)
                    store = np.append(store, expanded, axis=0)

            permute = np.expand_dims(np.zeros(2*size, dtype=int), axis=0)

            for row in store[1:]:
                p = list(permutations(row, 2*size))
                for item in p:
                    array = np.array(item)
                    expanded = np.expand_dims(array, axis=0)
                    permute = np.append(permute, expanded, axis=0)

            permute = np.unique(permute, axis=0).reshape(-1, size, 2)

            x = np.empty((permute.shape[0], size, 1), dtype=np.csingle)

            for i in range(permute.shape[0]):
                for j in range(size):
                    x[i, j] = complex(permute[i,j,0], permute[i,j,1])

            return x

        # Find minimal list of norm-1 vectors
        def sim_x():
            norm0 = gen_x()[1:]
            length = len(norm0)

            inds = np.arange(norm0.shape[0])

            for i in range(length-1):
                vector = norm0[i]

                for j in range(i+1,length):
```

```
                if inds[j]== -1:
                    continue
                else:
                    if inst.mult(vector, norm0[j]) != None:
                        inds[j]= -1

        return norm0[inds[inds!=-1]]

    self.x = sim_x()

    # Generate vectors x* from x
    def gen_xstar():
        x_star = np.empty((len(self.x), size), dtype=np.csingle)

        for i in range(len(self.x)):
            x_star[i] = np.transpose(np.conjugate(self.x[i]))

        return x_star

    self.xstar = gen_xstar()

    # Set class variables
    self.mod = mod
    self.size = size
```

Listing 6: The `gen_norm1()` class

# 7 Appendix III: Selected Code Listing for `numrange_Zpi()`

```
import numpy as np
import matplotlib.pyplot as plt
from Imports.linalg_Zpi import linalg_Zpi

class numrange_Zpi():
    def __init__(self, n, p, M):
        # Load in norm-0 vectors
        str_norm0 = "Imports/norm0/p="+str(p)+"_n="+str(n)+".npz"
        self.norm0 = np.load(str_norm0)["norm0"]

        # Load in norm-1 vectors
        str_norm1 = "Imports/norm1/p="+str(p)+"_n="+str(n)+".npz"
        self.norm1 = np.load(str_norm1)["norm1"]

        self.linalg = linalg_Zpi(n=n, p=p)

        self.M = M
        self.n = n
        self.p = p

    def W1(self, plot=True):
        W1 = np.empty(len(self.norm1), dtype=np.csingle)

        for i in range(len(self.norm1)):
            x = self.norm1[i]
            xstar = self.linalg.sim_a(np.conj(x.transpose()))

            W1[i] = self.linalg.prod((xstar, self.M, x))

        out = np.unique(W1)

        if plot==True:
            plt.scatter(out.real, out.imag, s=200, c='red')
            plt.axis([0, self.p-1, 0, self.p-1])
            plt.show

        return out
```

Listing 7: The `numrange_Zpi()` class