Fast Matrix Multiplication and the Wedderburn-Artin Theorem

By

Zachary Dubinsky

* * * * * * * * *

Submitted in partial fulfillment

of the requirements for

Honors in the Department of Computer Science

UNION COLLEGE

June 6, 2023

Abstract

Researchers Cohn and Umans [4][5] proposed a framework for fast matrix multiplication algorithms. Their approach is reliant on an application of the Wedderburn-Artin Theorem: a landmark classification result in modern algebra. We show experimental success for algebras whose components all have dimension 1. We advance the Cohn and Umans framework by developing new, extendable tools to couple with their design.

Contents

List of Figures

List of Tables

1 Introduction

Matrices are an indispensable tool in various fields including mathematics, engineering, physics, and computer science. Most simply, these objects represent an arrangement of entries by row and column positions. Consider the following 2×2 matrices:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$
$$B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}.$$

Additional structure is imposed by the **matrix multiplication** operation. Given two matrices, the matrix product is defined by dot products of rows and columns. In the case of 2×2 matrix multiplication, we compute the product as follows:

$$\begin{aligned} AB &= \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \\ &= \begin{bmatrix} 1 \cdot 5 + 2 \cdot 7 & 1 \cdot 6 + 2 \cdot 8 \\ 3 \cdot 5 + 4 \cdot 7 & 3 \cdot 6 + 4 \cdot 8 \end{bmatrix}. \end{aligned}$$

More generally, let A and B be two $n \times n$ matrices. Then the i^{th} entry of AB is equal to the vector dot product of the i^{th} row of A with the j^{th} column of B.

In the case of $n \times n$ matrix multiplication, there are n^2 entries in the product matrix to be computed. For each entry we perform n multiplications and $n - 1$ additions, or $\mathcal{O}(n)$ operations. Thus, the naive efficiency of $n \times n$ matrix multiplication is $\mathcal{O}(n^3)$. Strassen [17] gave a ground breaking algorithm for matrix multiplication with complexity $\mathcal{O}(n^{2.81})$. Coppersmith and Winograd [6] improved on this result with their $\mathcal{O}(n^{2.372})$ algorithm. These algorithms sparked an investigation into determining the optimal efficiency for matrix multiplication. Over the last fifty years, researchers have sought after the lowest exponent ω for which a matrix multiplication algorithm with running time $\mathcal{O}(n^\omega)$ exists.

Cohn and Umans [4][5] published a theoretical framework for fast matrix multiplication. Their work relies on two ideas: computing a representation of matrices as elements of a group algebra, and an application of the Wedderburn-Artin Theorem. The representation step has been implemented for certain cases by Anderson [1]. This thesis focuses on the latter step. Namely, we advance the Cohn and Umans framework by implementing an algorithm compute the mapping guaranteed by the Wedderburn-Artin Theorem. In

composition with their framework, we have the tools to implement a simple instance of Cohn and Umans matrix multiplication.

In Section 2, we present a high-level overview of the Cohn and Umans framework. In Section 3, we delve further into the Wedderburn-Artin Theorem to characterize an implementation of this result, and present the relevant design decisions for the implementation. Section 4 and Section 5 describe the course of implementation from theory to practice. Results are shown in Section 6 and areas for advancement on this topic are provided in Section 7.

2 Cohn and Umans Matrix Multiplication

Here, we present the Cohn and Umans [4][5] framework for fast matrix multiplication. We begin by recalling the notion of a **finite group** G, and the **group algebra** $\mathbb{C}[G]$. These structures are the backbone of Cohn and Umans' algorithm. The reader is referred to Dummit and Foote [8] for a background on modern algebra.

2.1 The Embedding of Matrices

The first stage of the Cohn and Umans matrix multiplication algorithm is producing a representation for matrices. Suppose we have a matrix with entries in $\mathbb{C}$,

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}.$$

We want to **embed** this matrix in the group algebra $\mathbb{C}[G]$, where G is a group containing at least four elements. Let $g_1, g_2, g_3, g_4 \in G$ be distinct elements of the group G. We obtain the following representation:

$$\bar{A} = 1g_1 + 2g_2 + 3g_3 + 4g_4 \in \mathbb{C}[G]. \tag{1}$$

The matrix A is now expressed as the element $\bar{A}$ of the group algebra $\mathbb{C}[G]$, where $\mathbb{C}$ is the base field of the matrix A. We remark that this is a naive description of the embedding. For a working implementation of matrix multiplication our group and embedding strategy must satisfy the *triple-product property*. For now, we highlight the features of this embedding structure that are necessary for the topic.

Elements of the group algebra $\mathbb{C}[G]$ are formal sums much like a polynomial in an indeterminate x. The coefficients in Equation 1 are elements of the field $\mathbb{C}$, defined with the usual multiplication and addition op-

erations from $\mathbb{C}$. The group elements are defined with a multiplication operation between them. However, field elements and group elements have no operations defined between them.

Elements of a group algebra can be multiplied similarly to polynomials:

$$\bar{A}\bar{B} = \Big(\sum_{i=1}^{n} \bar{A}_i g_i\Big)\Big(\sum_{j=1}^{n} \bar{B}_j g_j\Big) = \sum_{i=1}^{n}\sum_{j=1}^{n}(a_i b_j)(g_i g_j). \tag{2}$$

We remark that the transformation from a matrix to an element of a group algebra is invertible. With an appropriate choice of group for 2 matrices, and a correct embedding, we can compute their product in the algebra. We then apply the inverse transformation to recover the product matrix from the product of algebra elements.

2.2 The Wedderburn-Artin Theorem

The second stage of the Cohn and Umans algorithm is where speedups are realized. We rely on an application of the Wedderburn-Artin Theorem [2][18], and we present this result in the context of this application.

Let G be a finite group and $\mathbb{C}[G]$ the group algebra. By the Wedderburn-Artin Theorem there exists an invertible function Φ such that,

$$\Phi : \mathbb{C}[G] \longrightarrow \mathbb{C}^{d_1 \times d_1} \times \cdots \times \mathbb{C}^{d_c \times d_c} \tag{3}$$

for $d_i \in \mathbb{N}$. The right-hand-side is a block diagonal matrix of $d_i \times d_i$ matrices with entries in $\mathbb{C}$. We call this form a **Weddeburn decomposition** of the group algebra $\mathbb{C}[G]$, and each $\mathbb{C}^{d_i \times d_i}$ is a **Wedderburn component**.

We can compute products of group algebra elements in this Wedderburn decomposition space. Let $\bar{A}, \bar{B} \in \mathbb{C}[G]$. Then,

$$\begin{aligned}
\Phi(\bar{A})\Phi(\bar{B}) &= \begin{bmatrix} A_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & A_c \end{bmatrix} \begin{bmatrix} B_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & B_c \end{bmatrix} \\
&= \begin{bmatrix} A_1B_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & A_cB_c \end{bmatrix} \\
&= \Phi(\bar{A}\bar{B}).
\end{aligned} \tag{4}$$

Each $A_i, B_i \in \mathbb{C}^{d_i \times d_i}$ is a $d_i \times d_i$ matrix in with complex entries. These are the Wedderburn components.

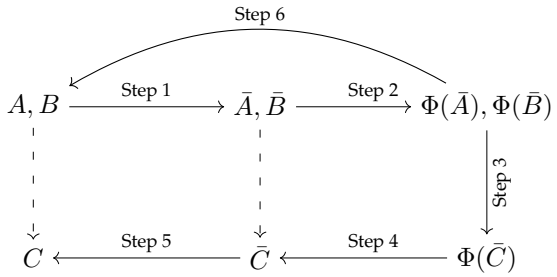

Figure 1: Cohn and Umans Matrix Multiplication

Since Φ is invertible, we can apply its inverse to $\Phi(\bar{A}\bar{B})$ and obtain the product $\bar{A}\bar{B} \in \mathbb{C}[G]$.

2.3 Matrix Multiplication

We now present the full Cohn and Umans [4][5] algorithm. Let G be a finite group, satisfying the *triple-product property*, and let A, B be complex-valued matrices. We present MATRIXMULTIPLICATION(A, B, G) in Algorithm 1. We also illustrate these steps graphically in Figure 2.3.

Speedups are realized during Step 3 and improved upon by Step 6. The former is the block diagonal matrix multiplication step. As shown in Equation 4, we can compute the matrix products of the block matrices pointwise. Each block has dimension $d_i \times d_i$. We achieve improvements on the naive exponent $\omega = 3$ when the following inequality holds:

$$\sum_{i=1}^{k} d_i^3 < n^3. \tag{5}$$

Cohn and Umans have shown many group algebras afford a decomposition such that the inequality holds. However, we can improve on this inequality. Step 6 indicates a recursive call to the Cohn and Umans algorithm. Since computing the product of two block diagonal matrices is equivalent to computing individual products of the blocks, we further reduce the problem by decomposing the block sub-product. This leads to even more speedups. We show the recursive pattern in more detail on Line 11 of Algorithm 1. Using this argument, Cohn and Umans were able to demonstrate a theoretical upper bound of $\omega = 2.41$ [5].

Cohn and Umans give a rigorous treatment of Steps 1 and 5. Their computation is well-understood, and implementations have been produced by Anderson [1]. Step 3 is also well-understood, the algorithm for which is naive matrix multiplication.

Steps 2 and 4 remain elusive. Algorithms for computing the Wedderburn-Artin theorem are computationally complex and niche. The package Wedderga [19] of the computational discrete algebra system GAP [10] is the closest we have found. However, the Wedderga implementation only returns the dimensions

Algorithm 1 Cohn and Umans Matrix Multiplication

Require: $A, B \in \mathbb{C}^{n \times n}$
Require: $|G| \geq n^2$
Require: G is a group with the *Triple-Product Property*
1: **function** MATRIXMULTIPLICATION(A, B, G)
2: **if** DIM(A)≤ 4 **then** ▷ Base Case: No speedups in reduction
3: **return** AB
4: **end if**
5: $\bar{A} \leftarrow$EMBED(A, G) ▷ Embedding
6: $\bar{B} \leftarrow$EMBED(B, G)
7: $\Phi \leftarrow$COMPUTEWATHM(G) ▷ Define the Wedderburn-Artin Theorem mapping for $\mathbb{C}[G]$
8: $A_1, ..., A_k \leftarrow \Phi(\bar{A})$ ▷ Compute the matrix blocks
9: $B_1, ..., B_k \leftarrow \Phi(\bar{B})$
10: **for** $i \in [1, k]$ **do**
11: $C_i \leftarrow$MATRIXMULTIPLICATION(A_i, B_i, G') ▷ Call with optimal group for dimensions
12: **end for**
13: $\bar{C} \leftarrow \Phi^{-1}(C_1, ..., C_k)$ ▷ Inverse Wedderburn-Artin Theorem mapping
14: $C \leftarrow$ UNEMBED($\bar{C}, G$) ▷ Inverse embedding
15: **return** C
16: **end function**

of the Wedderburn components. In an effort perform Cohn and Umans matrix multiplication, we implemented an algorithm which returns Φ as an explicit mapping of elements of a group algebra to matrices in the Wedderburn decompositions.

3 Design Decisions

In this section I present the major design choices for my implementation of the Wedderburn-Artin Theorem.

3.1 On the Choice of Field

In Equation 3 we present a case of the Wedderburn-Artin Theorem for the group algebra of a finite group over the complex numbers. We discuss a restriction on the input types.

As we show in Section 4, representing a finite group is fairly simple because there are a discrete number of elements. However, the fields $\mathbb{R}$ and $\mathbb{C}$ do not share this trait. These fields contain transcendental elements that cannot be expressed by any finite number of bits. Issues with numerical stability would arise, so it is desirable to select a field that can be represented with finite bit length.

The rational numbers $\mathbb{Q}$ can be represented as a pair of integers: the numerator and the denominator. Provided we do not surpass the threshold of integer size, any rational number can be represented on a

discrete system. Let G be a finite group, the Wedderburn-Artin Theorem [2][18] yields the following:

$$\Phi : \mathbb{Q}[G] \longrightarrow D_1^{d_1 \times d_1} \times \cdots \times D_c^{d_c \times d_c} \tag{6}$$

for $d_i \in \mathbb{N}$. Upon inspection, Equation 3 and Equation 6 are similar as they both return a block diagonal matrix. The distinguishing feature is in the entries of each block: algebras over $\mathbb{C}$ yield matrices over the complex numbers; algebras over $\mathbb{Q}$ yield matrices over the fields D_i.

It is often the case that $D_i = \mathbb{Q}$. However, sometimes D_i is called a **cyclotomic field**. Cyclotomic fields, $\mathbb{Q}(\zeta_n)$, are the usual rationals with the n^{th} root of unity adjoined. An example of such a field is the Gaussian rationals, $\mathbb{Q}(\zeta_4) = \mathbb{Q}(\sqrt{-1})$, but cyclotomics are not the focus of this thesis. We mention such fields because it was shown by the Brauer-Witt Theorem [20] that all of these D_i's are a cyclotomic, even if only the trivial case of $n \leq 2$.

By using algebras of a finite group over the rationals we can represent both the inputs and outputs discretely. We also avoid the issues of numerical stability from $\mathbb{R}$ and $\mathbb{C}$. In fact, we can use any field that can be represented discretely as we discuss in Section 6.2.

3.2 Requirements and Programming Languages

The subroutines in this paper were presented in a theoretical context, and thus trade implementation feasibility for literary simplicity. Many rely on niche functionality that is beyond the scope of this implementation effort. In order to select and language and produce meaningful code, we had to outline our requirements and devise solutions.

First, we rely on a number of concepts from linear algebra. We require data structures for representing the rational numbers, matrices, and vectors. We also need algorithms for solving systems of equations, computing echelon form, and performing other elementary linear algebra operations.

Second, ideas from abstract algebra are imbued in many of our subroutines. The best example of this is our use of extension fields. We need data structures and algorithms for representing, altering, and performing computation within these extension fields. In particular, we need an algorithm for factoring in abstract polynomial rings.

Our investigation into producing Wedderburn decompositions began initially in GAP [10]. GAP is a robust, computational discrete algebra system. It is a functional programming language focused on group theory. This system proved difficult to write in, though useful for testing.

Some of our work was completed in Python. The Sympy [14] library provides all of the linear algebra tools for a partial implementation. However, Sympy lacks sufficient functionality for working with

extension fields. This motivated a return to computational discrete algebra systems with more capabilities.

Our final implementation was done in SageMath [16]. SageMath is effectively a Python interface to a number of other computational discrete algebra systems. The simplicity of the Python style pairs well with the relatively high capabilities of the language. They provide all of the data structures and algorithms we needed to implement the Wedderburn-Artin Theorem. With all of our design decisions accommodated, this language proved most ideal.

4 Working in the Representation

We now describe our representation strategy for algebras, and the linear algebraic constructions we use for performing computations with them. Throughout this section we let $\mathcal{A} = \mathbb{Q}[G]$ for a finite group $G = \{g_1, ..., g_n\}$ for brevity.

4.1 The Standard Representations

By definition of a finite dimensional group algebra, every element of $\mathcal{A}$ is of the form,

$$u_1 g_1 + \cdots + u_n g_n$$

with coefficients $u_1, ..., u_n \in \mathbb{Q}$. It is often easier to represent these elements by a vector of their coefficients, indexed by the subscript on group elements:

$$u_1 g_1 + \cdots + u_n g_n = \begin{bmatrix} u_1 & \cdots & u_n \end{bmatrix}.$$

Moreover, we can identify g_i by the i^{th} unit vector in $\mathbb{Q}^n$. We say G is a **standard basis** of the algebra $\mathcal{A}$. The standard basis of group elements is canonical for $\mathcal{A} = \mathbb{Q}[G]$.

We also need a representation for the product operation on elements of an algebra. We begin with algebras given over the standard basis of group elements. For this, we rely on the closure axiom of groups: that the product of any two group elements is another group element. We define the **structure constants** of this basis as the tensor $C \in \{0, 1\}^{n \times n \times n}$ such that,

$$C(i, j, k) = \begin{cases} 1 \text{ if } g_i g_j = g_k \\ 0 \text{ if } g_i g_j \neq g_k \end{cases}. \tag{7}$$

We can compute the product of two elements $a, b \in \mathcal{A}$, given with respect to the standard basis of group elements, in the following way:

$$\begin{aligned} ab &= \Big(\sum_{i=1}^{n} u_i g_i\Big)\Big(\sum_{j=1}^{n} u'_j g_j\Big) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} u_i u'_j g_i g_j \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} u_i u'_j \sum_{k=1}^{n} C(i,j,k) g_k \end{aligned}$$

for $u_i, u'_j \in \mathbb{Q}$. We note that the right-hand-side can also be represented as a vector with respect to the standard basis of group elements.

To obtain structure constants we first impose an indexed ordering of the group basis. We then compute the products $g_i g_j$ and store the result in our structure constant tensor. This can be done by hand, transcribed from a Cayley table, or implemented with some computational representation.

These structure constants $C \in \{0,1\}^{n\times n\times n}$ are the inputs to our program. They are effectively multi-dimensional arrays of length n along all axes. This length is sufficient to generate a standard basis of unit vectors for a complete representation of the algebra.

4.2 The General Representation

We will occasionally need to represent non-trivial subalgebras of $\mathcal{A}$ with distinct bases. Let $\mathcal{A}$ be an algebra and consider the subalgebra $I \subseteq \mathcal{A}$ with **derived basis** $B = \{a_1, ..., a_c\}$ for some $c \leq dim(\mathcal{A}) = n$. Then every element of I is of the form,

$$u_1 a_1 + \cdots + u_c a_c = \begin{bmatrix} u_1 & \dots & u_c \end{bmatrix}$$

for some $u_j \in \mathbb{Q}$. Further structure is imposed on this construction with a **change of basis**. Each basis element of I is of the form,

$$a_j = \sum_{\ell=1}^{n} u'_\ell g_\ell = \begin{bmatrix} u'_1 & \dots & u'_n \end{bmatrix}$$

for some $u'_\ell \in \mathbb{Q}$. Then we can express any element of I with respect to B, and with respect to the basis of $\mathcal{A}$.

This is necessary for computing products in a subalgebra if it does not have a standard basis. Then we must compute a change of basis and use the group structure constants $C \in \{0,1\}^{n\times n\times n}$ of $\mathcal{A}$ to compute

products:

$$\begin{aligned} a_i a_j &= \sum_{k=1}^{n} u_k g_k \sum_{\ell=1}^{n} u'_\ell g_\ell \\ &= \sum_{k=1}^{n} \sum_{\ell=1}^{n} u_k u'_\ell g_k g_\ell \\ &= \sum_{k=1}^{n} \sum_{\ell=1}^{n} u_k u'_\ell \sum_{p=1}^{n} C(k,\ell,p) g_p \end{aligned}$$

for some $u_k, u'_\ell \in \mathbb{Q}$. Our derived basis is enough to represent subalgebras or to create other interesting structures. We can even compute the products of elements in this space. The work that remains is to embellish these spaces.

We consider the same subalgebra algebra $I \subseteq \mathcal{A}$ with basis B. Now let $F \in \mathbb{Q}^{c\times c\times c}$ be the **derived structure constants** for this basis of $\mathcal{A}$ such that,

$$a_i a_j = \sum_{k=1}^{c} F(i,j,k) a_k.$$

Since $I \subseteq \mathcal{A}$, each $a_i \in B$ can be expressed over the canonical basis G. Note that we can always compute products in I with a change of basis and the structure constants in $C \in \{0,1\}^{n\times n\times n}$. However, with derived structure constants we can treat B as a standard basis. Identify each $a_i \in B$ with the i^{th} unit vector in $\mathbb{Q}^c$. Then we can multiply elements of I when expressed over the standard basis B with the structure constants in $F \in \mathbb{Q}^{c\times c\times c}$.

4.3 Constraining the Algebra

Let $\mathcal{A}$ be a group algebra algebra of a finite group over the rationals. Let $B = \{a_1, ..., a_n\}$ be some standard basis with structure constants in $C \in \mathbb{Q}^{n\times n\times n}$. The problem of constraining $\mathcal{A}$ concerns finding elements. Sets of elements, or just one, can often be classified by their invariants under the product operation. Respective examples are the center of an algebra or the identity element. We aim to express these product invariants as a linear system. We can then solve for a basis of the desired elements. Let $\alpha = x_1 a_1 + \cdots + x_n a_n$

be an indeterminate element of the algebra with $x_i \in \mathbb{Q}$. Then the product of the basis element a_1 with α is,

$$\begin{aligned} a_1\alpha &= a_1 \sum_{j=1}^{n} x_j a_j \\ &= \sum_{j=1}^{n} x_j a_1 a_j \\ &= \sum_{j=1}^{n} x_j \sum_{k=1}^{n} C(1,j,k) a_k \\ &= \sum_{j=1}^{n} x_j (C(1,j,1)a_1 + \cdots + C(1,j,n)a_n). \end{aligned}$$

This is a linear combination of the standard basis B. It is an element of $\mathcal{A}$, we express in the vector representation over B:

$$\begin{aligned} &= \sum_{j=1}^{n} x_j \begin{bmatrix} C(1,j,1) & \dots & C(1,j,n) \end{bmatrix} \\ &= x_1 \begin{bmatrix} C(1,1,1) & \dots & C(1,1,n) \end{bmatrix} + \cdots + x_n \begin{bmatrix} C(1,n,1) & \cdots & C(1,n,n) \end{bmatrix}. \end{aligned}$$

This sum of scaled vectors is equivalent to the matrix-vector product,

$$\begin{aligned} &= \begin{bmatrix} C(1,1,1) & \dots & C(1,n,1) \\ \vdots & & \vdots \\ C(1,1,n) & \dots & C(1,n,n) \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \\ &= C(1,\infty,\infty) \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}. \end{aligned}$$

We define the matrix $C(1,\infty,\infty) \in \mathbb{Q}^{n\times n}$ such that $C(1,\infty,\infty)\alpha = a_1\alpha$. Compute the left-hand-side as a matrix-vector product and the right-hand-side by the structure constants in C. This holds for any $\alpha \in \mathcal{A}$. More generally, $C(i,\infty,\infty)\alpha = a_i\alpha$ for the particular basis element a_i of $\mathcal{A}$. We use ∞ as a sentinel to indicate complete access to the full range of values of the specified axis of the tensor.

We remark that the argument is symmetric to show that $C(\infty,i,\infty)\alpha = \alpha a_i$. We also abstract this con-

struction to express the product of *every* basis element of the algebra with the indeterminate element α :

$$\begin{bmatrix} C(1,\infty,\infty) \\ \vdots \\ C(c,\infty,\infty) \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_1\alpha \\ \vdots \\ a_c\alpha \end{bmatrix}. \tag{8}$$

The right-hand-side denotes a $1 \times n^2$ vector with entries in $\mathbb{Q}$. Each $a_i\alpha$ are correspond to a n-length range of product column. As coordinates on the basis B, the range $a_i\alpha$ is equal to the computation of the same product by structure constants.

From our construction, the matrix on the left-hand-side is fixed for the structure constants in $C \in \mathbb{Q}^{n\times n\times n}$. We want to constrain the algebra by the properties of the product we want obeyed. This amounts to manipulating the system. Fix $\alpha \in \mathcal{A}$. If we set every $a_i\alpha = \alpha$ in the solution vector, the solutions to the system form a basis for $\beta \in \mathcal{A}$ satisfying $a_i\beta = \alpha$. Conversely, substituting α for the indeterminate vector yields a basis for $\mathcal{A}\alpha$. We return to this idea throughout, and manipulate further to find bases for structures of interest.

4.4 Lifting the Basis

Earlier, we presented a change of basis technique which takes an element of a subalgebra and changes its basis to that of the full algebra. We now present the opposite, that is, a technique to take an element of an algebra and changes its basis to that of a subalgebra.

Let $\mathcal{A}$ be an algebra and $I \subset \mathcal{A}$ a subalgebra with derived basis $B = \{u_1, ..., u_c\}$, given over some standard basis of $\mathcal{A}$. Let $a \in \mathcal{A}$, given over the same standard basis. We define the following matrix:

$$P = \begin{bmatrix} | & & | & | \\ u_c & \dots & u_1 & a \\ | & & | & | \end{bmatrix}. \tag{9}$$

Each column the transpose of the vector representation of the denoted element. Compute the row-echelon form of P. The first c entries of the last column are the coordinates of a with respect the basis B. We note that if $a \notin I$, then a cannot be expressed by this basis. The coefficients we obtain are zero in this case.

5 Implementations

Throughout this section we present our implementations with regards to the computation of explicit Wedderburn decompositions. We consider the n-dimensional group algebra $\mathcal{A} = \mathbb{Q}[G]$ over some arbitrary standard basis $B = \{a_1, ..., a_n\}$. Structure constants for B are in $C \in \mathbb{Q}^{n \times n \times n}$. B and C are defined in this way for generality, but we can use the group basis and corresponding structure constants for the initial inputs.

5.1 The Center of the Group Algebra

My first algorithm computes the center of the algebra: $Z(\mathcal{A}) = \{a \in \mathcal{A} : ax = xa \; \forall x \in \mathcal{A}\}$. This is the set of all elements of $\mathcal{A}$ that commute with every other element of $\mathcal{A}$. It is a subset of $\mathcal{A}$, and an algebra in its own right.

5.1.1 Theory

We have from Ivanyos and Ronyai [11] that the Wedderburn-Artin Theorem applies to the algebra $Z(\mathcal{A})$. In fact, there is a correspondence between the Wedderburn decomposition of the whole algebra and that of its center:

$$\begin{aligned} \Phi(\mathcal{A}) &= D_1^{d_1 \times d_1} \times \cdots \times D_c^{d_c \times d_c} \\ \Phi(Z(\mathcal{A})) &= D_1 \times \cdots \times D_c \end{aligned} \tag{10}$$

Moreover, we have that $D_i^{d_i \times d_i} = D_i\mathcal{A}$ by the span of their respective bases where the right-hand-side is viewed as a free module.

This is to motivate a reduction to the commutative case. It is much easier to derive Φ for commutative algebras then for non-commutative algebras. Subsequently lifting Φ to the elements of the whole algebra does not prove difficult.

5.1.2 Basis

We begin by finding a basis for $Z(\mathcal{A})$. We use the linear system strategy presented in Section 4.3. Let $\alpha = x_1a_1 + \ldots + x_na_n \in \mathcal{A}$ be an indeterminate element and consider the following two systems:

$$\begin{bmatrix} C(1,\infty,\infty) \\ \vdots \\ C(n,\infty,\infty) \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_1\alpha \\ \vdots \\ a_n\alpha \end{bmatrix} \tag{11}$$

$$\begin{bmatrix} C(\infty,1,\infty) \\ \vdots \\ C(\infty,n,\infty) \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \alpha a_1 \\ \vdots \\ \alpha a_n \end{bmatrix}. \tag{12}$$

From the solution vectors we can see that Equation 11 and Equation 12 compute the same products in opposite order. Then $\alpha \in Z(\mathcal{A})$,

$$\iff \begin{bmatrix} a_1\alpha \\ \vdots \\ a_n\alpha \end{bmatrix} = \begin{bmatrix} \alpha a_1 \\ \vdots \\ \alpha a_n \end{bmatrix} \tag{13}$$

$$\iff \begin{bmatrix} C(1,\infty,\infty) \\ \vdots \\ C(n,\infty,\infty) \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} C(\infty,1,\infty) \\ \vdots \\ C(\infty,n,\infty) \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$\iff \begin{bmatrix} C(1,\infty,\infty) \\ \vdots \\ C(n,\infty,\infty) \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} - \begin{bmatrix} C(\infty,1,\infty) \\ \vdots \\ C(\infty,n,\infty) \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \vec{0}$$

$$\iff \left(\begin{bmatrix} C(1,\infty,\infty) \\ \vdots \\ C(n,\infty,\infty) \end{bmatrix} - \begin{bmatrix} C(\infty,1,\infty) \\ \vdots \\ C(\infty,n,\infty) \end{bmatrix} \right) \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \vec{0}.$$

The parenthesized matrices are both $n^2 \times n$ and fixed, so we compute their difference and solve for $x_1, ..., x_n$. Note that $\vec{0}$ is a $1 \times n^2$ vector. Solutions are coordinates for α over B such that $a_i\alpha = \alpha a_i$ for all basis elements $a_i \in B$. Then α commutes with all elements of the algebra by distributivity.

Form this system in SageMath and use the Sage kernel function to find solutions. I Construct a matrix with each solution as a row and compute the reduced row-echelon form to ensure linear independence.

The rows form a derived basis for $Z(\mathcal{A})$. This algorithm has running time polynomial in n.

5.1.3 Structure Constants

We now have a derived basis $Z = \{z_1, ..., z_c\} \subseteq Z(\mathcal{A})$ for the center, given with respect to a standard basis B. We form structure constants for Z. In this way, we can compute products of central elements without using a change of basis.

We follow the technique presented in Section 4.4. Compute the product of basis elements $z_i z_j$, given with respect to B, using the structure constants in $C \in \mathbb{Q}^{n \times n \times n}$. This can be done using the change of basis technique from Section 4.2. Then form the following matrix:

$$P = \begin{bmatrix} | & & | & | \\ z_1 & \cdots & z_c & z_i z_j \\ | & & | & | \end{bmatrix}$$

Compute the reduced row-echelon form of P. As presented in Section 4.4, the first c entries of the last column are the coordinates of $z_i z_j \in Z(\mathcal{A})$ with respect to the basis B. Call these coordinates $u_1, ..., u_c \in \mathbb{Q}$. Then,

$$z_i z_j = u_1 z_1 + \cdots + u_c z_c$$

and,

$$z_i z_j = \sum_{k=1}^{c} F(i,j,k) z_k$$

where $F(i,j,k) = u_k$ and $F \in \mathbb{Q}^{c \times c \times c}$. These are the derived structure constants for $Z(\mathcal{A})$. Then use SageMath to form this matrix, compute its reduced row-echelon form, and store the coefficients in the structure constant tensor. This algorithm has running time polynomial in c and n. With these structure constants, we can now treat $Z = \{z_1, ..., z_c\}$ as a standard basis of $Z(\mathcal{A})$.

5.2 The Ideals of a Commutative Group Algebra

This section presents three algorithms regarding the ideals of commutative, finite dimensional group algebras. Correctness of these algorithms was proved by Friedl and Rònyai [9], but Bremner [3] gives the "cleanest" presentation.

Throughout this section we work with $Z(\mathcal{A})$ as our commutative group algebra of dimension $1 \leq c \leq dim(\mathcal{A}) = n$ over the rationals. We treat $Z = \{z_1, ..., z_c\} \subseteq \mathbb{Q}^c$ as a standard basis with structure constants in $F \in \mathbb{Q}^{c \times c \times c}$.

5.2.1 Generating an Ideal

Let $v \in Z(\mathcal{A})$ be an arbitrary element. A **principle ideal** is an ideal generated by a single element:

$$Z(\mathcal{A})v = \{zv : \forall z \in Z(\mathcal{A})\}. \tag{14}$$

We note that $Z(\mathcal{A})$ is a commutative subalgebra of $\mathcal{A}$, and thus any principle ideal of $Z(\mathcal{A})$ is also a commutative subalgebra. We use our technique from Section 4.3 to find the basis of the algebra. In this case, we need not solve for the indeterminates that yield a certain element, our indeterminate is $v = u_1 z_1 + \cdots + u_c v_c$.

Note that $Z(\mathcal{A})$ is commutative, so its left and right ideals are the same. Let $u_i \in \mathbb{Q}$ be the coordinate on the i^{th} basis vector in Z for $1 \leq i \leq c$. We compute the product of v with every basis element to obtain a new basis for $Z(\mathcal{A})v$ using the following matrix:

$$\begin{bmatrix} F(1, \infty, \infty) \\ \vdots \\ F(c, \infty, \infty) \end{bmatrix} \begin{bmatrix} u_1 \\ \vdots \\ u_c \end{bmatrix} = \begin{bmatrix} z_1 v \\ \vdots \\ z_c v \end{bmatrix}.$$

Observe that we have changed the structure constants tesnor for this system. Since we treat Z as a standard basis we must use the structure constants for Z. On the right-hand-side we obtain a $1 \times c^2$ vector with entries in $\mathbb{Q}$. Each $z_i v$ represents a c-length range of coordinates for a basis vector of the ideal, given with respect to Z.

We form a matrix with each basis vector as a row and compute the reduced row-echelon form. We obtain $Y = \{y_1, ..., y_d\} \subseteq \mathbb{Q}^d$ linearly independent vectors ($1 \leq d \leq c$). This is the basis of the ideal generated by v. Use SageMath to compute the linear system, construct Y, and compute the reduced row-echelon form. This algorithm has running time polynomial in c.

5.2.2 The Identity of an Ideal

We now have a derived basis $Y = \{y_1, ..., y_d\}$, given with respect to Z, for a principle ideal of $Z(\mathcal{A})$. Z is a standard basis for the center with structure constants in $\mathbb{F}^{c \times c \times c}$. We want to find the identity element of this ideal, we utilize the approach from Section 4.3.

We need to find the indeterminate $e = x_1y_1 + \cdots + x_dy_d$ such that $ey_i = y_ie$ for $1 \leq i \leq d$. We remark that a construction similar to Equation 8 is exhaustive because Y is a derived basis. We were able to derive the sub-matrices, $C(i, \infty, \infty)$, for a standard basis in short fashion. We will demonstrate that this is not the case here, and present an alternative approach.

First, let $y_i \in Y$. We denote the j^{th} coordinate of this basis vector as $y_{ij} \in \mathbb{Q}$. Consider the indeterminate product y_ie for some $y_i \in Y$:

$$\begin{aligned}
y_ie &= \sum_{j=1}^{c} y_{ij}z_j \sum_{k=1}^{d}\sum_{\ell=1}^{c} x_k y_{k\ell} z_\ell \\
&= \sum_{j=1}^{c} y_{ij} \sum_{k=1}^{d}\sum_{\ell=1}^{c} x_k y_{k\ell} z_j z_\ell \\
&= \sum_{j=1}^{c} y_{ij} \sum_{k=1}^{d}\sum_{\ell=1}^{c} x_k y_{k\ell} \sum_{m=1}^{c} F(j,\ell,m) z_m \\
&= \sum_{j=1}^{c} y_{ij} \sum_{k=1}^{d}\sum_{\ell=1}^{c} x_k y_{k\ell} (F(j,\ell,1)z_1 + \cdots + F(j,\ell,c)z_c).
\end{aligned}$$

We can express the sum as a vector with respect to the standard basis of $Z(\mathcal{A})$:

$$\begin{aligned}
&= \sum_{j=1}^{c} y_{ij} \sum_{k=1}^{d}\sum_{\ell=1}^{c} x_k y_{k\ell} \begin{bmatrix} F(j,\ell,1) & \dots & F(j,\ell,c) \end{bmatrix} \\
&= \sum_{j=1}^{c} y_{ij} \sum_{k=1}^{d}\sum_{\ell=1}^{c} y_{k\ell} \Big(x_1 \begin{bmatrix} F(j,\ell,1) & \dots & F(j,\ell,c) \end{bmatrix} + \cdots + x_d \begin{bmatrix} F(j,\ell,1) & \dots & F(j,\ell,c) \end{bmatrix} \Big) \\
&= \sum_{j=1}^{c} y_{ij} \sum_{k=1}^{d}\sum_{\ell=1}^{c} y_{k\ell} \begin{bmatrix} F(j,\ell,1) & \dots & F(j,\ell,1) \\ \vdots & & \vdots \\ F(j,\ell,c) & \dots & F(j,\ell,c) \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}.
\end{aligned}$$

We have shown that this process does permit the formation of a linear system. We can see the product of a $c \times d$ matrix with a $1 \times d$ indeterminate vector. Advancing further proves challenging in finding the appropriate dot products to construct the cd equations for the system. Bremner [3] provides the following equation to solve for the variables $x_1, ..., x_d$:

$$\sum_{j=1}^{d} \Big(\sum_{\ell=1}^{c}\sum_{m=1}^{c} y_{j\ell}y_{km}F(\ell,m,p) \Big) x_j = y_{kp} \quad (1 \leq k \leq d, 1 \leq p \leq c). \tag{15}$$

Use SageMath to iteratively compute the entries of the matrix by Equation 15. Then use Sage's solver function to return the unique solution as the coordinates of the identity element with respect to the basis Y.

Then then compute a change of basis to Z, as shown in Section 4.2. This algorithm is polynomial in c and d.

5.2.3 The Defining Polynomial

Consider some $v \in I \subseteq Z(\mathcal{A})$. The **defining polynomial** of v is the smallest degree polynomial $f \in \mathbb{Q}[x]$ such that $f(v) = 0$.

We wish to compute the defining polynomial for elements of ideals of the commutative algebra $Z(\mathcal{A})$. To do so, we use the technique from Section 4.4. We start with the generator of the ideal $v \in Z(\mathcal{A})$, and the identity of the ideal $Z(\mathcal{A})v$, both given with respect to the basis $Z = \{z_1, ..., z_c\}$. We want to express e as a linear combination of powers of v. Form the matrix,

$$P = \begin{bmatrix} | & & | & | \\ v^{j-1} & \cdots & v & e \\ | & & | & | \end{bmatrix}.$$

By augmenting powers of v to the left until the next power of v would cause P to be rank deficient. We can compute powers of v using the structure constants in $F \in \mathbb{Q}^{c \times c \times c}$. Compute the reduced row-echelon form of P. Then the first j entries of the last column are the coefficients $c_1, ..., c_j \in \mathbb{Q}$ such that $e = c_j v^{j-1} + \cdots + c_2 v + c_1 e$.

Since augmenting v^j would have made P rank deficient, we have that $v^j = e$. Then,

$$\begin{aligned} 0 &= v^j - e \\ &= v^j - (c_j v^{j-1} + \cdot + c_2 v + c_1 e) \\ \implies m_v(x) &= x^j - (c_j x^{j-1} + \cdots + c_2 x + c_1) \end{aligned}$$

where $m_v(x) \in \mathbb{Q}[x]$ is the defining polynomial of v. Use SageMath to form the matrix, compute the reduced row-echelon form, and define this polynomial in a polynomial ring data structure. The algorithm has running time polynomial in c.

5.3 Idempotents of a Commutative Group Algebra

This algorithm is the crown jewel of our implementation efforts. Up until this point we had only relied on elementary linear algebra operations. Greater functionality is now required and more robust components of SageMath are utilized.

We first present the theory for the algorithm as a whole, then give a formal presentation of the compu-

tation which relies on all of our subroutines from Section 5.2. Throughout this section, we let $Z(\mathcal{A})$ be a commutative group algebra over $\mathbb{Q}$. We let $Z = \{z_1, ..., z_c\}$ be a standard basis with structure constants in $F \in \mathbb{Q}^{c \times c \times c}$.

5.3.1 Theory

An outline of the theory behind this algorithm is necessary for understanding. Our results are from Drozd and Kirichenko [7], Theorem 2.41 in particular. We let $\mathcal{A}$ be a group algebra of dimension n.

We showed in Equation 10 that Φ maps the center of the algebra $\mathcal{A}$ to one-dimensional matrices over $D_1, ..., D_c$. It is the case that $D_i = e_i Z(\mathcal{A})$ is a principle ideal of $Z(\mathcal{A})$.

There turns out to be an even stronger formulation of this idea. Suppose we have elements $e_1, ..., e_c \in Z(\mathcal{A})$ with the following properties:

1. Idempotence: $e_i^2 = e_i$.
2. Pairwise orthogonality: $e_i e_j = 0$ if $i \neq j$.
3. Primitivity: $e_1 + \cdots + e_c = 1 \in Z(\mathcal{A})$.
4. Centrality: $e_1, ..., e_c \in Z(\mathcal{A})$.

We will refer to these elements generally as **idempotents** throughout, with the other properties implicit. Our function Φ can be defined in terms of these elements:

$$\begin{aligned} \Phi(Z(\mathcal{A})) &= D_1 \times \cdots \times D_c \\ \Phi(Z(\mathcal{A})) &= e_1 Z(\mathcal{A}) \times \cdots \times e_c Z(\mathcal{A}) \\ \Phi(z) &= \begin{bmatrix} e_1 z & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & e_c z \end{bmatrix} \end{aligned} \tag{16}$$

for some $z \in Z(\mathcal{A})$. This follows directly from the Wedderburn-Artin Theorem [2][18]. Moreover, for the

Table 1: Subroutines for Ideal Splitting

Name	Inputs	Outputs
IDEALBASIS(v)	v An element of the algebra $Z(\mathcal{A})$.	I: A basis for the ideal $uZ(\mathcal{A})$.
IDEALID(I)	I: A basis for the ideal.	e : The identity element of the ideal I.
MINPOLY(v, I)	I: A basis for the ideal. v: An element of the ideal I.	f: The defining polynomial of $u \in Z(\mathcal{A})$.

entire algebra and some element $a \in \mathcal{A}$ we have,

$$\begin{aligned} \Phi(\mathcal{A}) &= D_1^{d_1 \times d_1} \times \cdots \times D_c^{d_c \times d_c} \\ \Phi(\mathcal{A}) &= e_1 \mathcal{A} e_1 \times \cdots \times e_c \mathcal{A} e_c. \\ \Phi(a) &= \begin{bmatrix} e_1 a e_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & e_c a e_c \end{bmatrix}. \end{aligned} \tag{17}$$

Note that these ideals are not principle, but two-sided since $\mathcal{A}$ is not necessarily commutative. This is also directly from the Wedderburn-Artin Theorem.

The burden of computation is then on finding these idempotents. Their most identifiable feature is that, when all the other properties are maintained, they are the identity elements of the principle ideal they generate. To see this, let $e_i \in Z(\mathcal{A})$ be a primitive and orthogonal idempotent. Let $z \in e_i Z(\mathcal{A})$ be an arbitrary element so $z = e_i z'$ for some $z' \in Z(\mathcal{A})$. Then,

$$\begin{aligned} & z = e_i z' \\ \implies & e_i z = e_i^2 z' \\ \implies & e_i z = e_i z' \\ \implies & e_i z = z \end{aligned}$$

by idempotence. The other direction follows from commutativity of $e_i, z \in Z(\mathcal{A})$. By definition of the identity, we have that $e_i \in e_i Z(\mathcal{A})$ is the identity. Then we can locate idempotents by computing identities. We will show in our algorithm that the idempotents will be pairwise orthogonal and primitive.

5.3.2 Implementation

In Section 5.2 we defined three subroutines, all of which will be used here. Names, inputs, and outputs for these subroutines are given in Figure 1.

Algorithm 2 Algorithm for Finding Idempotents

```
1:  function IDMPFINDER(I, E)
2:      𝔽 = ℚ
3:      ext = 1
4:      for v ∈ I do
5:          if v ≠ 0⃗ and IDEALBASIS(v)≠ I then
6:              f =MINPOLY(v, I)
7:              if f.FACTOR(𝔽).LENGTH== 1 then
8:                  ext =PRIMITIVEELEMENT(ext, v)
9:                  𝔽 = ℚ(ext)
10:             else
11:                 for g ∈ f.FACTOR(𝔽) do
12:                     J =IDEALBASIS(g(v))
13:                     E =IDMPFINDER(J, E)
14:                 end for
15:             end if
16:             return E
17:         end if
18:     end for
19:     e =IDEALID(I)
20:     E.APPEND(e)
21:     return E
22: end function
```

We present pseudocode for our idempotent finder in in Algorithm 2. This is conceptually similar to that which was first produced by Friedl and Rònyai [9], but carries the nuance of implementation hurdles that we addressed. The algorithm is called IDMPFINDER(I, E). The parameter I is a basis for the ideal, given with respect to the standard basis Z. The parameter E is a list that maintains the identity elements, or idempotents, that we have found. The initial call will be IDMPFINDER$(Z, [])$ with the standard basis and an empty list, E. We can use the standard basis for $Z(\mathcal{A})$ because we have structure constants in $F \in \mathbb{Q}^{c\times c\times c}$ from Section 5.1.3.

On Line 2, we set the field $\mathbb{F} = \mathbb{Q}$ which is the base field of the ideal $I \subseteq Z(\mathcal{A})$. On Line 3, we set $ext = 1$ which is a trivial extension of $\mathbb{F}$. We will return to this idea on Line 7. Lines 4 and 5 iterate over non-trivial basis elements to see if any of them "split" the current ideal. A trivial element is one that generates the same ideal, or one that generates $\{0\}$. Suppose we have an element v that generates a non-trivial ideal. On Line 6, we compute the defining polynomial of $v \in I$ as $f \in \mathbb{Q}[x]$.

On Lines 7 and 11 we attempt to factor f over $\mathbb{F}$. The simpler case is when $f = gh$ factors over $\mathbb{F}$. Here, $g(v)$ and $h(v)$ generate two mutually exclusive ideals with identities e_g and e_h such that $e_g + e_h = 1 \in I$. So their identities are primitive in I. Since f is a minimal polynomial, $f(v) = g(v)h(v) = 0$ by definition. Then the generators of the ideal, $g(v)$ and $h(v)$, are pairwise orthogonal. We iterate over each factor, g, on Line 11 and find a basis for the ideal generated by $g(v)$ on Line 12 as J. We then recursively call IDMPFINDER(J, E)

on these new ideals.

If f does not factor then it may be the case that the identity element of the ideal we are currently working with is one of the idempotents we are looking for. As we discussed, this idempotent is orthogonal and primitive in the ideals we have descended from. It remains to prove that the ideal cannot be reduced, and thus the idempotent cannot split to obtain idempotents that are primitive in this ideal. We enter the if-clause on Line 7 and extend $\mathbb{F}$ by v on Line 9 using the Primitive Element Theorem. We explain this further below. This procedure is repeated. We extend $\mathbb{F}$ by basis elements of I until f factors or we have processed every basis element of I. In the latter case, $I = \mathbb{F}$ as an extension field and the identity must be primitive, it is already orthogonal. We append the identity of I on Line 20 and return E on line 21.

There are two subroutines to be mentioned that we outsourced for an implemented. The first is PRIMITIVEELEMENT(ext, v) which we call on Line 8. There is a result called the Primitive Element Theorem which states that if α and β are algebraic over $\mathbb{Q}$ then $\mathbb{Q}(\alpha, \beta) = \mathbb{Q}(\gamma)$ where $\gamma = \alpha + k\beta$ for some $k \in \mathbb{Z}$. This reduces the complexity of factoring over $\mathbb{F}$ by reducing compound extensions of $\mathbb{Q}$ to a single element extension. SageMath has an implemented algorithm that finds such a $k \in \mathbb{Z}$ to perform this reduction. While this step is not theoretically necessary, it is helpful in a practical implementation.

The other subroutine is FACTOR($\mathbb{F}$). This computes the factors of a polynomial $f \in \mathbb{F}[x]$ where $\mathbb{F}$ is a finite extension of $\mathbb{Q}$. We call this on Line 7 and 11. Landau [12] and Lenstra [13] gave algorithms for factoring polynomials over the rataionals and over finite extensions of the rationals in polynomial time. We were able to use SageMath's polynomial ring data structure, extension field structure, and factoring algorithms for this.

We showed that each idempotent generates an ideal in Equation 16. We have just shown that every generator is pairwise orthogonal. We have also shown that any ideal we generate will have identities, primitive in the ideal they descended from. Then the primitivity of identity elements descends down the recursive calls until we arrive at the idempotent generators.

It was shown by Friedl and Rònyai [9] in their Theorem 7.6 that this algorithm is correct, and has runtime polynomial in c and K. The variable c is the dimension of the basis Z. The variable K is the bound on the absolute value of a structure constant which we take to be the absolute maximum value of a rational type in SageMath. Then IDMPFINDER($\mathcal{A}, E$) returns a list of elements $e_1, ..., e_c$ with which we can derive Φ as show in Equation 16 and Equation 17.

5.4 Computing Φ

With all of our set-up from the previous sections we have almost enough to compute Φ. We note that this section encroaches on our future work. Section 5.4.1 has been implemented in full. Section 5.4.2 has a partial implementation, the merit of which is discussed in Section 6.1.

We begin with the group algebra $\mathcal{A} = \mathbb{Q}[G]$ of a finite group G. We compute a basis $Z(\mathcal{A})$ and its structure constants. We use our idempotent finding procedure to obtain the elements $e_1, ..., e_c$ from IDEALIDMPS($Z(\mathcal{A}), E$). Each of these idempotents are given with respect to the standard basis of $Z(\mathcal{A})$. Since $Z(\mathcal{A})$ is given with respect to the basis of $\mathcal{A}$, we compute a change of basis as shown in Section 4.2. Then $e_1, ..., e_c$ are expressed over the standard basis of $\mathcal{A}$ with structure constants in $C \in \mathbb{Q}^{n\times n\times n}$.

5.4.1 The Wedderburn components of $\mathcal{A}$

Before computing Φ, we need to compute bases for each of the two-sided ideals $e_1\mathcal{A}e_1, ..., e_c\mathcal{A}e_c$. We consider a fixed idempotent e, given with respect to the standard basis of $\mathcal{A}$. We use the linear system shown in Section 4.3 to find a basis for the ideal $e\mathcal{A}e$. Similarly to generating the basis of $Z(\mathcal{A})$, we are not solving for a particular element but generating many elements. Let e_i denote the i^{th} coordinate of e in this example:

$$\begin{bmatrix} C(1,\infty,\infty) \\ \vdots \\ C(n,\infty,\infty) \end{bmatrix} \begin{bmatrix} e_1 \\ \vdots \\ e_n \end{bmatrix} = \begin{bmatrix} a_1e \\ \vdots \\ a_ne \end{bmatrix}$$

Where a_ie are n-entry ranges of the $1 \times n^2$ product vector. We create a matrix E and store each range as a row. Note that this ideal is two sided, so we also need to generate the basis of $e\mathcal{A}$. We use the same idea as in Section 5.1.2:

$$\begin{bmatrix} C(\infty,1,\infty) \\ \vdots \\ C(\infty,n,\infty) \end{bmatrix} \begin{bmatrix} e_1 \\ \vdots \\ e_n \end{bmatrix} = \begin{bmatrix} ea_1 \\ \vdots \\ ea_n \end{bmatrix}.$$

We obtain store each n-entry range as a row of E once more. We compute the reduced row-echelon form of E to obtain a linearly independent basis of $e\mathcal{A}e$, given with respect to the basis of $\mathcal{A}$.

For each $e_i\mathcal{A}e_i$, the number of basis vectors in the reduced row-echelon form of E is the dimension of $e_i\mathcal{A}e_i = D_i^{d_i\times d_i}$. We can now characterize the Wedderburn components of the group algebra $\mathcal{A}$ by their dimensions. This subroutine has running time polynomial in n, the size of the group G.

5.4.2 Computing the Wedderburn decomposition

This subroutine is the last in our implementation efforts and also represents an area for future work. We present a particular case for which our algorithm was effective. We note that Φ is a linear transformation of the elements of $\mathcal{A}$ over a standard basis. Then Φ is defined by a matrix.

Let $\mathcal{A}$ be the group algebra with basis $B = \{a_1, ..., a_n\}$. Let $e_1, ..., e_c$ be idempotents for $\mathcal{A}$. Consider the case where each idempotent generates a one-dimensional, two-sided ideal of $\mathcal{A}$. That is, $e_i\mathcal{A}e_i$ has e_i as its only basis element for all $1 \leq i \leq c$. Note that $c = n$ in this case. Then our mapping is $\Phi : \mathcal{A} \longrightarrow \mathbb{Q} \times \cdots \times \mathbb{Q}$. It is given by the following, invertible, linear transformation:

$$M^{-1} = \begin{bmatrix} | & & | \\ e_1 & \dots & e_c \\ | & & | \end{bmatrix} \tag{18}$$

$$M = (M^{-1})^{-1}. \tag{19}$$

Equation 18 corresponds to the inverse transformation Φ^{-1}. Equation 19 corresponds to Φ. M expresses elements of $\mathcal{A}$, given by the basis B, with respect to the basis $e_1, ..., e_c$. M^{-1} expresses elements of the Wedderburn space, given by the basis $e_1, ..., e_c$, as elements of $\mathcal{A}$ with respect to the basis B. This is our Wedderburn decomposition.

6 Results

We now synthesize the findings and achievements of this body of work in terms of producing explicit Wedderburn decompositions, and performing Cohn and Umans matrix multiplication. We make note of such results that lead us to places for future work.

6.1 On the Computation of Φ

The goal of this thesis was to advance the Cohn and Umans fast matrix multiplication algorithm by implementing the missing piece: explicit Wedderburn decompositions. I was able to complete this implementation with mixed success.

We defined the function Φ as the linear transformation of elements of the algebra $\mathbb{Q}[G]$ to the Wedderburn decomposition space in a particular case. We were was able to compute Φ for a number of group algebras when the Wedderburn components were only rational fields. The running time of this algorithm

is polynomial in the size of the group. For algebras that map to matrices of higher dimensions, or to extensions of the rationals, there is an unimplemented step. This is known as the explicit isomorphism problem. The computational complexity of defining an explicit isomorphism was shown to be equivalent to integer factorization by Rònyai [15].

However, we were able to find the primitive, central, orthogonal idempotents for a number of algebras which is non-trivial. With these elements we determined the dimensions of the Wedderburn components of many group algebras. This functionality is provided by the Wedderga [19] library of GAP [10], but with an entirely different implementation. From what I have seen, my implementation seems to be a novelty for performing this computation.

6.2 Generalized Applications

My implementation of the Wedderburn-Artin Theorem is for a specific application in fast matrix mutliplication algorithms. My aim was to produce an algorithm that calculates the explicit Wedderburn decompositions of the finite dimensional group algebra $\mathbb{Q}[G]$. It turns out that these algorithms are more general.

In the context of pure mathematics, the Wedderburn-Artin Theorem applies to a more broad class of structures called **semisimple** algebras. The algebra $\mathbb{Q}[G]$ of a finite group is always semisimple because $\mathbb{Q}$ has characteristic 0. However, the algebra $\mathbb{F}[G]$ for some other field or ring $\mathbb{F}$ may also be semisimple. If we have a representation for $\mathbb{F}$, we can also compute Wedderburn decompositions of the algebra $\mathbb{F}[G]$ using the same ideas in these algorithms with slight conceptual changes.

In an even further extension, Bremner [3] generalizes the notion of a Wedderburn decomposition to algebras that are not semisimple. Our implementation efforts were guided by Bremner's example of decomposition the partial transformation semigroup $\mathbb{Q}[PT_2]$. This finite dimensional algebra is not semisimple, but we have implemented algorithms that restrict this algebra to its semisimple part. We were able to determine the dimensions of the Wedderburn components of this algebra.

6.3 Generalized Implementations

In Section 5.1 we described an algorithm that computes the center of the group algebra $\mathbb{Q}[G]$. This algorithm turns out to be more general. We can actually compute the center of any finite dimensional, associative, and semisimple algebra. All we need is a data structure for the base field and structure constants for the algebra. No conceptual changes are necessary.

The splitting procedure detailed in Section 5.3 maintains the same generality. We can decompose an algebra that is finite dimensional, associative, and semisimple provided a representation for the field. In

the case of finite fields the algorithm actually becomes simpler, an outline is given by Friedl and Rònyai [9].

If we wish to decompose algebras that are not semisimple there is some prep work to be done. In the case of Bremner's example with $\mathbb{Q}[PT_2]$, we had to find a particular subset of the algebra called **the radical**. We then had to restrict the algebra to its **quotient** of the algebra over the radical. Algorithms for the computation of the radical of an associative algebra and the quotient of the algebra over the radical are implemented in my codebase.

These are complex ideas, but their computation produces something of a subspace of the algebra that is semisimple. We can then apply all of our algorithms in the usual way to obtain the idempotents, but the computation of the Φ function requires an additional step that is beyond the scope of this paper.

7 Future Work

There are many lines of continuation for this thesis topic with two main directions. The first is advancing the Cohn and Umans framework. These algorithms can currently produce the mapping, guaranteed by the Wedderburn-Artin Theorem, for some algebras of the rationals over a finite group. This is the case where the Wedderburn components are all one-dimensional matrices over the rationals. In order to obtain the $\omega = 2.41$ upper bound from Cohn and Umans [5], we will need to extend the algorithm to represent Wedderburn components of cyclotomic fields and matrices of dimension greater than 1. This is the codomain for the particular group that gives rise to the $\omega = 2.41$ bound. We need an algorithm to solve the so-called explicit isomorphism problem which was shown to be in **NP** by Rònyai [15].

The other place for advancement is in pure mathematics. My algorithm is able to determine the Wedderburn components of many group algebras. I have implemented two additional subroutines, mentioned briefly in Section 6.2, that have been used to determine the Wedderburn components of an algebra of the rationals over a finite semigroup. This functionality is not provided by Wedderga [19], and would represent an advancement in the classification of finite-dimensional associative algebras. Further, integrating the algorithms presented in this paper with a computational discreet algebra system would allow us utilize the tools of a more robust system and advance the capabilities, as well as the efficiency, of our developments.

References

[1] Matthew Anderson, Zongliang Ji, and Anthony Yang Xu. "Matrix Multiplication: Verifying Strong Uniquely Solvable Puzzles". In: *Theory and Applications of Satisfiability Testing – SAT 2020*. Ed. by Luca Pulina and Martina Seidl. Cham: Springer International Publishing, 2020, pp. 464–480.

[2] Emil Artin. “Zur Theorie der hyperkomplexen Zahlen”. In: *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg* 5.1 (Jan. 1927), pp. 251–260. ISSN: 1865-8784. DOI: 10.1007/BF02952526. URL: https://doi.org/10.1007/BF02952526.

[3] Murray R Bremner. “How to compute the Wedderburn decomposition of a finite-dimensional associative algebra”. In: *Groups Complexity Cryptology* 3 (1 2011), pp. 47–66. DOI: https://doi.org/10.1515/gcc.2011.003. URL: https://www.degruyter.com/document/doi/10.1515/gcc.2011.003/html#Harvard.

[4] H. Cohn and C. Umans. “A group-theoretic approach to fast matrix multiplication”. In: *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science* (Oct. 2003), pp. 438–449. DOI: 10.1109/sfcs.2003.1238217. URL: http://dx.doi.org/10.1109/SFCS.2003.1238217.

[5] H. Cohn et al. “Group-theoretic algorithms for matrix multiplication”. In: *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*. 2005, pp. 379–388. DOI: 10.1109/SFCS.2005.39.

[6] Don Coppersmith and Shmuel Winograd. “Matrix multiplication via arithmetic progressions”. In: *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. 1987, pp. 1–6.

[7] Yuri A. Drozd and V. V. Kirichenko. “Finite dimensional algebras”. In: Berlin, Heidelberg: Springer Berlin, Heidelberg, 2012, pp. 1–81. ISBN: 978-3-642-76244-4. DOI: https://doi.org/10.1007/978-3-642-76244-4. URL: https://link.springer.com/book/10.1007/978-3-642-76244-4#bibliographic-information.

[8] David S Dummit and Richard M Foote. *Abstract algebra*. Vol. 1999. Prentice Hall Englewood Cliffs, NJ, 1991.

[9] K. Friedl and L. Rònyai. “Polynomial Time Solutions of Some Problems of Computational Algebra”. In: STOC '85. Providence, Rhode Island, USA: Association for Computing Machinery, 1985, 153â162. ISBN: 0897911512. DOI: 10.1145/22145.22162. URL: https://doi.org/10.1145/22145.22162.

[10] *GAP – Groups, Algorithms, and Programming, Version 4.11.1*. The GAP Group. 2021. URL: www.gap-system.org.

[11] G. Ivanyos and L. Rònyai. “Computations in Associative and Lie Algebras”. In: *Some Tapas of Computer Algebra*. Ed. by Arjeh M. Cohen, Hans Cuypers, and Hans Sterk. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 91–120. ISBN: 978-3-662-03891-8. DOI: 10.1007/978-3-662-03891-8_5. URL: https://doi.org/10.1007/978-3-662-03891-8_5.

[12] Susan Landau. "Factoring Polynomials over Algebraic Number Fields is in Polynomial Time". In: *SIAM Journal on Computing* 14.1 (1985), pp. 184–195. DOI: 10.1137/0214015. eprint: https://doi.org/10.1137/0214015. URL: https://doi.org/10.1137/0214015.

[13] A. K. Lenstra. "Factoring polynomials over algebraic number fields". In: *Computer Algebra*. Ed. by J. A. van Hulzen. Berlin, Heidelberg: Springer Berlin Heidelberg, 1983, pp. 245–254. ISBN: 978-3-540-38756-5.

[14] Aaron Meurer et al. "SymPy: symbolic computing in Python". In: *PeerJ Computer Science* 3 (Jan. 2017), e103. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.103. URL: https://doi.org/10.7717/peerj-cs.103.

[15] L. Rònyai. "Simple Algebras Are Difficult". In: STOC '87. New York, New York, USA: Association for Computing Machinery, 1987, 398â408. ISBN: 0897912217. DOI: 10.1145/28395.28438. URL: https://doi.org/10.1145/28395.28438.

[16] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 9.8)*. 2023. URL: https://www.sagemath.org.

[17] Volker Strassen et al. "Gaussian elimination is not optimal". In: *Numerische mathematik* 13.4 (1969), pp. 354–356.

[18] J. H. MacLagan Wedderburn. "On Hypercomplex Numbers". In: *Proceedings of the London Mathematical Society* s2-6.1 (Jan. 1908), pp. 77–118. ISSN: 0024-6115. DOI: 10.1112/plms/s2-6.1.77. eprint: https://academic.oup.com/plms/article-pdf/s2-6/1/77/4301463/s2-6-1-77.pdf. URL: https://doi.org/10.1112/plms/s2-6.1.77.

[19] G. K. Bakshi, O. Broche Cristo, A. Herman, A. Konovalov, S. Maheshwary, A. Olivieri, G. Olteanu, A. del Rio and I. Van Gelder. *Wedderga — Wedderburn Decomposition of Group Algebras*. 2020. URL: www.cs.st-andrews.ac.uk/~alexk/wedderga.

[20] Toshihiko Yamada. "The Schur subgroup of the Brauer group. I". In: *Journal of Algebra* 27.3 (1973), pp. 579–589. ISSN: 0021-8693. DOI: https://doi.org/10.1016/0021-8693(73)90066-5. URL: https://www.sciencedirect.com/science/article/pii/0021869373900665.