Recursive Constructions of Arrays With Low Autocorrelation

Nathan Jolly
BSc (Hons)

School of Mathematical Sciences
Monash University

Submitted in total fulfilment of the requirements
of the degree of Doctor of Philosophy

March 10, 2015

Acknowledgements

I thank my three PhD supervisors: **Tom Hall** supported me for three years, helped me to learn the material, and stood up for my results in the times when I didn't think I had enough; **Heiko Dietrich** supported me both mathematically and financially in my final year, and he helped in many ways to make much of my later work infinitely easier; **Imants Svalbe** has been a good sounding board and source of encouragement throughout the whole time.

I am also grateful to the university and governmental funding system which allowed me to be supported financially for four years.

Abstract

Matrices with low autocorrelation are very useful in applications such as digital watermarking. This thesis introduces a new class of matrices with low autocorrelation, and presents a new method of constructing such matrices to unbounded sizes.

In 2001, Arasu and de Launey [1] presented several recursive constructions of perfect and what they call "almost perfect" arrays over the quaternary (4th roots of unity) alphabet. "Recursive" means that under certain conditions, the construction can be applied repeatedly, thereby generating larger and larger arrays while preserving low autocorrelation.

Arasu and de Launey employed Laurent polynomials over two indeterminates to state and prove the recursive constructions. While this policy made the proofs expedient, it also obfuscated the simple yet powerful nature of the constructions themselves.

In this thesis, we present some of those recursive constructions in the language of arrays, with no recourse to polynomials. This research direction has led to three new contributions:

1. A translation of an algorithm by Arasu and de Launey to "inflate" perfect quaternary arrays, leading to joint work with Barerra Acevedo giving infinitely large perfect arrays over the basic quaternions;

2. A new algebra of arrays involving array convolution as the multiplicative operation as well as special functions called "index functions", culminating in a simpler statement and proof of Arasu and de Launey's recursive construction of almost perfect arrays;

3. A new recursive construction of what we call "n-perfect" arrays, which extends both the concept and recursive construction of Arasu and de Launey's almost perfect arrays.

Declaration

This is to certify that:

- this thesis comprises only my original work towards the PhD except where indicated in the Preface;
- due acknowledgement has been made in the text to all other material used;
- this thesis is fewer than one hundred thousand words in length, exclusive of tables, maps, bibliographies and appendices.

Nathan Jolly

Preface

All work towards this thesis was carried out during the period of PhD candidature at Monash University, Clayton Campus. None of the work has been submitted for any other qualification. Except for the results of other authors who are acknowledged as they are introduced, the results of Chapters 5 through 7 are to the best of my knowledge original contributions, with one exception: The mathematics of Section 5.5 was due to our colleague Barerra Acevedo, and was taken from our joint paper [4]. What appears in Section 5.5 is our own rendering of our Barerra Acevedo's work. The quaternion inflation example in Section 5.6 has been adapted from the example given in [4]. The results of Chapter 6 appear in [13].

Contents

Chapter 1

Introduction and Historical Background

1.1 Introduction

This work gives a recursive construction yielding ***n*-perfect arrays of unbounded sizes** over the complex numbers. This work also gives a new **algebra of arrays** which can be used to state and prove array constructions which preserve good correlation. Finally, this work, in conjunction with work by our colleague Barerra Acevedo, shows the existence of **perfect arrays of unbounded sizes** over the basic quaternions $\{\pm 1, \pm \mathrm{i}, \pm \mathrm{j}, \pm \mathrm{k}\}$.

In 2001, Arasu and de Launey [1] presented several recursive constructions of perfect and what they call "almost perfect" arrays over the quaternary (4th roots of unity) alphabet. "Recursive" means that under certain conditions, the construction can be applied repeatedly, thereby generating larger and larger arrays while preserving low autocorrelation.

Laurent polynomials over two indeterminates were employed in [1] to state and prove the recursive constructions. While this policy made the proofs expedi-

ent, it also obfuscated the simple yet powerful nature of the constructions themselves. It is one aim of this thesis to re-formulate these constructions in a rigorous algebraic set-up.

We worked on translating the concepts and constructions of Arasu and de Launey's work which was taken out of the language of polynomials and into the natural language of arrays (matrices) and sequences (tuples). Our first result, appearing in Chapter 5, was a translation of Arasu and de Launey's "inflation of perfect arrays" [1]. We were able to translate their proposed construction to our new theoretical framework, using only basic definitions of array autocorrelation. This allowed us to extend the algorithm to the basic quaternions $\mathbb{H}_8 = \{\pm 1, \pm \mathrm{i}, \pm \mathrm{j}, \pm \mathrm{k}\}$, which, when combined with **Lee sequences** [15], yields **perfect arrays of unbounded sizes** over $\mathbb{H}_8$.

We also noted that polynomial multiplication corresponds directly to array convolution, and, since the set $\mathbb{C}[x, y, x^{-1}, y^{-1}]$ of Laurent polynomials forms a $\mathbb{C}$-algebra, we translated this to a new $\mathbb{C}$-algebra where the underlying set consists of $m \times n$ arrays over the complex numbers, the additive operation is ordinary array addition, and the multiplicative operation is array convolution. We also introduced the concept of **index functions**, which allow arbitrary rearrangements of array elements to be expressed—which would be difficult to do using polynomial theory. The result is a new **algebra of arrays**, and is presented in Chapter 6. In Section 6.6, we state and prove—using our algebra—a recursive construction of Arasu and de Launey's almost perfect arrays [1, Lemma 25]. (Here "algebra" refers to the usual algebraic object with the same name, namely, a ring with a compatible structure of a vector space.) This restatement makes it clear that Arasu and de Launey's construction uses only basic array techniques like concatenation, column shifting, and interleaving. Apart from making the construction easier to work with, it also makes it clear that the underlying alphabet of the arrays involved is

preserved, which is not immediate from the polynomial presentation.

Finally, we extended the concept of Arasu and de Launey's almost perfect arrays (whose $m \times n$ autocorrelation array is all zeroes except for $+mn$ in the $(0,0)$-position and $-mn$ halfway down the first column) to nth roots of unity. Specifically, we defined an **n-perfect array** to be one whose $m \times n$ autocorrelation array is zeroes everywhere, except for the first column, where the nth roots of unity appear in sequence and multiplied by mn, with zeroes in between. (Almost perfect arrays become "2-perfect" under this regime.) These arrays are defined formally in Section 7.3 where examples are given. Next, we used the simple concepts of concatenation, column shifting, and interleaving to create a new recursive construction similar to [1, Lemma 25], thereby yielding a construction of n-perfect arrays over the complex numbers whose dimensions are strictly increasing.

1.2 Summary

The present work consists of 8 chapters.

- In Chapter 1, Introduction and Historical Background, a brief overview of the material in the thesis is given, as well as an overview of the material in [1], which most of the work in the thesis is based on. We also give some historical background on sequences, arrays and watermarking.

Background and Preliminaries

- In Chapter 2, Arrays and Sequences, some general definitions about arrays and sequences are given, as well as notation which is used throughout.

- In Chapter 3, Correlation and Convolution, we define auto- and cross-correlation as well as convolution of arrays and sequences, and we use these concepts to

define perfect arrays, as well as Arasu and de Launey's "almost perfect" arrays. We also define perfect autocorrelation of arrays over the quaternions.

- In Chapter 4, Sequences and Arrays With Low Autocorrelation, we give some of the well-known examples of low autocorrelation sequences and arrays from the literature, and we demonstrate the application of low-autocorrelation arrays by giving an example watermark.

Chapters 1 and 4 can be considered the literature review, and Chapters 2 and 3 provide necessary background for understanding the material that follows.

Main Contributions

- In Chapter 5, Inflation of Perfect Complex and Quaternion Arrays, we present the inflation of perfect quaternary arrays of Arasu and de Launey [1] in the language of arrays, rather than the original polynomial version, and we generalise the arrays to the complex alphabet. This leads to our publication [4], where we present inflation of arrays over the pure quaternions.

- In Chapter 6, An Algebra of Arrays, we present a new way of manipulating arrays algebraically as whole objects, without using the polynomial method of [1]. We then use this algebra to state and prove one of the constructions of almost perfect arrays in [1] in a much more direct way than polynomials allow.

- In Chapter 7, A Recursive Construction of n-Perfect Arrays, we present a new recursive construction of what we call "n-perfect" arrays, which are certain generalisations of almost perfect arrays. Our construction provides a natural generalisation of a recursive construction for almost perfect arrays given by Arasu and de Launey.

- In Chapter 8, Conclusion.

1.3 Historical Background

Sequences with good correlational properties have been researched for over 60 years, and arrays with good correlational properties (introduced by Calabro and Wolf [6] in 1968) have been researched for over 40 years.

Sequences and arrays are very interconnected. Arrays with good autocorrelational properties can be constructed by making their columns various shifts of a single sequence. (See, for example, Tirkel and Hall [25].) In 2001, Hall et al. [10] created families of arrays using quadratic shifts of a *Legendre sequence* (which we define in Section 4.2). Each array in the family has good (low) off-peak autocorrelation, and distinct arrays in the family have low cross-correlation with each other. We discuss these arrays in Section 4.4.

Earlier work on constructing arrays out of sequences has been done by Baumert [5], who folded certain M-sequences (discussed in Section 4.3) into the rows of an array, giving the array good autocorrelational properties. Games [7] did similar work in 1985 on folding M-sequences into rows of an array. In 1971, Weng [27] showed how to fold M-sequences diagonally into arrays with coprime dimensions.

In the other direction, sequences with good correlational properties can be constructed out of the rows and columns of arrays. In 1961, Heimiller [12] gave a construction of perfect sequences of length p^2 (p an odd prime) by unfolding a special $p \times p$ array, whose entries are pth roots of unity.

The most prominent use of arrays with good correlational properties is digital watermarking, which we discuss in Section 1.4. Other uses of low autocorrelation arrays are frequency-hopping patterns for radar and sonar communications [19, 21], time-hopping patterns for UWB radio [22], and two-dimensional optical

orthogonal coding [23].

1.4 Digital Watermarking

Digital watermarking is one of the most prominent applications of arrays with good correlation. A digital watermark is an imperceptible marker embedded into noise-tolerant data, such as an audio, video or image file. It can be used to identify copyright ownership of the file, for example to distinguish between an official version and a pirated copy of a file. Watermarks can also be used to store personal data in a file. For example, the personal details of a patient can be stored as a watermark somewhere inside the patient's CAT scan.

"Watermarking" is the process of embedding a watermark into a digital file. A watermark differs from metadata in that it is not appended to the file; it is actually encoded into the data of the file itself. This means that the data is actually changed by the watermark. If a watermark changes the data too much, then it can be perceived and then tampered with, and/or it can ruin the data, making the audio, video or image unpleasant to view or hear. Thus watermarks must be designed such that they do not change the data too much.

A major problem with watermarks is that they can often be easily removed. For example, if an image file is watermarked it is often sufficient to simply photocopy the image and then rescan it to remove the watermark. Another method is to rotate the image to the left or right by a small amount, and then back. Because the trigonometric process of rotation does not work smoothly with the individual pixels of an image, a subtly embedded watermark can be easily removed. A solution is to make the watermark much less subtle, so that rotation and photocopying would not remove it. The problem with this is that the watermark would be easily visible in the image, which defeats the purpose of storing the data imperceptibly.

Thus watermarking is always a trade-off between robustness and imperceptibility. For some applications imperceptibility is more important. For example, a file may be watermarked with a piece of information which is useless in itself and only exists to determine whether the file has been tampered with: if the watermark has been removed then the absence of the information will make make it known that the file was tampered with. In other applications robustness is more important, especially when one is only wishing to store data, rather than hide it. More information on the concepts and process of watermarking can be found in Hartung and Kutter [11].

A digital image can be viewed as a rectangular array of pixels, where each pixel can take a range of values. An example is a grayscale image, whose pixels typically use values from 0 to 255, where 0 is black, 255 is white and the values in between represent shades of gray. An alteration of a digital image entails changing the values of the pixels in the array. This is done by adding the entries of a low autocorrelation array—called a *watermark*—to the entries of the image. If the pixel values are changed too much, then the image will look different, and the alteration becomes obvious. Thus a digital watermark must change the pixel values in an image, but only by a small amount. Watermarking was introduced by A. Z. Tirkel et al. in 1993 [24].

In Section 4.5, we use a low autocorrelation array from [10] to watermark a 7×7 sample image.

1.5 Summary of Arasu and de Launey's Work

Arasu and de Launey [1] used the arithmetic of Laurent polynomials to propose recursive constructions of perfect quaternary arrays (which they call PQAs) and almost perfect quaternary arrays (APQAs). Under certain conditions, these recur-

sive constructions can be continually applied, yielding infinite families of PQAs and APQAs.

Recall that a *Laurent polynomial* (in the two indeterminates x and y and over the complex numbers $\mathbb{C}$) is an expression of the form

$$A(x,y) = \sum_{i,j\in\mathbb{Z}} a_{i,j} x^i y^j$$

where the coefficients $a_{i,j}$ are complex numbers and only finitely many of them are nonzero. The set of Laurent polynomials is denoted $\mathbb{C}[x, y, x^{-1}, y^{-1}]$; this differs from the set $\mathbb{C}[x, y]$ of polynomials with complex coefficients only in that the indeterminates are now allowed to have negative exponents.

The set of Laurent polynomials is a $\mathbb{C}$-algebra under the following operations of addition and multiplication:

$$\sum_{i,j\in\mathbb{Z}} a_{i,j} x^i y^j + \sum_{i,j\in\mathbb{Z}} b_{i,j} x^i y^j = \sum_{i,j\in\mathbb{Z}} (a_{i,j} + b_{i,j}) x^i y^j;$$
$$\left(\sum_{i,j\in\mathbb{Z}} a_{i,j} x^i y^j\right)\left(\sum_{i,j\in\mathbb{Z}} b_{i,j} x^i y^j\right) = \sum_{k,l\in\mathbb{Z}} \left(\sum_{i,j\in\mathbb{Z}} a_{i,j} b_{k-i,l-j}\right) x^k y^l.$$

If

$$A(x,y) = \sum_{i,j\in\mathbb{Z}} a_{i,j} x^i y^j$$

is a Laurent polynomial, reducing it modulo $x^m - 1, y^n - 1$ yields an ordinary polynomial in $\mathbb{C}[x, y]$ with exactly mn coefficients:

$$A(x,y) \equiv \sum_{k=0}^{m-1} \sum_{l=0}^{n-1} a'_{k,l} x^k y^l \pmod{x^m - 1, y^n - 1}$$

where

$$a'_{k,l} = \sum_{i\in k+m\mathbb{Z}} \sum_{j\in l+n\mathbb{Z}} a_{i,j}.$$

Clearly, when working modulo $x^m - 1, y^n - 1$, one can always consider the exponents of x modulo m and the exponents of y modulo n. For example,

$$4x^3y^{-2} + x^6y^4 \equiv 4x^0y^4 + x^0y^4 \equiv 5y^4 \pmod{x^3 - 1, y^6 - 1}.$$

Thus every Laurent polynomial taken modulo $x^m - 1, y^n - 1$ defines a unique $m \times n$ array by writing its coefficients in an $m \times n$ matrix. (See [1] for more details.) It turns out that addition of Laurent polynomials corresponds to addition of arrays, and multiplication of Laurent polynomials corresponds to convolution of arrays. Furthermore, multiplying a Laurent polynomial $A(x, y)$ by the monomial $x^u y^v$ has the effect of cyclically shifting the array represented by $A(x, y)$ down by u places and right by v places, and reducing $A(x^{-1}, y^{-1})$ modulo $x^m - 1, y^n - 1$ has the effect of reversing the array represented by $A(x, y)$ in both the horizontal and vertical directions. With the above operations, as well as a few more, it follows that polynomial algebra can be used to manipulate arrays in a basic way.

In Sections 3.1 and 6.3 we show that auto- and cross-correlation can be recast as special types of convolutions. In polynomial terms, it turns out that if $A(x, y)$ and $B(x, y)$ represent $m \times n$ arrays A and B respectively, and $*$ represents complex conjugation, then $A(x^{-1}, y^{-1})B^*(x, y)$ represents the left-shifted cross-correlation array of A and B. (The right-shifted version is represented by $A(x, y)B^*(x^{-1}, y^{-1})$.)

Laurent polynomials therefore allow us to express all the basic manipulations of arrays, including auto- and cross-correlation. Arasu and de Launey use this method throughout [1] to state and prove their array constructions.

One drawback of this approach is that it is hard to state in a theorem exactly what an array construction entails. Many of the theorems in [1] state the existence of certain arrays, but the details of the construction are hidden in the proof. We have worked to obviate some of those constructions, and state them in simple terms as theorems. For example, in Section 6.6 we give a version of [1, Lemma 25]

(a recursive construction of almost perfect arrays) in array terms, rather than polynomial terms. In our version, the construction uses only the relatively simple array operations of concatenation (defined in Definition 2.9), column-shifting (Definition 2.16) and interleaving (Definition 6.15). In contrast, the version in [1, Lemma 25] does not state the construction in the theorem, and one must "translate" the polynomial algebra in the proof to glean the construction. Thus, while Laurent polynomials can provide easy proofs of an array construction, they also obfuscate the underlying simplicity of the construction, making the results both harder to use and extend. As a case in point, it is only through our simplified version of [1, Lemma 25] that we were able to extend Arasu and de Launey's work and create our recursive construction of n-perfect arrays (which we give in Chapter 7).

It is one of the goals of this thesis to show that many of the ideas in [1] are simpler than they look in the paper, and that Laurent polynomials are not necessary to state and prove some quite powerful array constructions.

Part I

Background and Preliminaries

Chapter 2

Arrays and Sequences

2.1 Basic Definitions

Definition 2.1. *An $m \times n$* array *is an $m \times n$ matrix of complex numbers. If A is an $m \times n$ array, we write $A[i, j]$ for the (i, j)-entry of A, where $0 \leq i \leq m - 1$ and $0 \leq j \leq n - 1$.*

We adopt the convention that

$$A[u, v] = A[u \bmod m, v \bmod n]$$

for all $u, v \in \mathbb{Z}$. This makes the array A periodic, with a period of m vertically and a period of n horizontally.

The theory of arrays can be developed for arbitrary complex entries. However, in applications such as watermarking it is useful to restrict the entries of an array to a finite subset of the complex numbers.

Definition 2.2. *The* alphabet *of an array is the subset of $\mathbb{C}$ where all the entries of the array come from. If an array has no specific alphabet, then it is called a* complex array *or an* array over the complex numbers.

Definition 2.3. *An array is called* binary *if its alphabet is* $\{\pm 1\}$*,* ternary *if its alphabet is* $\{0, \pm 1\}$*, and* quaternary *if its alphabet is* $\{\pm 1, \pm \mathrm{i}\}$ *where* $\mathrm{i}^2 = -1$.

Example 2.4. *The arrays*

$$A = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \qquad C = \begin{bmatrix} 1 & \mathrm{i} \\ -1 & -\mathrm{i} \end{bmatrix}$$

are examples of binary, ternary and quaternary arrays respectively.

Sometimes the binary and ternary alphabets are written as $\{+, -\}$ and $\{+, -, 0\}$ instead of $\{+1, -1\}$ and $\{+1, -1, 0\}$ respectively. In this case, the binary and ternary arrays in Example 2.4 above become

$$A = \begin{bmatrix} + & + \\ - & + \end{bmatrix} \qquad \text{and} \qquad B = \begin{bmatrix} 0 & - \\ + & 0 \end{bmatrix}.$$

We will use ± 1 instead of $+$ and $-$ in this thesis.

Alphabets usually consist of roots of unity, with the binary (2nd roots) and quaternary (4th roots) alphabets being the most widely used. Sometimes 0 is added to the alphabet, as in the ternary case. We now introduce the basic algebraic manipulation of arrays.

Definition 2.5. *The* complex conjugate A^* *of an array* A *has entries* $A[i, j]^*$*, where* $\cdot^*$ *denotes the usual complex conjugation.*

Example 2.6. *The complex conjugate of the array*

$$A = \begin{bmatrix} 1 & \mathrm{i} \\ -1 & -\mathrm{i} \end{bmatrix} \qquad \textit{is} \qquad A^* = \begin{bmatrix} 1 & -\mathrm{i} \\ -1 & \mathrm{i} \end{bmatrix}.$$

It is immediate that $(A + B)^* = A^* + B^*$ and that $(A^*)^* = A$ for all $m \times n$ arrays A and B.

Definition 2.7. *The* dot product *of two $m \times n$ arrays A and B is*

$$A \bullet B = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} A[i, j]B[i, j] \in \mathbb{C}.$$

Example 2.8. *If*

$$A = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \qquad \textit{and} \qquad B = \begin{bmatrix} 1 & \mathrm{i} \\ -1 & -\mathrm{i} \end{bmatrix}$$

then the dot product is

$$A \bullet B = 0 \cdot 1 + (-1) \cdot \mathrm{i} + 1 \cdot (-1) + 0 \cdot (-\mathrm{i}) = -1 - \mathrm{i}.$$

It is easily proven from the definition that the dot product is commutative and distributive over addition of arrays. Also, it is clear that $(A \bullet B)^* = A^* \bullet B^*$.

Dot products of arrays can be decomposed into sums of dot products of rows or of columns. If we denote the ith row of A by $A[i, \cdot]$ and the jth column of A by $A[\cdot, j]$, we can write

$$\begin{aligned} A \bullet B &= \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} A[i, j]B[i, j] \\ &= \sum_{i=0}^{m-1} \left(\sum_{j=0}^{n-1} A[i, j]B[i, j] \right) \\ &= \sum_{i=0}^{m-1} A[i, \cdot] \bullet B[i, \cdot] \end{aligned}$$

so that $A \bullet B$ becomes a sum of m dot products of rows of A with rows of B.

Similarly, we can also write

$$\begin{aligned} A \bullet B &= \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} A[i, j]B[i, j] \\ &= \sum_{j=0}^{n-1} \left(\sum_{i=0}^{m-1} A[i, j]B[i, j] \right) \\ &= \sum_{j=0}^{n-1} A[\cdot, j] \bullet B[\cdot, j] \end{aligned}$$

so that $A \bullet B$ becomes a sum of n dot products of columns of A with columns of B. The two interpretations of $A \bullet B$ are interchangeable.

2.2 Concatenation, Cyclic Shifting, Reversal and Column Shifting

Informally, array concatenation is the formation of a new array by joining several copies of an array either vertically or horizontally.

Definition 2.9. *Let A be a $m \times n$ array, and let $k \geq 1$. The* horizontal concatenation *of k copies of A is the $m \times kn$ array whose (i, j)-entry is $A[i, j \bmod n]$. The* vertical concatenation *of k copies of A is the $km \times n$ array whose (i, j)-entry is $A[i \bmod m, j]$.*

Example 2.10. *Let*

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

Then the horizontal concatenation of 3 *copies of A is*

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

and the vertical concatenation of 2 *copies of A is*

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

Informally, the cyclic shift of an array is a rotation of its elements in both the vertical and horizontal directions. The following definition makes use of the periodicity of arrays that we introduced in Section 2.1.

Definition 2.11. *For* $u, v \in \mathbb{Z}$*, the* (u, v)-shift *of an* $m \times n$ *array* A *is the* $m \times n$ *array whose* (i, j)*-entry is* $A[i + u, j + v]$*. We denote this array by* $A^{s_{u,v}}$*.*

Example 2.12. *If*

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix},$$

then the $(1, 2)$*-shift of* A *is*

$$A^{s_{1,2}} = \begin{bmatrix} 6 & 4 & 5 \\ 9 & 7 & 8 \\ 3 & 1 & 2 \end{bmatrix}.$$

It can be seen that $A^{s_{1,2}}$ *is* A *with its entries rotated one place in the vertical direction and two places in the horizontal direction.*

It is easy to verify that the order of vertical and horizontal shifting does not matter. In symbols, this means that $(A^{s_{u,0}})^{s_{0,v}} = (A^{s_{0,v}})^{s_{u,0}} = A^{s_{u,v}}$. It is also easy to verify that $(A^{s_{u,v}})^{s_{x,y}} = A^{s_{u+x,v+y}}$.

Definition 2.13. *The* reversal *of an* $m \times n$ *array* A *is the* $m \times n$ *array whose* (i, j)*-entry is* $A[-i, -j]$*. We denote this array by* A^r*.*

Example 2.14. *If*

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix},$$

then the reversal of A is

$$A^r = \begin{bmatrix} 1 & 3 & 2 \\ 7 & 9 & 8 \\ 4 & 6 & 5 \end{bmatrix}.$$

The following proposition is immediate.

Proposition 2.15. *The* $(0,0)$*-entry of an array is the only entry that is fixed by a reversal. It is easy to verify that* $(A^r)^r = A$ *and that* $(A^{s_{u,v}})^r = (A^r)^{s_{-u,-v}}$*. It is also easy to verify that* $(A^*)^r = (A^r)^*$ *and that* $(A^*)^{s_{u,v}} = (A^{s_{u,v}})^*$*.*

Definition 2.16. *For* $u \in \mathbb{Z}$*, the* u-column shift *of an* $m \times n$ *array* A *is the* $m \times n$ *array whose* (i, j)*-entry is* $A[i + uj, j]$*. We denote this array by* A^{c_u}*.*

If

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 \end{bmatrix}$$

then

$$A^{c_1} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 1 \\ 3 & 4 & 5 & 1 & 2 \\ 4 & 5 & 1 & 2 & 3 \\ 5 & 1 & 2 & 3 & 4 \end{bmatrix} \quad \text{and} \quad A^{c_2} = \begin{bmatrix} 1 & 3 & 5 & 2 & 4 \\ 2 & 4 & 1 & 3 & 5 \\ 3 & 5 & 2 & 4 & 1 \\ 4 & 1 & 3 & 5 & 2 \\ 5 & 2 & 4 & 1 & 3 \end{bmatrix}.$$

In words, A^{c_u} is the array A with the jth column ($j \in \mathbb{Z}_n$) shifted uj places up.

2.3 Sequences

Definition 2.17. *A* sequence *is a one-dimensional array. The* length *of the sequence is the number of elements in the sequence. Thus, a sequence of length n can be either a* $1 \times n$ *or* $n \times 1$ *array.*

A sequence of length n is typically written $A = (A[0], \ldots, A[n-1])$, where the $A[i]$ are the elements of the sequence. The same periodic convention

$$A[u] = A[u \bmod n] \qquad (u \in \mathbb{Z})$$

applies for sequences as well as arrays.

All the operations for arrays apply for sequences as well, and they are simpler because there is only one dimension considered. For example, the dot product of two length n sequences A and B is

$$A \bullet B = \sum_{i=0}^{n-1} A[i]B[i].$$

The (u, v)-shift for arrays becomes a u-shift for sequences, and is defined by

$$A^{s_u}[i] = A[i + u].$$

Similarly, the reversal of a sequence is defined by

$$A^{r}[i] = A[-i].$$

2.4 Higher-Dimensional Arrays

Arrays can extend to any number of dimensions in a canonical way. If $n_1, \ldots, n_k \geq 1$, then we can have an array A of dimensions $n_1 \times \cdots \times n_k$, where $A[i_1, \ldots, i_k]$ is the $(i_1, \ldots, i_k)$-element of A. As in the one- and two-dimensional cases, the index $i_l \in \mathbb{Z}$ can always be considered modulo n_l, making A periodic in each of its dimensions.

Operations such as cyclic shifting extend to higher-dimensional arrays canonically as well. For example, we define the $(u_1, \ldots, u_k)$-*shift* of A to be the $n_1 \times \cdots \times n_k$ array whose $(i_1, \ldots, i_k)$-entry is $A[i_1 + u_1, \ldots, i_k + u_k]$. Similarly, the *reversal* of A is the $n_1 \times \cdots \times n_k$ array whose $(i_1, \ldots, i_k)$-entry is $A[-i_1, \ldots, -i_k]$.

The dot product of $n_1 \times \cdots \times n_k$ arrays A and B is the complex number

$$A \bullet B = \sum_{i_1=0}^{n_1-1} \cdots \sum_{i_k=0}^{n_k-1} A[i_1, \ldots, i_k] B[i_1, \ldots, i_k].$$

Chapter 3

Correlation and Convolution

3.1 Correlation and Convolution

Informally, correlation is a dot product of one array with a cyclic shift of a second array. It is called *cross-correlation* when the two arrays are different, and *autocorrelation* when the two arrays are the same.

Definition 3.1. *The* cross-correlation array *of A and B is the $m \times n$ array* $\mathbf{CC}(A, B)$ *with entries*

$$\mathbf{CC}(A, B)[u, v] = A \bullet (B^*)^{s_{u,v}}.$$

The autocorrelation array *of A is* $\mathbf{AC}(A) = \mathbf{CC}(A, A)$.

Example 3.2. *Let*

$$A = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \qquad \textit{and} \qquad B = \begin{bmatrix} 1 & \mathrm{i} \\ -1 & -\mathrm{i} \end{bmatrix}.$$

Then the cross-correlation and autocorrelation arrays are:

$$\mathbf{CC}(A,B) = \begin{bmatrix} -1+\mathrm{i} & -1+\mathrm{i} \\ 1-\mathrm{i} & 1-\mathrm{i} \end{bmatrix}; \quad \mathbf{AC}(A) = \begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix}; \quad \mathbf{AC}(B) = \begin{bmatrix} 4 & 0 \\ -4 & 0 \end{bmatrix}.$$

It follows easily from the definition that

$$\mathbf{CC}(A,B)[u,v] = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} A[i,j]B^*[i+u, j+v] \tag{3.1}$$

and

$$\mathbf{AC}(A)[u,v] = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} A[i,j]A^*[i+u, j+v]. \tag{3.2}$$

Equations (3.1) and (3.2) above are more commonly given as the definitions for auto- and cross-correlation, but they do not emphasise the dot-product nature of correlation, as Definition 3.1 does.

Definition 3.3. *The* convolution array *of A and B is the $m \times n$ array $A \circledast B$ with entries*

$$(A \circledast B)[u,v] = A \bullet (B^{s_{u,v}})^r.$$

Example 3.4. *The convolution array of*

$$A = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \qquad \textit{and} \qquad B = \begin{bmatrix} 1 & \mathrm{i} \\ -1 & -\mathrm{i} \end{bmatrix}$$

is

$$A \circledast B = \begin{bmatrix} -1-\mathrm{i} & -1-\mathrm{i} \\ 1+\mathrm{i} & 1+\mathrm{i} \end{bmatrix}.$$

Definition 3.3 expresses the elements of the convolution array $A \circledast B$ as dot products. In Chapter 6, we will show that these dot products allow us to manipulate arrays algebraically, without looking at the elements of the arrays. In

particular, we will prove the formula

$$\mathbf{CC}(A, B) = A^r \circledast B^*, \tag{3.3}$$

so

$$\mathbf{AC}(A) = A^r \circledast A^*, \tag{3.4}$$

and hence auto- and cross-correlation are recast in terms of convolution. The convolution algebra we develop in Chapter 6 will then allow us to prove correlation properties of arrays algebraically, without using the elements of the arrays. This was introduced in [13]. Presently, the only other way to manipulate arrays without appealing to their elements is with polynomials, which has been done in [1].

The elements of the convolution array $A \circledast B$ are not usually expressed as dot products. Instead, they are most often given as summations, similar to Equations (3.1) and (3.2). It follows easily that

$$(A \circledast B)[u, v] = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} A[i, j]B[u - i, v - j], \tag{3.5}$$

because

$$(A \circledast B)[u, v] = A \bullet (B^{s_{u,v}})^r = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} A[i, j](B^{s_{u,v}})^r[i, j].$$

We simplify $(B^{s_{u,v}})^r[i, j]$. Setting $C = B^{s_{u,v}}$, we have that $C[i, j] = B[i + u, j + v]$ and thus

$$(B^{s_{u,v}})^r[i, j] = C^r[i, j] = C[-i, -j] = B[u - i, v - j].$$

Equation (3.5) follows.

Correlation and convolution extend to higher dimensional arrays in a canonical

way. If A and B are $n_1 \times \cdots \times n_k$ arrays, then

$$\begin{aligned}
\mathbf{AC}(A)[u_1, \ldots, u_k] &= \sum_{i_1=0}^{n_1-1} \cdots \sum_{i_k=0}^{n_k-1} A[i_1, \ldots, i_k] A^*[i_1 + u_1, \ldots, i_k + u_k] \\
\mathbf{CC}(A, B)[u_1, \ldots, u_k] &= \sum_{i_1=0}^{n_1-1} \cdots \sum_{i_k=0}^{n_k-1} A[i_1, \ldots, i_k] B^*[i_1 + u_1, \ldots, i_k + u_k] \\
(A \circledast B)[u_1, \ldots, u_k] &= \sum_{i_1=0}^{n_1-1} \cdots \sum_{i_k=0}^{n_k-1} A[i_1, \ldots, i_k] B[u_1 - i_1, \ldots, u_k - i_k].
\end{aligned}$$

In Chapter 5, we will be dealing with autocorrelation of 4-dimensional arrays.

We saw in Chapter 2 that dot products of arrays can be expressed as sums of dot products of rows or columns. The same can be done for auto- and cross-correlation, as these are forms of dot products of arrays.

For example, if we decompose arrays into rows then we can write

$$\mathbf{CC}(A, B)[u, v] = A \bullet (B^*)^{s_{u,v}} = \sum_{i=0}^{m-1} A[i, \cdot] \bullet (B^*)^{s_{u,v}}[i, \cdot].$$

Now $(B^*)^{s_{u,v}}[i, \cdot]$ is the ith row of $(B^*)^{s_{u,v}}$, which is the $(i + u)$th row of B^*, shifted left by v. In symbols, this is $(B^*[i + u, \cdot])^{s_v}$. (Recall that $B^*[i + u, \cdot]$ is a *sequence*.) Thus

$$\mathbf{CC}(A, B)[u, v] = \sum_{i=0}^{m-1} A[i, \cdot] \bullet (B^*[i + u, \cdot])^{s_v} = \sum_{i=0}^{m-1} \mathbf{CC}(A[i, \cdot], B[i + u, \cdot])[v]$$

so that $\mathbf{CC}(A, B)[u, v]$ becomes a sum of m cross-correlations of rows of A with rows of B. Similarly, if we decompose arrays into columns then we get

$$\mathbf{CC}(A, B)[u, v] = \sum_{j=0}^{n-1} \mathbf{CC}(A[\cdot, j], B[\cdot, j + v])[u]$$

where $A[\cdot, j]$ is the jth column of A and $B[\cdot, j + v]$ is the $(j + v)$th column of B.

It is immediate that

$$\begin{aligned}\mathbf{AC}(A)[u,v] &= \mathbf{CC}(A,A)[u,v] \\ &= \sum_{i=0}^{m-1} \mathbf{CC}(A[i,\cdot], A[i+u,\cdot])[v] \\ &= \sum_{j=0}^{n-1} \mathbf{CC}(A[\cdot,j], A[\cdot,j+v])[u].\end{aligned}$$

Note that $A[i,\cdot]$ and $A[i+u,\cdot]$ are distinct rows of A, so the cross-correlations

$$\mathbf{CC}(A[i,\cdot], A[i+u,\cdot])$$

are not in general autocorrelations, unless the rows of A are shifts of the same sequence. (We will see this behaviour in Chapter 5.) Similarly,

$$\mathbf{CC}(A[\cdot,j], A[\cdot,j+v])$$

is not in general an autocorrelation, unless the columns of A are all shifts of the same sequence.

As array convolution is also a dot product of arrays, it too can be decomposed into sums of convolutions of rows/columns. We omit this, however, since we will not be using it in later chapters.

3.2 Perfect Arrays

The $(0,0)$-entry of $\mathbf{AC}(A)$ is

$$\mathbf{AC}(A)[0,0] = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} A[i,j]A^*[i,j] = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} |A[i,j]|^2 \tag{3.6}$$

where $|\cdot|$ denotes the usual complex norm, that is, complex magnitude. If the entries of A are roots of unity, with perhaps a few zeroes, then each term $|A[i, j]|^2$ will be 0 or 1 and the sum $\mathbf{AC}(A)[0, 0]$ will be a nonnegative integer of at most mn.

Definition 3.5. *The* peak autocorrelation *of A is the number $\mathbf{AC}(A)[0, 0]$. All other entries of $\mathbf{AC}(A)$ are called* off-peak autocorrelations.

It is immediate from Equation (3.6) that if the peak autocorrelation is zero, then the entire array will be zero. Conversely, if an array is zero, then every entry of the autocorrelation array will be zero, including the peak entry. We thus have the following lemma, which will come in handy in Chapter 7.

Lemma 3.6. *Let A be any array. Then $A = \mathbf{0}$ if and only if $\mathbf{AC}(A) = \mathbf{0}$, where $\mathbf{0}$ denotes the zero array.*

The off-peak autocorrelations may be signed integers or even complex numbers. Arrays with very low off-peak autocorrelation (with respect to absolute value) are widely sought in applications.

Example 3.7. *The following ternary array and its autocorrelation array are from [10]:*

$$A = \begin{bmatrix} 0 & 1 & -1 & -1 & -1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -1 & 0 & -1 & 0 & -1 & 1 \\ -1 & -1 & 1 & 1 & 1 & -1 & -1 \\ -1 & 0 & 1 & -1 & 1 & 0 & -1 \end{bmatrix} \qquad \mathbf{AC}(A) = \begin{bmatrix} 42 & 0 & 0 & 0 & 0 & 0 & 0 \\ -7 & 0 & 0 & 0 & 0 & 0 & 0 \\ -7 & 0 & 0 & 0 & 0 & 0 & 0 \\ -7 & 0 & 0 & 0 & 0 & 0 & 0 \\ -7 & 0 & 0 & 0 & 0 & 0 & 0 \\ -7 & 0 & 0 & 0 & 0 & 0 & 0 \\ -7 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The largest absolute value of the off-peak autocorrelations is 7*, and this is about* 16.67 *percent of the peak, which is low enough for this array to be used as a watermark, for example. The array above is an example of a class of arrays defined in [10]. We give the construction of these arrays in Section 4.4, and we use the above array as a watermark in Section 4.5.*

There is no absolute cut-off limit as to what defines "low" autocorrelation in applications. As can be seen from Example 3.7 above, it is possible to have off-peak autocorrelations which are zero. An array whose off-peak autocorrelations are *all* zero is given a special name.

Definition 3.8. *An array is* perfect *if all its off-peak autocorrelations are zero.*

Example 3.9. *The following* 3×6 *quaternary array is from [1]:*

$$A = \begin{bmatrix} -\mathrm{i} & -\mathrm{i} & -1 & 1 & 1 & 1 \\ -1 & -\mathrm{i} & -\mathrm{i} & -1 & 1 & -\mathrm{i} \\ -\mathrm{i} & 1 & -\mathrm{i} & 1 & \mathrm{i} & \mathrm{i} \end{bmatrix}.$$

Since the autocorrelation array of A is

$$\mathbf{AC}(A) = \begin{bmatrix} 18 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

it follows that A is an example of a perfect array.

Arrays with low but not perfect off-peak autocorrelation are sufficient for applications, such as digital watermarking. However, since perfect arrays are both useful and theoretically elegant, they continue to be studied. It is perfectly permissible in applications to have an array which has mainly zero off-peak autocorrelation, with a very few large nonzero entries equally spaced throughout the

autocorrelation array. Such arrays are generally called "almost perfect", and the precise definition that follows comes from [1]:

Definition 3.10. *Let A be an $m \times n$ array, where m is even. Then A is* almost perfect *if* $\mathbf{AC}(A)[0,0] = mn$, $\mathbf{AC}(A)[m/2,0] = -mn$ *and all other off-peak autocorrelations are zero.*

Example 3.11. *The following* 8×4 *almost perfect binary array and its autocorrelation array come from [13]:*

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \\ -1 & -1 & -1 & -1 \\ -1 & 1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ -1 & 1 & 1 & -1 \end{bmatrix} \quad \textit{with} \quad \mathbf{AC}(A) = \begin{bmatrix} 32 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -32 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

The fact that $m \times n$ almost perfect arrays have autocorrelation entries which are exactly mn and $-mn$ is used in Chapter 6 to produce an infinite recursive construction of almost perfect arrays. This is based on a result found in [1]. In Chapter 7, we generalise the construction further to yield an infinite recursive construction of what we call "n-perfect" arrays.

3.3 Auto- and Cross-Correlation of Sequences

Cross-correlation and autocorrelation of sequences follow similarly to arrays:

$$\mathbf{CC}(A, B)[u] = A \bullet (B^*)^{s_u} = \sum_{i=0}^{n-1} A[i]B^*[i+u]$$
$$\mathbf{AC}(A)[u] = A \bullet (A^*)^{s_u} = \sum_{i=0}^{n-1} A[i]A^*[i+u].$$

The convolution sequence of A and B is defined by

$$(A \circledast B)[u] = A \bullet (B^{s_u})^r$$

which has summation form

$$(A \circledast B)[u] = \sum_{i=0}^{n-1} A[i]B[u-i]. \tag{3.7}$$

Equation (3.7) above is the more common definition of $A \circledast B$. Finally, Equations (3.3) and (3.4) have sequence analogs:

$$\mathbf{CC}(A, B) = A^r \circledast B^* \tag{3.8}$$
$$\mathbf{AC}(A) = A^r \circledast A^*. \tag{3.9}$$

These will be proved along with the array versions in Chapter 6.

3.4 Perfect Arrays over the Quaternions

Quaternions are a noncommutative extension of the complex numbers. Whereas a complex number can be thought of as a point in $\mathbb{R}^2$, a quaternion can be thought of as a point in $\mathbb{R}^4$. When used as an alphabet, the extra two dimensions of a quater-

nion offer more ways to create arrays with low or even perfect autocorrelation.

Recall that a *quaternion* is an element of the $\mathbb{R}$-algebra with basis $\{1, \mathrm{i}, \mathrm{j}, \mathrm{k}\}$ such that

$$\mathrm{i}^2 = \mathrm{j}^2 = \mathrm{k}^2 = \mathrm{ijk} = -1. \qquad (*)$$

Thus quaternions are expressions of the form

$$q = a + b\mathrm{i} + c\mathrm{j} + d\mathrm{k}$$

where $a, b, c, d \in \mathbb{R}$, the elements $\mathrm{i}, \mathrm{j}, \mathrm{k}$ obey the relations in (*) above, and the usual laws of associativity and distributivity apply.

Quaternions were invented by Sir William Rowan Hamilton in 1843. In his honour, the set of all quaternions is denoted $\mathbb{H}$. It can be shown that $\mathbb{H}$ is a non-commutative division ring, which contains $\mathbb{C}$ as a subring. (A division ring is a ring where every nonzero element has a multiplicative inverse, but multiplication is not necessarily commutative.) The noncommutativity of $\mathbb{H}$ comes from the fact that

$$\begin{aligned} \mathrm{ij} &= \mathrm{k} & \mathrm{ji} &= -\mathrm{k} \\ \mathrm{jk} &= \mathrm{i} & \mathrm{kj} &= -\mathrm{i} \\ \mathrm{ki} &= \mathrm{j} & \mathrm{ik} &= -\mathrm{j}. \end{aligned}$$

Quaternions can be used instead of complex numbers to form entries for arrays (and sequences). Perfect sequences over the quaternions were introduced in [14], and perfect arrays over the quaternions were introduced in [2].

Definition 3.12. *The* basic quaternions *are* $\mathbb{H}_8 = \{\pm 1, \pm \mathrm{i}, \pm \mathrm{j}, \pm \mathrm{k}\}$.

It is easy to verify that $\mathbb{H}_8$ is a group under quaternion multiplication. In fact, $\mathbb{H}_8$ is commonly known as the "quaternion group", and is one of the five non-

isomorphic finite groups of order 8.

Basic quaternions have been used as an alphabet to create perfect arrays in [2] and [4]. Our contribution to the joint paper [4] was a simplification of the process of "inflation of perfect arrays", which was introduced in [1]. This simplification will be discussed fully in Chapter 5, as well as our extension of the inflation process to arrays over the basic quaternions.

The algebra of quaternions is similar to that of the complex numbers, but since quaternion multiplication is noncommutative, there is sometimes both a left and a right version of a definition (such as autocorrelation), depending on whether we choose to represent the product of $x, y \in \mathbb{H}$ as xy or yx.

Definition 3.13. *The* conjugate *of the quaternion* $q = a + b\mathrm{i} + c\mathrm{j} + d\mathrm{k}$ *is*

$$q^* = a - b\mathrm{i} - c\mathrm{j} - d\mathrm{k}.$$

It is immediate that $q^* = q$ if and only if $q \in \mathbb{R}$, and that $(q_1 q_2)^* = q_2^* q_1^*$. Note that $q_2^* q_1^*$ is different in general from $q_1^* q_2^*$. We also have that $(q_1 + q_2)^* = q_1^* + q_2^*$.

Definition 3.14. *The* norm *of the quaternion* $q = a + b\mathrm{i} + c\mathrm{j} + d\mathrm{k}$ *is the nonnegative real number*

$$|q|^2 = qq^* = q^* q = a^2 + b^2 + c^2 + d^2.$$

The dot product of two $m \times n$ arrays A and B over $\mathbb{H}$ is

$$A \bullet B = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} A[i, j]B[i, j] \in \mathbb{H}$$

which is the same as for arrays over $\mathbb{C}$. Note, however, that the quaternion $A[i, j]B[i, j]$ is different in general from $B[i, j]A[i, j]$, so the dot product of quaternion arrays is *not* commutative. The noncommutativity of the dot product will

lead us to two distinct definitions of auto- and cross-correlation of quaternion arrays.

Definition 3.15. *Let A and B be $m \times n$ arrays over $\mathbb{H}$. The* right cross-correlation array *of A and B is the $m \times n$ array $\mathbf{CC}_R(A, B)$ with entries*

$$\mathbf{CC}_R(A, B)[u, v] = A \bullet (B^*)^{s_{u,v}}.$$

Similarly, the left cross-correlation array *of A and B is the $m \times n$ array $\mathbf{CC}_L(A, B)$ with entries*

$$\mathbf{CC}_L(A, B)[u, v] = (B^*)^{s_{u,v}} \bullet A.$$

The right autocorrelation array *of A is $\mathbf{AC}_R(A) = \mathbf{CC}_R(A, A)$, and the* left autocorrelation array *of A is $\mathbf{AC}_L(A) = \mathbf{CC}_L(A, A)$.*

We give some examples of the differences between left and right autocorrelation of arrays over $\mathbb{H}$.

Example 3.16. *The right and left autocorrelation arrays of*

$$A = \begin{bmatrix} -\mathrm{i} & 1 & 1 \\ \mathrm{i} & \mathrm{j} & \mathrm{k} \end{bmatrix}$$

are

$$\mathbf{AC}_R(A) = \begin{bmatrix} 6 & 1 - \mathrm{i} - \mathrm{j} - \mathrm{k} & 1 + \mathrm{i} + \mathrm{j} + \mathrm{k} \\ -2 & 2\mathrm{j} & -2\mathrm{j} \end{bmatrix}$$

and

$$\mathbf{AC}_L(A) = \begin{bmatrix} 6 & 1 + \mathrm{i} + \mathrm{j} + \mathrm{k} & 1 - \mathrm{i} - \mathrm{j} - \mathrm{k} \\ -2 & -2\mathrm{k} & 2\mathrm{k} \end{bmatrix},$$

respectively.

Example 3.17. *The quaternion array*

$$A = \begin{bmatrix} 1 & \mathrm{i} \\ \mathrm{j} & \mathrm{k} \end{bmatrix}$$

has both right and left perfect autocorrelation:

$$\mathbf{AC}_R(A) = \mathbf{AC}_L(A) = \begin{bmatrix} 4 & 0 \\ 0 & 0 \end{bmatrix}.$$

Example 3.17 above raises the question of whether a quaternion array can have perfect right autocorrelation but not perfect left autocorrelation, or vice-versa. The following theorem, due to Kuznetsov [14], shows that this never happens for quaternion sequences; it can be extended in a canonical way to quaternion arrays.

Theorem 3.18. *A quaternion array has perfect left autocorrelation if and only if it has perfect right autocorrelation.*

Due to Theorem 3.18, the notion of a perfect quaternion array is well-defined. In Chapter 5, we will expound on our contribution to the inflation of perfect arrays over the basic quaternions, as well as over the complex numbers.

Chapter 4

Sequences and Arrays with Low Autocorrelation

4.1 Introduction

Sequences over the binary and quaternary alphabets are ideal for applications, such as watermarking: There are several classes of sequences over these alphabets with "optimal" autocorrelation, meaning a very large peak autocorrelation and the smallest off-peak autocorrelation possible—preferably zero, or a constant number very close to zero.

Over the binary alphabet, the only known perfect sequence (up to shifting and multiplying by -1) is

$$(1, 1, 1, -1).$$

The sequence

$$(0, \mathrm{i}, 1, -\mathrm{i}, 1, \mathrm{i})$$

is also perfect. It is an example of a *Lee sequence*, which are defined in [15] and used in [4].

In the sections that follow, we will discuss some of the known sequences and arrays with low but not perfect autocorrelation. We will then use one of the low-autocorrelation arrays from Section 4.4 in a watermarking example, which appears in Section 4.5. In this way, we show that sequences and arrays with low (but not perfect) autocorrelation are entirely useful in applications, even though sequences and arrays with technically perfect autocorrelation are highly sought.

4.2 Legendre Sequences

Let p be an odd prime. Let L_p be the length p sequence defined by

$$L_p[k] = \begin{cases} 0 & \text{if } k = 0; \\ 1 & \text{if } k \text{ is a quadratic residue modulo } p; \\ -1 & \text{otherwise.} \end{cases}$$

Then L_p is called the *Legendre sequence of length* p, and it is known that

$$\mathbf{AC}(L_p) = (p-1, -1, \ldots, -1).$$

The first few Legendre sequences are easy to construct by hand:

$$L_3 = (0, 1, -1)$$
$$L_5 = (0, 1, -1, -1, 1)$$
$$L_7 = (0, 1, 1, -1, 1, -1, -1).$$

There is one Legendre sequence for each odd prime p. For more information on Legendre sequences, see [9, 10].

4.3 M-Sequences

For each $k \geq 1$, a binary sequence D of length $2^k - 1$ can be constructed which has autocorrelation

$$\mathbf{AC}(D) = (2^k - 1, -1, \ldots, -1).$$

The construction of D is as follows.

We start with a polynomial

$$h(x) = a_k x^k + \cdots + a_1 x + a_0 \in \mathbb{Z}_2[x]$$

of degree k (so that $a_k \neq 0$). We construct a sequence C according to the recurrence

$$a_k C[i+k] + \cdots + a_1 C[i+1] + a_0 C[i] = 0 \qquad (i \geq 0)$$

where all addition and multiplication is modulo 2 and the initial conditions

$$C[k-1], \ldots, C[1], C[0] \in \mathbb{Z}_2$$

are not all zero but otherwise arbitrary.

For example, taking the degree 4 polynomial

$$h(x) = x^4 + x + 1$$

and the initial conditions

$$C[3] = 0, \quad C[2] = 0, \quad C[1] = 0, \quad C[0] = 1,$$

we construct C according to the following recurrence:

$$C[i+4] + C[i+1] + C[i] = 0 \qquad (i \geq 0)$$

or, recalling that all operations are done modulo 2, we rewrite:

$$C[i+4] = C[i+1] + C[i] \qquad (i \geq 0).$$

This particular sequence is periodic of length $2^4 - 1 = 15$, and is given by

$$C = (0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1).$$

The final step is to convert the sequence C (currently over $\mathbb{Z}_2$) into the binary alphabet. If we define the sequence D by the rule

$$D[i] = (-1)^{C[i]} \qquad (i \geq 0)$$

then we get

$$D = (1, 1, 1, -1, 1, 1, -1, -1, 1, -1, 1, -1, -1, -1, -1).$$

It can be checked that

$$\mathbf{AC}(D) = (15, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1).$$

The sequence D will always be periodic and of length (period) at most $2^k - 1$. If the period is exactly $2^k - 1$ (as in the example above), then D is called a *maximal-length sequence* or *M-sequence*. Furthermore, the polynomial $h(x)$ which generated C (and thence D) is called a *primitive polynomial*.

Not all polynomials are primitive polynomials. For example, the degree 4 polynomial

$$h(x) = x^4 + x^2 + 1$$

will generate the sequence C with recurrence

$$C[i+4] = C[i+2] + C[i] \qquad (i \geq 0).$$

With the same initial conditions

$$C[3] = 0, \quad C[2] = 0, \quad C[1] = 0, \quad C[0] = 1$$

as above, we get the sequence

$$C = (0, 0, 0, 1, 0, 1)$$

which is periodic of length 6. Converting C to the binary alphabet gives

$$D = (1, 1, 1, -1, 1, -1)$$

and it can be checked that

$$\mathbf{AC}(D) = (6, -2, 2, -2, 2, -2)$$

which is good but not optimal autocorrelation.

A primitive polynomial of degree k (at least one exists for each k) generating C will require k initial conditions $C[k-1], \ldots, C[0] \in \mathbb{Z}_2$. It turns out that any initial conditions chosen (so long as they are not all zero) will generate the same m-sequence, up to shifting. This is because every nonzero binary string of length

k (examples for length 4 are 0001 and 0110) appears somewhere in the sequence C. More information on M-sequences can be found in [17].

4.4 Hall-Osborne-Tirkel Arrays

In 2001, Hall et al. [10] presented a new family $\mathcal{F}_p$ of $p-1$ arrays (p is any prime) over the ternary alphabet. Each array in the family has low autocorrelation, and the cross-correlation between distinct arrays in the family is also low. We call these arrays *Hall-Osborne-Tirkel arrays*, for the sake of this section.

The construction for the arrays is as follows. For any prime p, we construct $p-1$ arrays $A_0, \ldots, A_{p-2}$ of dimensions $p \times p$. For $1 \leq m \leq p-1$ and $0 \leq j \leq p-1$, we make the jth column of A_m the shifted sequence $L_p^{s_{mj(j-1)/2}}$. (Recall that L_p is the Legendre sequence of length p.)

It is proved in [10] that each array in $\mathcal{F}_p$ has the same autocorrelation array, and that the off-peak autocorrelation values are from the set $\{-p, 0\}$, while the peak autocorrelation is $p(p-1)$. It is also proved that the cross-correlation values between any two distinct arrays in $\mathcal{F}_p$ are from the set $\{-p, 0, p\}$.

Example 4.1. *Recalling from Section 4.2 that* $L_3 = (0, 1, -1)$*, we have*

$$\mathbb{F}_3 = \left\{ \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 1 \\ -1 & 1 & -1 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 \\ 1 & -1 & 1 \\ -1 & 0 & -1 \end{bmatrix} \right\}.$$

Note that both arrays in the family have been shifted one place to the left to make them look symmetrical. The autocorrelation array of both the arrays in the family

is

$$\begin{bmatrix} 6 & 0 & 0 \\ -3 & 0 & 0 \\ -3 & 0 & 0 \end{bmatrix}$$

and the cross-correlation array between the two arrays in the family is

$$\begin{bmatrix} 3 & 0 & 0 \\ 0 & -3 & -3 \\ -3 & 3 & 3 \end{bmatrix}.$$

Example 4.2. *Recall that* $L_7 = (0, 1, 1, -1, 1, -1, -1)$. *The first two arrays in* $\mathcal{F}_7$ *are*

$$A_1 = \begin{bmatrix} 0 & 1 & -1 & -1 & -1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -1 & 0 & -1 & 0 & -1 & 1 \\ -1 & -1 & 1 & 1 & 1 & -1 & -1 \\ -1 & 0 & 1 & -1 & 1 & 0 & -1 \end{bmatrix}$$

and

$$A_2 = \begin{bmatrix} 0 & 1 & -1 & -1 & -1 & 1 & 0 \\ 1 & -1 & 0 & -1 & 0 & -1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ -1 & 0 & 1 & -1 & 1 & 0 & -1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}.$$

(Again, note that both arrays have been shifted one place to the left to preserve

vertical symmetry.) The autocorrelation array for both A_1 and A_2 is

$$\begin{bmatrix} 42 & 0 & 0 & 0 & 0 & 0 & 0 \\ -7 & 0 & 0 & 0 & 0 & 0 & 0 \\ -7 & 0 & 0 & 0 & 0 & 0 & 0 \\ -7 & 0 & 0 & 0 & 0 & 0 & 0 \\ -7 & 0 & 0 & 0 & 0 & 0 & 0 \\ -7 & 0 & 0 & 0 & 0 & 0 & 0 \\ -7 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

and the cross-correlation array between A_1 and A_2 is

$$\begin{bmatrix} 7 & 7 & 7 & 7 & 7 & 0 & 7 \\ -7 & 7 & 7 & -7 & 0 & -7 & 0 \\ 7 & 0 & 0 & 7 & -7 & -7 & -7 \\ 7 & -7 & -7 & 7 & -7 & 7 & -7 \\ 0 & -7 & -7 & 0 & 7 & -7 & 7 \\ -7 & 7 & 7 & -7 & -7 & 7 & -7 \\ -7 & -7 & -7 & -7 & 7 & 7 & 7 \end{bmatrix}.$$

4.5 Watermarking Example

In this section, we apply a watermark to a sample image to show how the basic process works. Our watermark will be the 7×7 ternary array

$$A_1 = \begin{bmatrix} 0 & 1 & -1 & -1 & -1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -1 & 0 & -1 & 0 & -1 & 1 \\ -1 & -1 & 1 & 1 & 1 & -1 & -1 \\ -1 & 0 & 1 & -1 & 1 & 0 & -1 \end{bmatrix}$$

from Example 4.2. Recall that the autocorrelation array of A_1 is

$$\mathbf{AC}(A_1) = \begin{bmatrix} 42 & 0 & 0 & 0 & 0 & 0 & 0 \\ -7 & 0 & 0 & 0 & 0 & 0 & 0 \\ -7 & 0 & 0 & 0 & 0 & 0 & 0 \\ -7 & 0 & 0 & 0 & 0 & 0 & 0 \\ -7 & 0 & 0 & 0 & 0 & 0 & 0 \\ -7 & 0 & 0 & 0 & 0 & 0 & 0 \\ -7 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Care must be taken to apply a watermark only to a portion of an image that has a low dynamic range of pixel values. We will show later in the section what

happens if this prescription is ignored. For now, assume that the 7×7 array

$$I = \begin{bmatrix} 300 & 299 & 298 & 297 & 296 & 295 & 294 \\ 299 & 298 & 297 & 297 & 296 & 295 & 294 \\ 298 & 298 & 297 & 296 & 295 & 294 & 293 \\ 297 & 297 & 296 & 295 & 295 & 294 & 294 \\ 296 & 296 & 296 & 295 & 294 & 293 & 293 \\ 295 & 295 & 295 & 294 & 294 & 293 & 292 \\ 295 & 294 & 294 & 293 & 293 & 292 & 292 \end{bmatrix}$$

is a portion of a larger image, such as a CAT scan. For example, I could represent a small low-resolution part of the skull that does not change drastically in brightness from pixel to pixel. (The absolute brightness of the pixels does not matter; only the range of values in the image must be small.)

We apply a shifted version of the watermark array to an image, and the information is stored in particular shift values chosen. We choose to store the information $(2, 3)$, and thus the shifted version of the watermark W becomes:

$$A_1^{s_{2,3}} = \begin{bmatrix} 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 \\ -1 & 0 & -1 & 1 & 1 & -1 & 0 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 \\ -1 & 1 & 0 & -1 & -1 & 0 & 1 \\ -1 & -1 & 1 & 0 & 0 & 1 & -1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Applying this watermark to the array A gives

$$I + A_1^{s_{2,3}} = \begin{bmatrix} 301 & 298 & 297 & 298 & 297 & 294 & 293 \\ 300 & 297 & 298 & 296 & 295 & 296 & 293 \\ 297 & 298 & 296 & 297 & 296 & 293 & 293 \\ 298 & 298 & 295 & 294 & 294 & 293 & 295 \\ 295 & 297 & 296 & 294 & 293 & 293 & 294 \\ 294 & 294 & 296 & 294 & 294 & 294 & 291 \\ 295 & 295 & 295 & 294 & 294 & 293 & 293 \end{bmatrix}$$

which is precisely the array that is transmitted over communication systems.

To recover the watermark, we perform a cross-correlation of the watermarked image $I + A_1^{s_{2,3}}$ with every possible shift of the original watermark array A_1. The result is the cross-correlation array

$$\mathbf{CC}(I + A_1^{s_{2,3}}, A_1) = \begin{bmatrix} 9 & 5 & 4 & -2 & 11 & 17 & 13 \\ 17 & 13 & 11 & 6 & 7 & 11 & 13 \\ -15 & -19 & -14 & 33 & -17 & -11 & -12 \\ -9 & -6 & -7 & -11 & -11 & -13 & -12 \\ -6 & -3 & -2 & -13 & -2 & -8 & 0 \\ 2 & 5 & 2 & -7 & 4 & -2 & 3 \\ 2 & 5 & 6 & -6 & 8 & 6 & 1 \end{bmatrix}$$

which we show below with the entries taken as absolute values for clarity:

$$\begin{bmatrix} 9 & 5 & 4 & 2 & 11 & 17 & 13 \\ 17 & 13 & 11 & 6 & 7 & 11 & 13 \\ 15 & 19 & 14 & \mathbf{33} & 17 & 11 & 12 \\ 9 & 6 & 7 & 11 & 11 & 13 & 12 \\ 6 & 3 & 2 & 13 & 2 & 8 & 0 \\ 2 & 5 & 2 & 7 & 4 & 2 & 3 \\ 2 & 5 & 6 & 6 & 8 & 6 & 1 \end{bmatrix}.$$

As can be seen above, the $(2, 3)$-entry of the cross-correlation array $\mathbf{CC}(I+A_1^{s_{2,3}}, A_1)$ is 33, which is clearly larger (although not too much larger) in absolute value to all the other entries. Thus our stored information must have been $(2, 3)$, so the watermark is recovered.

We will finish off this section with an example of an image which resists the watermarking of the array A_1 that we have been using. As discussed in previous sections, watermarks work best when applied to arrays whose pixels cross-correlate well (in other words, give low values) with the watermarking array. The array I above was such an example because all the pixel values were in the small range 292–300. We will now attempt to watermark an image whose pixels use the much larger pixel range 2–300. The image to be watermarked is:

$$J = \begin{bmatrix} 300 & 213 & 60 & 62 & 115 & 70 & 14 \\ 34 & 56 & 300 & 2 & 201 & 48 & 96 \\ 300 & 173 & 5 & 200 & 14 & 298 & 5 \\ 10 & 106 & 20 & 251 & 107 & 250 & 52 \\ 241 & 3 & 39 & 4 & 48 & 300 & 7 \\ 113 & 8 & 150 & 100 & 276 & 123 & 70 \\ 5 & 290 & 6 & 299 & 2 & 180 & 300 \end{bmatrix}.$$

We will watermark B with the same information $(2, 3)$. Thus, the watermarked image is

$$J + B^{s_{2,3}} = \begin{bmatrix} 301 & 212 & 59 & 63 & 116 & 69 & 13 \\ 35 & 55 & 301 & 1 & 200 & 49 & 95 \\ 299 & 173 & 4 & 201 & 15 & 297 & 5 \\ 11 & 107 & 19 & 250 & 106 & 249 & 53 \\ 240 & 4 & 39 & 3 & 47 & 300 & 8 \\ 112 & 7 & 151 & 100 & 276 & 124 & 69 \\ 5 & 291 & 7 & 300 & 3 & 181 & 301 \end{bmatrix}.$$

Following the same process as before, we will now attempt to recover the watermark by cross-correlating $J + B^{s_{2,3}}$ with every possible shift of the watermark B. This gives the cross-correlation array

$$\begin{bmatrix} 771 & -626 & -244 & -606 & -919 & 456 & 394 \\ -53 & -640 & 281 & -407 & 451 & 1006 & -733 \\ -790 & 597 & -437 & 216 & 499 & -956 & 2561 \\ 215 & 344 & -211 & -64 & 348 & 201 & -886 \\ 550 & -1214 & 1235 & -1020 & 225 & -861 & -354 \\ -174 & 917 & -373 & 725 & -1015 & -128 & -313 \\ -519 & 622 & -251 & 1156 & 411 & 282 & -669 \end{bmatrix}$$

which, when the absolute values of the entries are taken, gives

$$\begin{bmatrix} 771 & 626 & 244 & 606 & 919 & 456 & 394 \\ 53 & 640 & 281 & 407 & 451 & 1006 & 733 \\ 790 & 597 & 437 & \mathbf{216} & 499 & 956 & 2561 \\ 215 & 344 & 211 & 64 & 348 & 201 & 886 \\ 550 & 1214 & 1235 & 1020 & 225 & 861 & 354 \\ 174 & 917 & 373 & 725 & 1015 & 128 & 313 \\ 519 & 622 & 251 & 1156 & 411 & 282 & 669 \end{bmatrix}.$$

The $(2, 3)$-entry has been placed in bold, but as can be seen, it is indistinguishable from the other entries. Thus the watermark cannot be recovered from J.

Part II

Main Contributions

Chapter 5

Inflation of Perfect Complex and Quaternion Arrays

5.1 Introduction

In 2001, Arasu and de Launey [1] introduced the process of inflating perfect quaternary arrays. The process begins by representing a given perfect array as a Laurent polynomial in $\mathbb{C}[x, y, x^{-1}, y^{-1}]$, that is, a $\mathbb{C}$-linear combination of $x^i y^j$ with $i, j \in \mathbb{Z}$. Associated with this polynomial is a family of what Arasu and de Launey call "inflation polynomials", all of which come together to generate a single polynomial representing a perfect array of larger dimensions than the initial one. The process ends by converting the generated polynomial back to an array again.

In this chapter, we present a translation of the inflation process into the language of arrays and correlation, so that polynomial algebra is not needed. Under this translation, "inflation polynomials" now have a simple interpretation as "inflation arrays", which are presented in Section 5.2. Our translation of the inflation process itself is given in Section 5.3, and we give an example in Section 5.4. Then, in Section 5.5, we give an extension of the inflation process for arrays over

the pure quaternions. (This extension appears in our joint paper [4].)

5.2 Inflation Arrays

The following class of arrays will be used in the process of inflating perfect arrays; this definition is motivated from [1, Definition 17].

Definition 5.1. *A family $\Lambda_0, \ldots, \Lambda_{s-1}$ of s arrays, each of size $u \times v$, is a* family of s inflation arrays of dimensions $u \times v$ *if the following two conditions are satisfied:*

1. $\sum_{k=0}^{s-1} \mathbf{AC}(\Lambda_k)$ *is a $u \times v$ array whose $(0,0)$-entry is suv, and all other entries are zero;*

2. $\mathbf{CC}(\Lambda_i, \Lambda_j)$ *is a $u \times v$ array of 1s for all $i \neq j$.*

Example 5.2. *Let*

$$\Lambda_0 = \begin{bmatrix} -1 & -1 & -1 \\ 1 & 1 & 1 \\ -1 & -1 & -1 \end{bmatrix} \qquad \Lambda_1 = \begin{bmatrix} -1 & -1 & 1 \\ 1 & -1 & -1 \\ -1 & 1 & -1 \end{bmatrix}$$

$$\Lambda_2 = \begin{bmatrix} -1 & 1 & -1 \\ 1 & -1 & -1 \\ -1 & -1 & 1 \end{bmatrix} \qquad \Lambda_3 = \begin{bmatrix} -1 & 1 & -1 \\ -1 & 1 & -1 \\ -1 & 1 & -1 \end{bmatrix}.$$

It can be checked that

$$\sum_{k=0}^{3} \mathbf{AC}(\Lambda_k) = \begin{bmatrix} 36 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \textit{and} \quad \mathbf{CC}(\Lambda_i, \Lambda_j) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

for all $i \neq j$. Thus $\{\Lambda_0, \Lambda_1, \Lambda_2, \Lambda_3\}$ comprise a family of 4 inflation arrays of dimensions 3×3.

We now give a construction of a family of $p+1$ inflation arrays of dimensions $p \times p$, for every odd prime p. This construction appears in [4, Section 4.1] and is proved in [1, Theorem 19].

Let p be an odd prime. Let L_p be the Legendre sequence of length p. Then

$$\mathbf{AC}(L_p) = (p-1, -1, \ldots, -1),$$

as discussed in Section 4.2. If, as in [1, Theorem 19], we modify L_p to $\overline{L}_p$ by setting

$$\overline{L}_p[0] = \mathrm{i}^{(p+1)/2},$$

we get that

$$\mathbf{AC}(\overline{L}_p) = (p, -1, \ldots, -1).$$

(Recall that i denotes the primitive 4th root of unity.) The following theorem will show how to create a family $\Lambda_0, \ldots, \Lambda_p$ of $p+1$ inflation arrays of dimensions $p \times p$ using the modified Legendre sequence.

Theorem 5.3. *For any odd prime p, create $p+1$ arrays $\Lambda_0, \ldots, \Lambda_p$ according to the following rules:*

1. *For $0 \le k \le p-1$, the jth column of Λ_k is $\Lambda_k[\cdot, j] = \overline{L}_p^{s_{jk}}$;*
2. *The ith* row *of Λ_p is $\Lambda_p[i, \cdot] = \overline{L}_p$.*

Then $\Lambda_0, \ldots, \Lambda_p$ form a set of $p+1$ inflation arrays of dimensions $p \times p$.

Proof. See [1, Theorem 19]. □

Example 5.4. *Take $p = 5$. Then*

$$L_p = (0, 1, -1, -1, 1)$$

and since $\mathrm{i}^{(p+1)/2} = -\mathrm{i}$*, the modified Legendre sequence becomes*

$$\overline{L}_p = (-\mathrm{i}, 1, -1, -1, 1).$$

Using the modified $\overline{L}_p$ *above as a seed column, and following the algorithm outlined above, we construct six inflation arrays* $\Lambda_0, \ldots, \Lambda_5$*:*

$$\Lambda_0 = \begin{bmatrix} -\mathrm{i} & -\mathrm{i} & -\mathrm{i} & -\mathrm{i} & -\mathrm{i} \\ 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \qquad \Lambda_1 = \begin{bmatrix} -\mathrm{i} & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -\mathrm{i} \\ -1 & -1 & 1 & -\mathrm{i} & 1 \\ -1 & 1 & -\mathrm{i} & 1 & -1 \\ 1 & -\mathrm{i} & 1 & -1 & -1 \end{bmatrix}$$

$$\Lambda_2 = \begin{bmatrix} -\mathrm{i} & -1 & 1 & 1 & -1 \\ 1 & -1 & -\mathrm{i} & -1 & 1 \\ -1 & 1 & 1 & -1 & -\mathrm{i} \\ -1 & -\mathrm{i} & -1 & 1 & 1 \\ 1 & 1 & -1 & -\mathrm{i} & -1 \end{bmatrix} \qquad \Lambda_3 = \begin{bmatrix} -\mathrm{i} & -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & -\mathrm{i} & -1 \\ -1 & -\mathrm{i} & -1 & 1 & 1 \\ -1 & 1 & 1 & -1 & -\mathrm{i} \\ 1 & -1 & -\mathrm{i} & -1 & 1 \end{bmatrix}$$

$$\Lambda_4 = \begin{bmatrix} -\mathrm{i} & 1 & -1 & -1 & 1 \\ 1 & -\mathrm{i} & 1 & -1 & -1 \\ -1 & 1 & -\mathrm{i} & 1 & -1 \\ -1 & -1 & 1 & -\mathrm{i} & 1 \\ 1 & -1 & -1 & 1 & -\mathrm{i} \end{bmatrix} \qquad \Lambda_5 = \begin{bmatrix} -\mathrm{i} & 1 & -1 & -1 & 1 \\ -\mathrm{i} & 1 & -1 & -1 & 1 \\ -\mathrm{i} & 1 & -1 & -1 & 1 \\ -\mathrm{i} & 1 & -1 & -1 & 1 \\ -\mathrm{i} & 1 & -1 & -1 & 1 \end{bmatrix}.$$

It can be checked that

$$\sum_{k=0}^{5} \mathbf{AC}(\Lambda_k) = \begin{bmatrix} 150 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \textit{and} \quad \mathbf{CC}(\Lambda_i, \Lambda_j) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

for all $i \neq j$.

It will be important in Section 5.5 to know that the autocorrelation values of $\Lambda_0, \ldots, \Lambda_p$ are all real numbers. The following theorem guarantees this.

Theorem 5.5. *The entries of $\mathbf{AC}(\Lambda_0), \ldots, \mathbf{AC}(\Lambda_p)$ are all real numbers.*

Proof. Let $0 \leq k \leq p-1$. Then $\Lambda_k[\cdot, j] = \overline{L}_p^{s_{jk}}$, so if we decompose $\mathbf{AC}(\Lambda_k)$ into a sum of dot products of columns, we get that $\mathbf{AC}(\Lambda_k)$ is a sum of p different autocorrelations of the sequence $\overline{L}_p$. Since the values of $\mathbf{AC}(\overline{L}_p)$ are all either p or -1, it follows that $\mathbf{AC}(\Lambda_k)$ is a real number. A similar result follows if $\mathbf{AC}(\Lambda_p)$ is deconstructed into a sum of dot products of rows. □

Note that

$$\mathrm{i}^{(p+1)/2} = \begin{cases} \pm 1 & \text{if } p \equiv 3 \pmod 4 \\ \pm \mathrm{i} & \text{if } p \equiv 1 \pmod 4, \end{cases}$$

so $\Lambda_0, \ldots, \Lambda_p$ will be binary arrays if $p \equiv 3 \pmod 4$ and quaternary arrays if $p \equiv 1 \pmod 4$.

If the inflation arrays $\Lambda_0, \ldots, \Lambda_p$ are all binary, Theorem 19 of [1] allows us to combine the $p+1$ arrays in pairs, and create a new set $\Gamma_0, \ldots, \Gamma_{(p-1)/2}$ of $(p+1)/2$ inflation arrays, all of the same dimensions $p \times p$. For $0 \leq k \leq (p-1)/2$, the construction for Γ_k is

$$\Gamma_k = \left(\frac{1+\mathrm{i}}{2}\right)(\Lambda_{2k} + \mathrm{i}\Lambda_{2k+1}).$$

Note that because $\Lambda_0, \ldots, \Lambda_p$ are all binary arrays, the array Λ_{2k} will have entries ± 1 and the array $\mathrm{i}\Lambda_{2k+1}$ will have entries $\pm \mathrm{i}$. The entries of $\Lambda_{2k} + \mathrm{i}\Lambda_{2k+1}$ will thus all be of the form $\pm 1 \pm \mathrm{i}$, which are the vertices of the square of side length 2 centred around the origin of the complex plane. Multiplying these numbers by $(1 + \mathrm{i})/2$ has the effect of rotating them counterclockwise by 45 degrees, as well as scaling their distance from the origin by a factor of $1/\sqrt{2}$. The result is that the numbers $\pm 1 \pm \mathrm{i}$ will be transformed into elements of $\{\pm 1, \pm \mathrm{i}\}$, making Γ_k a quaternary array.

Example 5.6. *Recall from Example 5.2 that*

$$\Lambda_0 = \begin{bmatrix} -1 & -1 & -1 \\ 1 & 1 & 1 \\ -1 & -1 & -1 \end{bmatrix} \qquad \Lambda_1 = \begin{bmatrix} -1 & -1 & 1 \\ 1 & -1 & -1 \\ -1 & 1 & -1 \end{bmatrix}$$

$$\Lambda_2 = \begin{bmatrix} -1 & 1 & -1 \\ 1 & -1 & -1 \\ -1 & -1 & 1 \end{bmatrix} \qquad \Lambda_3 = \begin{bmatrix} -1 & 1 & -1 \\ -1 & 1 & -1 \\ -1 & 1 & -1 \end{bmatrix}$$

form a set of 4 *inflation arrays of dimensions* 3×3. *Pairing* Λ_0 *with* Λ_1 *and* Λ_2 *with* Λ_3*, we get that*

$$\Lambda_0 + \mathrm{i}\Lambda_1 = \begin{bmatrix} -1-\mathrm{i} & -1-\mathrm{i} & -1+\mathrm{i} \\ 1+\mathrm{i} & 1-\mathrm{i} & 1-\mathrm{i} \\ -1-\mathrm{i} & -1+\mathrm{i} & -1-\mathrm{i} \end{bmatrix}$$

$$\Lambda_2 + \mathrm{i}\Lambda_3 = \begin{bmatrix} -1-\mathrm{i} & 1+\mathrm{i} & -1-\mathrm{i} \\ 1-\mathrm{i} & -1+\mathrm{i} & -1-\mathrm{i} \\ -1-\mathrm{i} & -1+\mathrm{i} & 1-\mathrm{i} \end{bmatrix}$$

and multiplying the above arrays by $(1 + \mathrm{i})/2$ *gives*

$$\Gamma_0 = \begin{bmatrix} -\mathrm{i} & -\mathrm{i} & -1 \\ \mathrm{i} & 1 & 1 \\ -\mathrm{i} & -1 & -\mathrm{i} \end{bmatrix} \quad \textit{and} \quad \Gamma_1 = \begin{bmatrix} -\mathrm{i} & \mathrm{i} & -\mathrm{i} \\ 1 & -1 & -\mathrm{i} \\ -\mathrm{i} & -1 & 1 \end{bmatrix}.$$

It can be checked that

$$\sum_{k=0}^{1} \mathbf{AC}(\Lambda_k) = \begin{bmatrix} 18 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

and that

$$\mathbf{CC}(\Lambda_i, \Lambda_j) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

for all $i \neq j$.

5.3 The Inflation Process

Let A be a perfect $m \times n$ array over fth roots of unity (so that all entries are nonzero and of unit magnitude). Assume that we have the following ingredients:

1. A family $\Lambda_0, \ldots, \Lambda_{mn-1}$ of mn inflation arrays of dimensions $u \times v$, whose entries are gth roots of unity;

2. A bijection $\phi : \mathbb{Z}_m \times \mathbb{Z}_n \to \mathbb{Z}_{mn}$.

We will use $\Lambda_0, \ldots, \Lambda_{mn-1}$ and ϕ to inflate A into B, a 4-dimensional $m \times n \times u \times v$ array which will be perfect and over an alphabet of $\mathrm{lcm}(f, g)$-th roots of unity. Finally, if m is coprime to u and n is coprime to v, then B can be converted into C, a perfect $mu \times nv$ array over $\mathrm{lcm}(f, g)$-th roots of unity.

For $(i, j, k, l) \in \mathbb{Z}_m \times \mathbb{Z}_n \times \mathbb{Z}_u \times \mathbb{Z}_v$, define $B[i, j, k, l]$ by the rule

$$B[i, j, k, l] = A[i, j]\Lambda_{\phi(i,j)}[k, l].$$

Theorem 5.7. *B is perfect.*

Proof. Let $(q, r, s, t) \in \mathbb{Z}_m \times \mathbb{Z}_n \times \mathbb{Z}_u \times \mathbb{Z}_v$. Then

$$\begin{aligned}
\mathbf{AC}(B)[q, r, s, t] &= \sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\sum_{k=0}^{u-1}\sum_{l=0}^{v-1} B[i, j, k, l]B^*[i + q, j + r, k + s, l + t] \\
&= \sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\sum_{k=0}^{u-1}\sum_{l=0}^{v-1} \Big(A[i, j]\Lambda_{\phi(i,j)}[k, l]\Big)\Big(A[i + q, j + r]\Lambda_{\phi(i+q,j+r)}[k + s, l + t]\Big)^* \\
&= \sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\sum_{k=0}^{u-1}\sum_{l=0}^{v-1} (A[i, j]A^*[i + q, j + r])\Big(\Lambda_{\phi(i,j)}[k, l]\Lambda^*_{\phi(i+q,j+r)}[k + s, l + t]\Big) \\
&= \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} (A[i, j]A^*[i + q, j + r]) \sum_{k=0}^{u-1}\sum_{l=0}^{v-1}\Big(\Lambda_{\phi(i,j)}[k, l]\Lambda^*_{\phi(i+q,j+r)}[k + s, l + t]\Big) \\
&= \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} (A[i, j]A^*[i + q, j + r])\, \mathbf{CC}(\Lambda_{\phi(i,j)}, \Lambda_{\phi(i+q,j+r)})[s, t],
\end{aligned}$$

which we summarise as

$$\mathbf{AC}(B)[q, r, s, t] = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} (A[i, j]A^*[i + q, j + r])\, \mathbf{CC}(\Lambda_{\phi(i,j)}, \Lambda_{\phi(i+q,j+r)})[s, t]. \tag{5.1}$$

Recall that $(q, r) \in \mathbb{Z}_m \times \mathbb{Z}_n$. If $(q, r) = (0, 0)$, then Equation (5.1) gives

$$\begin{aligned}\mathbf{AC}(B)[q, r, s, t] &= \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (A[i, j]A^*[i, j])\, \mathbf{CC}(\Lambda_{\phi(i,j)}, \Lambda_{\phi(i,j)})[s, t] \\ &= \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} |A[i, j]|^2 \mathbf{AC}(\Lambda_{\phi(i,j)})[s, t] \\ &= \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \mathbf{AC}(\Lambda_{\phi(i,j)})[s, t].\end{aligned}$$

The last equality follows because A is over roots of unity, so $|A[i, j]|^2 = 1$ for all (i, j). Now ϕ maps $\mathbb{Z}_m \times \mathbb{Z}_n$ bijectively onto $\mathbb{Z}_{mn}$, so

$$\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \mathbf{AC}(\Lambda_{\phi(i,j)})[s, t] = \sum_{k=0}^{mn-1} \mathbf{AC}(\Lambda_k)[s, t].$$

To summarise, if $(q, r) = (0, 0)$ then

$$\mathbf{AC}(B)[q, r, s, t] = \sum_{k=0}^{mn-1} \mathbf{AC}(\Lambda_k)[s, t].$$

Because $\Lambda_0, \ldots, \Lambda_{mn-1}$ are inflation arrays, this means that, for $(q, r) = (0, 0)$,

$$\mathbf{AC}(B)[q, r, s, t] = \begin{cases} mnuv & \text{if } (s, t) = (0, 0) \\ 0 & \text{if } (s, t) \neq (0, 0). \end{cases} \tag{5.2}$$

Now we consider the case where $(q, r) \neq (0, 0)$. Then, working in $\mathbb{Z}_m \times \mathbb{Z}_n$, (i, j) is distinct from $(i + q, j + r)$ for all $(i, j) \in \mathbb{Z}_m \times \mathbb{Z}_n$. Since ϕ is a bijection, this means that $\phi(i, j)$ and $\phi(i + q, j + r)$ are always distinct elements of $\mathbb{Z}_{mn}$. It follows that $\Lambda_{\phi(i,j)}$ and $\Lambda_{\phi(i+q,j+r)}$ are always distinct inflation arrays, so

$$\mathbf{CC}(\Lambda_{\phi(i,j)}, \Lambda_{\phi(i+q,j+r)})[s, t] = 1$$

for all (s, t). Thus, for $(q, r) \neq (0, 0)$, Equation (5.1) gives

$$\mathbf{AC}(B)[q, r, s, t] = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} A[i, j]A^*[i + q, j + r] = \mathbf{AC}(A)[q, r].$$

Since A is perfect and $(q, r) \neq (0, 0)$, it follows that

$$\mathbf{AC}(B)[q, r, s, t] = \mathbf{AC}(A)[q, r] = 0 \tag{5.3}$$

for all (s, t). Equations (5.2) and (5.3) together prove that B is perfect. □

The final step in the inflation process is to convert the 4-dimensional $m \times n \times u \times v$ array B into a 2-dimensional $mu \times nv$ array C. If $(i, j) \in \mathbb{Z}_{mu} \times \mathbb{Z}_{nv}$, then the (i, j)-element of C is given by

$$C[i, j] = B[i \bmod m, j \bmod n, i \bmod u, j \bmod v].$$

If m is coprime to u and n is coprime to v, then the maps

$$\mathbb{Z}_{mu} \to \mathbb{Z}_m \times \mathbb{Z}_u : i \mapsto (i \bmod m, i \bmod u)$$
$$\mathbb{Z}_{nv} \to \mathbb{Z}_n \times \mathbb{Z}_v : j \mapsto (j \bmod n, j \bmod v)$$

are both bijections by the Chinese Remainder Theorem. Thus the elements of C are a rearrangement of the elements of B, and the following theorem proves that autocorrelation is preserved. A similar theorem is proved in [4, Theorem 3].

Theorem 5.8. *Let $m, n, u, v \geq 1$, with m coprime to u, and n coprime to v. Let K be a $m \times n \times u \times v$ array, and let L be the $mu \times nv$ array such that*

$$L[i, j] = K[i \bmod m, j \bmod n, i \bmod u, j \bmod v].$$

Then

$$\mathbf{AC}(L)[i, j] = \mathbf{AC}(K)[i \bmod m, j \bmod n, i \bmod u, j \bmod v].$$

Proof. For brevity, we use in this proof the notation $a_{(n)}$ to mean $a \bmod n$. It is not true in general that $(a + b)_{(n)} = a_{(n)} + b_{(n)}$, but it is always true that

$$(a + b)_{(n)} \equiv a_{(n)} + b_{(n)} \pmod{n}.$$

We have that

$$\begin{aligned}
\mathbf{AC}(L)[i, j] &= \sum_{x=0}^{mu-1} \sum_{y=0}^{nv-1} L[x, y] L^*[x + i, y + j] \\
&= \sum_{x=0}^{mu-1} \sum_{y=0}^{nv-1} K[x_{(m)}, y_{(n)}, x_{(u)}, y_{(v)}] K^*[(x + i)_{(m)}, (y + j)_{(n)}, (x + i)_{(u)}, (y + j)_{(v)}] \\
&= \sum_{x=0}^{mu-1} \sum_{y=0}^{nv-1} K[x_{(m)}, y_{(n)}, x_{(u)}, y_{(v)}] K^*[x_{(m)} + i_{(m)}, y_{(n)} + j_{(n)}, x_{(u)} + i_{(u)}, y_{(v)} + j_{(v)}].
\end{aligned}$$

By the Chinese Remainder Theorem, there are bijections

$$\begin{aligned}
\mathbb{Z}_{mu} &\to \mathbb{Z}_m \times \mathbb{Z}_u : x \mapsto (x_{(m)}, x_{(u)}) \\
\mathbb{Z}_{nv} &\to \mathbb{Z}_n \times \mathbb{Z}_v : y \mapsto (y_{(n)}, y_{(v)}).
\end{aligned}$$

If we write

$$q = x_{(m)} \qquad r = y_{(n)} \qquad s = x_{(u)} \qquad y = y_{(v)},$$

then we have

$$\begin{aligned}
\mathbf{AC}(L)[i, j] &= \sum_{q=0}^{m-1} \sum_{r=0}^{n-1} \sum_{s=0}^{u-1} \sum_{t=0}^{v-1} K[q, r, s, t] K^*[q + i_{(m)}, r + j_{(n)}, s + i_{(u)}, t + j_{(v)}] \\
&= \mathbf{AC}(K)[i_{(m)}, j_{(n)}, i_{(u)}, j_{(v)}].
\end{aligned}$$

□

Taking $K = B$ and $L = C$, it follows from Theorem 5.8 that C is perfect.

We make a final comment about the alphabet of the inflated array C. Recall that A is over f roots of unity and that the inflation arrays $\Lambda_0, \ldots, \Lambda_{mn-1}$ are over g roots of unity. Since the construction for B is

$$B[i, j, k, l] = A[i, j]\Lambda_{\phi(i,j)}[k, l],$$

it follows that B is over $\text{lcm}(f, g)$ roots of unity. And as C is a rearrangement of the elements of B, it follows that C is over the same alphabet as B.

5.4 Inflation Example

The 1×4 binary array

$$A = \begin{bmatrix} 1 & 1 & 1 & -1 \end{bmatrix}$$

is perfect. The arrays

$$\Lambda_0 = \begin{bmatrix} -1 & -1 & -1 \\ 1 & 1 & 1 \\ -1 & -1 & -1 \end{bmatrix} \qquad \Lambda_1 = \begin{bmatrix} -1 & -1 & 1 \\ 1 & -1 & -1 \\ -1 & 1 & -1 \end{bmatrix}$$

$$\Lambda_2 = \begin{bmatrix} -1 & 1 & -1 \\ 1 & -1 & -1 \\ -1 & -1 & 1 \end{bmatrix} \qquad \Lambda_3 = \begin{bmatrix} -1 & 1 & -1 \\ -1 & 1 & -1 \\ -1 & 1 & -1 \end{bmatrix}.$$

from Example 5.2 form a set of 4 inflation arrays of dimensions 3×3. We use the bijection

$$\phi : \mathbb{Z}_1 \times \mathbb{Z}_4 \to \mathbb{Z}_4, \quad (i, j) \mapsto j.$$

Following the algorithm in Section 5.3, the $1 \times 4 \times 3 \times 3$ array B becomes

$$\begin{aligned}
&\begin{bmatrix} A[0,0]\Lambda_{\phi(0,0)} & A[0,1]\Lambda_{\phi(0,1)} & A[0,2]\Lambda_{\phi(0,2)} & A[0,3]\Lambda_{\phi(0,3)} \end{bmatrix} \\
&= \begin{bmatrix} \Lambda_0 & \Lambda_1 & \Lambda_2 & -\Lambda_3 \end{bmatrix} \\
&= \left[\begin{bmatrix} -1 & -1 & -1 \\ 1 & 1 & 1 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} -1 & -1 & 1 \\ 1 & -1 & -1 \\ -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} -1 & 1 & -1 \\ 1 & -1 & -1 \\ -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & 1 \\ 1 & -1 & 1 \\ 1 & -1 & 1 \end{bmatrix} \right]
\end{aligned}$$

Finally, we rearrange the elements of the $1 \times 4 \times 3 \times 3$ array B to form the $(1 \cdot 3) \times (4 \cdot 3) = 3 \times 12$ array C according to the rule

$$C[i, j] = B[i \bmod 1, j \bmod 4, i \bmod 3, j \bmod 3].$$

This gives

$$C = \begin{bmatrix} -1 & -1 & -1 & 1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 \\ -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 \end{bmatrix}.$$

Since

$$\mathbf{AC}(C) = \begin{bmatrix} 36 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

it follows that C is perfect.

5.5 Inflation of Perfect Quaternion Arrays

The array inflation process of Section 5.3 can be extended to inflate arrays over the quaternion alphabet $\mathbb{H}$. (Recall that arrays over $\mathbb{H}$ and their correlation properties

are studied in Chapter 3.) The extension given here appears in [4].

Instead of inflating a perfect array A over $\mathbb{C}$, we instead inflate perfect arrays over $\mathbb{H}$. (Recall that although there is a left and a right autocorrelation of arrays over $\mathbb{H}$, the two concepts coincide for perfect arrays.) That is the only difference that we will discuss here. The mathematics in Section 5.2 for inflation arrays will remain the same. We will assume that all inflation arrays are over the quaternary alphabet $\{\pm 1, \pm \mathrm{i}\}$, so there are no quaternion entries. (All inflation arrays in Section 5.2 are of this kind.) Thus left or right auto- or cross-correlation will not be used; the usual $\mathbf{AC}(\cdot)$ or $\mathbf{CC}(\cdot)$ will suffice.

Let A be a perfect $m \times n$ array over the basic quaternions $\mathbb{H}_8$. As in Section 5.3, assume that we have the following ingredients:

1. A family $\Lambda_0, \ldots, \Lambda_{mn-1}$ of mn inflation arrays of dimensions $u \times v$;

2. A bijection $\phi : \mathbb{Z}_m \times \mathbb{Z}_n \to \mathbb{Z}_{mn}$.

We will use $\Lambda_0, \ldots, \Lambda_{mn-1}$ and ϕ to inflate A into B, a 4-dimensional $m \times n \times u \times v$ array which will be perfect. Because A is over $\mathbb{H}_8$ and $\Lambda_0, \ldots, \Lambda_{mn-1}$ are over $\{\pm 1, \pm \mathrm{i}\}$, the entries of B will be over $\mathbb{H}_8$. Finally, as in Section 5.3, if m is coprime to u and n is coprime to v, then B can be converted into C, a perfect $mu \times nv$ array over $\mathbb{H}_8$.

For $(i, j, k, l) \in \mathbb{Z}_m \times \mathbb{Z}_n \times \mathbb{Z}_u \times \mathbb{Z}_v$, define $B[i, j, k, l] = A[i, j]\Lambda_{\phi(i,j)}[k, l]$.

Theorem 5.9. *B is perfect.*

Proof. The proof will follow that of Section 5.3, but we will give due care when commuting terms in quaternion products. For simplicity, we will always use the right autocorrelation $\mathbf{AC}_R(\cdot)$ when dealing with the perfect quaternion array A. Recall from Theorem 3.18 that left and right autocorrelation coincide where perfect arrays are concerned.

Further recall that, for $a, b \in \mathbb{H}$, it is always true that $(ab)^* = b^*a^*$. Finally, note that the elements of $\mathbb{H}_8 = \{\pm 1, \pm \mathrm{i}, \pm \mathrm{j}, \pm \mathrm{k}\}$ all have norm 1. If $(q, r, s, t) \in \mathbb{Z}_m \times \mathbb{Z}_n \times \mathbb{Z}_u \times \mathbb{Z}_v$, then

$$\begin{aligned}
\mathbf{AC}_R(B)[q,r,s,t] &= \sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\sum_{k=0}^{u-1}\sum_{l=0}^{v-1} B[i,j,k,l]B^*[i+q, j+r, k+s, l+t] \\
&= \sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\sum_{k=0}^{u-1}\sum_{l=0}^{v-1} \Big(A[i,j]\Lambda_{\phi(i,j)}[k,l]\Big)\Big(A[i+q, j+r]\Lambda_{\phi(i+q,j+r)}[k+s, l+t]\Big)^* \\
&= \sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\sum_{k=0}^{u-1}\sum_{l=0}^{v-1} \Big(A[i,j]\Lambda_{\phi(i,j)}[k,l]\Big)\Big(\Lambda^*_{\phi(i+q,j+r)}[k+s, l+t]A^*[i+q, j+r]\Big) \\
&= \sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\sum_{k=0}^{u-1}\sum_{l=0}^{v-1} A[i,j]\Big(\Lambda_{\phi(i,j)}[k,l]\Lambda^*_{\phi(i+q,j+r)}[k+s, l+t]\Big)A^*[i+q, j+r] \\
&= \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} A[i,j]\left(\sum_{k=0}^{u-1}\sum_{l=0}^{v-1} \Lambda_{\phi(i,j)}[k,l]\Lambda^*_{\phi(i+q,j+r)}[k+s, l+t]\right)A^*[i+q, j+r] \\
&= \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} A[i,j]\Big(\mathbf{CC}(\Lambda_{\phi(i,j)}, \Lambda_{\phi(i+q,j+r)})[s,t]\Big)A^*[i+q, j+r],
\end{aligned}$$

which we summarise as

$$\mathbf{AC}_R(B)[q,r,s,t] = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} A[i,j]\Big(\mathbf{CC}(\Lambda_{\phi(i,j)}, \Lambda_{\phi(i+q,j+r)})[s,t]\Big)A^*[i+q, j+r]. \tag{5.4}$$

Recall that $(q, r) \in \mathbb{Z}_m \times \mathbb{Z}_n$. If $(q, r) = (0, 0)$, then Equation (5.4) gives

$$\mathbf{AC}_R(B)[q,r,s,t] = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} A[i,j]\Big(\mathbf{AC}(\Lambda_{\phi(i,j)})[s,t]\Big)A^*[i,j].$$

By Theorem 5.5, the values $\mathbf{AC}(\Lambda_{\phi(i,j)})[s,t]$ are all real numbers. As real numbers

commute with quaternions, we may write

$$\begin{aligned}\mathbf{AC}_R(B)[q,r,s,t] &= \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} |A[i,j]|^2 \mathbf{AC}(\Lambda_{\phi(i,j)})[s,t] \\ &= \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} \mathbf{AC}(\Lambda_{\phi(i,j)})[s,t].\end{aligned}$$

The last equality follows because A is over $\mathbb{H}_8$, so $|A[i,j]|^2 = 1$ for all (i,j). Now ϕ maps $\mathbb{Z}_m \times \mathbb{Z}_n$ bijectively onto $\mathbb{Z}_{mn}$, so

$$\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} \mathbf{AC}(\Lambda_{\phi(i,j)})[s,t] = \sum_{k=0}^{mn-1} \mathbf{AC}(\Lambda_k)[s,t].$$

To summarise, if $(q,r) = (0,0)$ then

$$\mathbf{AC}_R(B)[q,r,s,t] = \sum_{k=0}^{mn-1} \mathbf{AC}(\Lambda_k)[s,t].$$

Because $\Lambda_0, \ldots, \Lambda_{mn-1}$ are inflation arrays, this means that, for $(q,r) = (0,0)$,

$$\mathbf{AC}(B)_R[q,r,s,t] = \begin{cases} mnuv & \text{if } (s,t) = (0,0) \\ 0 & \text{if } (s,t) \neq (0,0). \end{cases} \tag{5.5}$$

Now we consider the case where $(q,r) \neq (0,0)$. Then, working in $\mathbb{Z}_m \times \mathbb{Z}_n$, (i,j) is distinct from $(i+q, j+r)$ for all $(i,j) \in \mathbb{Z}_m \times \mathbb{Z}_n$. Since ϕ is a bijection, this means that $\phi(i,j)$ and $\phi(i+q, j+r)$ are always distinct elements of $\mathbb{Z}_{mn}$. It follows that $\Lambda_{\phi(i,j)}$ and $\Lambda_{\phi(i+q,j+r)}$ are always distinct inflation arrays, so

$$\mathbf{CC}(\Lambda_{\phi(i,j)}, \Lambda_{\phi(i+q,j+r)})[s,t] = 1$$

for all (s, t). Thus, for $(q, r) \neq (0, 0)$, Equation (5.4) gives

$$\mathbf{AC}(B)_R[q, r, s, t] = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} A[i, j]A^*[i + q, j + r] = \mathbf{AC}_R(A)[q, r].$$

Since A is perfect and $(q, r) \neq (0, 0)$, it follows that

$$\mathbf{AC}_R(B)[q, r, s, t] = \mathbf{AC}_R(A)[q, r] = 0 \tag{5.6}$$

for all (s, t). Equations (5.5) and (5.6) together prove that B is perfect. □

The perfect $m \times n \times u \times v$ array B can be converted (using the Chinese Remainder Theorem) into a 2-dimensional $mu \times nv$ array C in the same way as in Section 5.3. The proof of Theorem 5.8 holds automatically for quaternion arrays because there are no products of more than two quaternions considered. The result is that C will be a perfect $mu \times nv$ array over the same quaternion alphabet as A.

5.6 Quaternion Inflation Example

The 1×4 quaternion array

$$A = \begin{bmatrix} 1 & \mathrm{i} & \mathrm{j} & \mathrm{k} \end{bmatrix}$$

is perfect. The arrays

$$\Lambda_0 = \begin{bmatrix} -1 & -1 & -1 \\ 1 & 1 & 1 \\ -1 & -1 & -1 \end{bmatrix} \qquad \Lambda_1 = \begin{bmatrix} -1 & -1 & 1 \\ 1 & -1 & -1 \\ -1 & 1 & -1 \end{bmatrix}$$

$$\Lambda_2 = \begin{bmatrix} -1 & 1 & -1 \\ 1 & -1 & -1 \\ -1 & -1 & 1 \end{bmatrix} \qquad \Lambda_3 = \begin{bmatrix} -1 & 1 & -1 \\ -1 & 1 & -1 \\ -1 & 1 & -1 \end{bmatrix}.$$

from Example 5.2 form a set of 4 inflation arrays of dimensions 3×3. We use the bijection

$$\phi : \mathbb{Z}_1 \times \mathbb{Z}_4 \to \mathbb{Z}_4, \quad (i, j) \mapsto j.$$

Following the algorithm in Section 5.3, the $1 \times 4 \times 3 \times 3$ array B becomes

$$\begin{aligned}
&\begin{bmatrix} A[0,0]\Lambda_{\phi(0,0)} & A[0,1]\Lambda_{\phi(0,1)} & A[0,2]\Lambda_{\phi(0,2)} & A[0,3]\Lambda_{\phi(0,3)} \end{bmatrix} \\
&= \begin{bmatrix} \Lambda_0 & \mathrm{i}\Lambda_1 & \mathrm{j}\Lambda_2 & \mathrm{k}\Lambda_3 \end{bmatrix} \\
&= \left[\begin{bmatrix} -1 & -1 & -1 \\ 1 & 1 & 1 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} -\mathrm{i} & -\mathrm{i} & \mathrm{i} \\ \mathrm{i} & -\mathrm{i} & -\mathrm{i} \\ -\mathrm{i} & \mathrm{i} & -\mathrm{i} \end{bmatrix} \begin{bmatrix} -\mathrm{j} & \mathrm{j} & -\mathrm{j} \\ \mathrm{j} & -\mathrm{j} & -\mathrm{j} \\ -\mathrm{j} & -\mathrm{j} & \mathrm{j} \end{bmatrix} \begin{bmatrix} -\mathrm{k} & \mathrm{k} & -\mathrm{k} \\ -\mathrm{k} & \mathrm{k} & -\mathrm{k} \\ -\mathrm{k} & \mathrm{k} & -\mathrm{k} \end{bmatrix}\right]
\end{aligned}$$

Finally, we rearrange the elements of the $1 \times 4 \times 3 \times 3$ array B to form the $(1 \cdot 3) \times (4 \cdot 3) = 3 \times 12$ array C according to the rule

$$C[i, j] = B[i \bmod 1, j \bmod 4, i \bmod 3, j \bmod 3].$$

This gives

$$C = \begin{bmatrix} -1 & -\mathrm{i} & -\mathrm{j} & \mathrm{k} & -1 & \mathrm{i} & -\mathrm{j} & -\mathrm{k} & -1 & -\mathrm{i} & \mathrm{j} & \mathrm{k} \\ 1 & -\mathrm{i} & -\mathrm{j} & \mathrm{k} & 1 & -\mathrm{i} & \mathrm{j} & -\mathrm{k} & 1 & \mathrm{i} & -\mathrm{j} & \mathrm{k} \\ -1 & \mathrm{i} & \mathrm{j} & \mathrm{k} & -1 & -\mathrm{i} & -\mathrm{j} & -\mathrm{k} & -1 & -\mathrm{i} & -\mathrm{j} & \mathrm{k} \end{bmatrix}.$$

It can be checked that

$$\mathbf{AC}_R(C) = \mathbf{AC}_L(C) = \begin{bmatrix} 36 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

so C is perfect.

5.7 Discussion

It was the author of this thesis who provided the translation and proof of inflation given in Section 5.3 into array terms. We gave a series of talks about this, and our work appears in the joint paper [4]; we have provided our own account of that work in Section 5.5, along with our own example in Section 5.6. The inflation proof in Section 5.4 does not take into account the non-commutativity of quaternion multiplication, thus calling for the new proof in Section 5.5.

Chapter 6

An Algebra of Arrays

6.1 Introduction

As mentioned in Section 5.1, Arasu and de Launey [1] used Laurent polynomials to represent arrays in their recursive constructions of perfect and almost perfect arrays. The laws of polynomial algebra can be used to express manipulations common to arrays; for example, $A(x, y)A^*(x^{-1}, y^{-1}) \bmod x^m - 1, y^n - 1$ represents the (right-shifted) $m \times n$ autocorrelation array of the array represented by $A(x, y)$. (As before, $*$ denotes complex conjugation of a polynomial.)

Typically, arrays are manipulated in terms of their elements, and thus autocorrelation calculations and proofs can be very complicated. One of the strengths of using polynomials is that they allow arrays to be manipulated as whole objects, rather than seen in terms of their elements.

In this chapter, we present a new way of manipulating arrays as whole objects, but without using polynomial theory. Specifically, we view array convolution as a commutative and associative multiplicative operation, and we introduce special functions, called index functions, to represent shifting, reversal, and other array operations. The result is an algebra of arrays which can be used to state and prove

some of the constructions in [1] in a much more direct way than polynomials allow. Due to the more general nature of index functions (which we introduce in Section 6.2), our algebraic approach allows one to describe more array operations than are possible using the polynomial approach.

In Section 6.6, we reconsider the polynomial construction in [1, Lemma 25] as an example. We demonstrate how this construction can be realised by a few basic array operations using index functions and array convolution. As a consequence, we make clear that the underlying alphabet of the arrays is preserved—which is not immediate from the proof via the polynomial method.

It has already been shown in [20, pp. 22–23] that convolution of length n sequences corresponds directly to multiplication of Laurent polynomials in $\mathbb{C}[x, x^{-1}]$ modulo $x^n - 1$. Specifically, if we represent the length n sequences A and B as polynomials by writing

$$A(x) = A[0] + A[1]x + \cdots + A[n-1]x^{n-1}$$

and

$$B(x) = B[0] + B[1]x + \cdots + B[n-1]x^{n-1},$$

then the polynomial product $A(x)B(x)$ taken modulo $x^n - 1$ represents the convolution sequence $A \circledast B$. In other words,

$$A(x)B(x) \equiv (A \circledast B)[0] + (A \circledast B)[1]x + \cdots + (A \circledast B)[n-1]x^{n-1} \pmod{x^n - 1}.$$

(Recall that $(A \circledast B)[k] = A \bullet (B^{s_k})^r$.) This extends immediately to two indeterminates, and thus it can be similarly shown that polynomial multiplication modulo $x^m - 1, y^n - 1$ corresponds directly to convolution of $m \times n$ arrays.

Writing $\mathbb{A}(m, n)$ to mean the set of all $m \times n$ arrays with complex entries,

then it follows that array convolution can be seen as a multiplication on $\mathbb{A}(m, n)$. Traditional matrix multiplication is not defined between two $m \times n$ arrays (unless $m = n$), and for the rest of this chapter we will agree not to use traditional matrix multiplication at all. In this chapter only, for keeping the notation simple, we write unambiguously AB for the convolution $A \circledast B$. We will prove in Section 6.4 that array convolution is an associative and distributive operation; other axioms of a standard multiplicative operation will be proved as needed.

The theory and results of this chapter appear in our paper [13].

6.2 Index Functions

Definition 6.1. *An* index function *is a function of the form* $f : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z} \times \mathbb{Z}$.

If $A \in \mathbb{A}(m, n)$ and f is an index function, then $A^f \in \mathbb{A}(m, n)$ is the array with entries

$$A^f[i, j] = A[f(i, j)].$$

If f and g are index functions and fg denotes the composition (with g applied first), then $(A^f)^g = A^{fg}$. It is important to note that $(A^f)^g$ means applying f to the array A before g.

Index functions behave well with respect to addition and scalar multiplication of arrays; for example, $(A + B)^f = A^f + B^f$ and that $(\lambda A)^f = \lambda(A^f)$ (we write the latter as λA^f). There is no general rule for $(A \circledast B)^f$; however, we describe some rules for specific index functions later.

For $i, j, u, v \in \mathbb{Z}$ write $(i, j) \equiv (u, v) \pmod{m, n}$ if $i \equiv u \pmod{m}$ and $j \equiv v \pmod{n}$. If f and g are index functions, then write $f \equiv g \pmod{m, n}$ if and only if $f(i, j) \equiv g(i, j) \pmod{m, n}$ for all integers i and j. It is immediate that $f \equiv g \pmod{m, n}$ implies $A^f = A^g$ for any $A \in \mathbb{A}(m, n)$.

Definition 6.2. *An index function f is* (m, n)-invertible *if it has an* (m, n)-inverse*, that is, an index function g such that $fg \equiv gf \equiv 1 \pmod{m, n}$, where* 1 *denotes the identity index function.*

Note that an (m, n)-inverse of f is not unique in general; it is clear that if f is invertible, then f^{-1} is an (m, n)-inverse for all m and n. Furthermore, if f is (m, n)-invertible, then f induces a bijection $\mathbb{Z}_m \times \mathbb{Z}_n \to \mathbb{Z}_m \times \mathbb{Z}_n$ given by $(i, j) \mapsto f(i, j) \pmod{m, n}$.

In Section 6.3, we write auto- and cross-correlation of arrays in terms of dot products. For this, the following theorem is very important.

Theorem 6.3. *Let $A \in \mathbb{A}(m, n)$, and let f be an (m, n)-invertible index function. If g is an (m, n)-inverse of f, then $A^f \bullet B = A \bullet B^g$.*

Proof. We have that

$$A^f \bullet B = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} A^f[i, j]B[i, j] = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} A[f(i, j)]B[i, j].$$

Substituting $(k, l) = f(i, j)$ for $(i, j) \in \mathbb{Z}_m \times \mathbb{Z}_n$, we have $(i, j) \equiv g(k, l) \pmod{m, n}$ and can write

$$A[f(i, j)]B[i, j] = A[k, l]B[g(k, l)]$$

for all i and j. Since the map $(i, j) \mapsto f(i, j) = (k, l) \pmod{m, n}$ is a bijection $\mathbb{Z}_m \times \mathbb{Z}_n \to \mathbb{Z}_m \times \mathbb{Z}_n$, it follows that

$$\begin{aligned} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} A[f(i, j)]B[i, j] &= \sum_{k=0}^{m-1} \sum_{l=0}^{n-1} A[k, l]B[g(k, l)] \\ &= \sum_{k=0}^{m-1} \sum_{l=0}^{n-1} A[k, l]B^g[k, l] = A \bullet B^g. \end{aligned}$$

□

6.3 An Algebra of Arrays

We define several index functions which represent some key ways of manipulating the elements of arrays (such as shifting and reversal). After showing how these index functions interact with each other under function composition, we use them to express the convolution AB as a special dot product. Finally, we express $\mathbf{AC}(A)$ and $\mathbf{CC}(A, B)$ as special convolutions, and we use these expressions to derive useful formulas involving index functions applied to $\mathbf{AC}(A)$ and $\mathbf{CC}(A, B)$.

Definition 6.4. *If $u, v \in \mathbb{Z}$, the index function $s_{u,v} : (i, j) \mapsto (i + u, j + v)$ is the* (u, v)-shift function.

The array $A^{s_{u,v}}$ denotes the array A with its columns and rows shifted cyclically u places up and v places left, respectively. In Chapter 2 we introduced the notation $A^{s_{u,v}}$ as a formal symbol. In this section we have given the symbol $s_{u,v}$ meaning by defining it as a special kind of index function.

Definition 6.5. *For $w \in \mathbb{Z}$, the index function $c_w : (i, j) \mapsto (i + wj, j)$ is a* column-shift function.

For $j = 0, \ldots, n - 1$, the jth column of A^{c_w} is the jth column of A shifted wj places upwards.

Definition 6.6. *The index function $r : (i, j) \mapsto (-i, -j)$ is the* reversing function.

The array A^r represents A with both its rows and columns reversed as sequences (vectors); note that the $(0, 0)$-entry is fixed.

Lemma 6.7. *The following relations are immediate:*

1. $s_{0,0} = c_0 = r^2 = 1$ *(where* 1 *denotes the identity index function).*
2. $s_{u,v}^{-1} = s_{-u,-v}$, $c_w^{-1} = c_{-w}$, *and* $r^{-1} = r$.

3. The following relations also hold:

$$s_{u,v}s_{x,y} = s_{u+x,v+y}, \qquad c_w c_z = c_{w+z},$$
$$s_{u,v}r = rs_{-u,-v}, \qquad c_w r = rc_w,$$
$$c_w s_{u,v} = s_{u+vw,v}c_w.$$

Recall that Definition 3.3 states that

$$(AB)[u,v] = A \bullet (B^{s_{u,v}})^r.$$

In the context of Chapter 3, the notation $(B^{s_{u,v}})^r$ used the symbols $s_{u,v}$ and r formally. In this chapter, we have viewed these symbols as specific index functions, while keeping their action on B unchanged. We can thus simply write

$$(AB)[u,v] = A \bullet B^{s_{u,v}r}, \tag{6.1}$$

where $s_{u,v}r$ is a composition of two index functions and has no meaning in the formal symbolism of Chapter 3.

The next theorem shows how $s_{u,v}$, c_w, and r behave when applied to the convolution AB.

Theorem 6.8. *If $A, B \in \mathbb{A}(m,n)$ and $u, v, w, x, y \in \mathbb{Z}$, then*

(1) $(AB)^{s_{u+x,v+y}} = A^{s_{u,v}}B^{s_{x,y}}$,

(2) $(AB)^{c_w} = A^{c_w}B^{c_w}$,

(3) $(AB)^r = A^r B^r$.

Proof. To prove (1), note that

$$\begin{aligned}(AB)^{s_{u+x,v+y}}[i,j] &= (AB)[i+u+x, j+v+y] = A \bullet B^{s_{i+u+x,j+v+y}r} \\ &= A \bullet B^{s_{i,j}s_{u,v}s_{x,y}r} = A \bullet (B^{s_{x,y}})^{s_{i,j}rs_{-u,-v}} \\ &= A^{s_{u,v}} \bullet (B^{s_{x,y}})^{s_{i,j}r} = A^{s_{u,v}}B^{s_{x,y}}[i,j]\end{aligned}$$

where Equation (6.1) is used in the final equality. To prove (2), note that

$$\begin{aligned}A^{c_w}B^{c_w}[i,j] &= A^{c_w} \bullet (B^{c_w})^{s_{i,j}r} = A \bullet B^{c_w s_{i,j}rc_{-w}} \\ &= A \bullet B^{s_{i+wj,j}c_w c_{-w}r} = A \bullet B^{s_{i+wj,j}r} \\ &= (AB)[i+wj, j] = (AB)^{c_w}[i,j].\end{aligned}$$

Part (3) is $A^rB^r[i,j] = A^r \bullet (B^r)^{s_{i,j}r} = A \bullet B^{rs_{i,j}r^2} = A \bullet B^{s_{-i,-j}r} = (AB)[-i,-j] = (AB)^r[i,j]$. □

The next theorem allows us to recast the auto- and cross-correlation arrays as special kinds of convolutions; Corollary 6.10 describes the behaviour of $\mathbf{AC}(A)$ when applying $s_{u,v}$, c_w and r to A.

Theorem 6.9. *If $A, B \in \mathbb{A}(m,n)$, then*

$$\mathbf{CC}(A,B) = A^rB^*;$$

in particular, $\mathbf{AC}(A) = A^rA^*$.

Proof. We have that

$$\mathbf{CC}(A,B)[k,l] = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} A[k,l]B^*[i+k, j+l] = A \bullet (B^*)^{s_{k,l}}.$$

Also, $(A^rB^*)[k,l] = A^r \bullet (B^*)^{s_{k,l}r} = A \bullet (B^*)^{s_{k,l}r^2} = A \bullet (B^*)^{s_{k,l}}$. □

Corollary 6.10. *If $A \in \mathbb{A}(m,n)$ and $u, v, w \in \mathbb{Z}$, then*

$$\mathbf{AC}(A^{s_{u,v}}) = \mathbf{AC}(A), \quad \mathbf{AC}(A^{c_w}) = \mathbf{AC}(A)^{c_w}, \quad \textit{and} \quad \mathbf{AC}(A^r) = \mathbf{AC}(A)^r.$$

Proof. This follows directly from

$$\begin{aligned}
\mathbf{AC}(A^{s_{u,v}}) &= (A^{s_{u,v}})^r (A^{s_{u,v}})^* = (A^r)^{s_{-u,-v}} (A^*)^{s_{u,v}} = A^r A^* = \mathbf{AC}(A), \\
\mathbf{AC}(A^{c_w}) &= (A^{c_w})^r (A^{c_w})^* = (A^r)^{c_w} (A^*)^{c_w} = (A^r A^*)^{c_w} = \mathbf{AC}(A)^{c_w}, \quad \text{and} \\
\mathbf{AC}(A^r) &= (A^r)^r (A^r)^* = (A^r A^*)^r = \mathbf{AC}(A)^r.
\end{aligned}$$

□

We use the formula $\mathbf{CC}(A,B) = A^r B^*$ to describe a relation between $\mathbf{CC}(A,B)$ and $\mathbf{CC}(B,A)$. The proof of the following theorem uses that array convolution is commutative; this follows easily from

$$(AB)[k,l] = A \bullet B^{s_{k,l}r} = A^r \bullet B^{s_{k,l}} = A^{r s_{-k,-l}} \bullet B = A^{s_{k,l}r} \bullet B = B \bullet A^{s_{k,l}r} = (BA)[k,l].$$

Theorem 6.11. *If $A, B \in \mathbb{A}(m,n)$, then*

$$\mathbf{CC}(B,A) = [\mathbf{CC}(A,B)^*]^r \quad \textit{and} \quad \mathbf{AC}(\mathbf{CC}(A,B)) = [\mathbf{AC}(A)\mathbf{AC}(B)]^r.$$

Proof. This follows from

$$[\mathbf{CC}(B,A)^*]^r = [(A^r B^*)^*]^r = [(A^*)^r B]^r = A^* B^r = B^r A^* = \mathbf{CC}(B,A)$$

and

$$\begin{aligned}
\mathbf{AC}(\mathbf{CC}(A,B)) &= \mathbf{AC}(A^r B^*) = (A^r B^*)^r (A^r B^*)^* = A (B^*)^r (A^*)^r B \\
&= A (A^*)^r B (B^*)^r = (A^r A^*)^r (B^r B^*)^r = [\mathbf{AC}(A)\mathbf{AC}(B)]^r.
\end{aligned}$$

□

6.4 Associativity and Distributivity of Array Convolution

Using the notation from Sections 6.2 and 6.3, we now prove associativity and distributivity of array convolution. Our proof of associativity involves the following technical lemma, which is of interest in its own right.

Lemma 6.12. *If* $A, B, C \in \mathbb{A}(m, n)$*, then* $(AB) \bullet C = A \bullet (B^r C)$.

Proof. Note that

$$(AB) \bullet C = \sum_{k=0}^{m-1} \sum_{l=0}^{n-1} (AB)[k, l]C[k, l] = \sum_{k=0}^{m-1} \sum_{l=0}^{n-1} (A \bullet B^{s_{k,l}r})C[k, l]$$

and

$$\begin{aligned} A \bullet (B^r C) &= \sum_{k=0}^{m-1} \sum_{l=0}^{n-1} A[k, l](B^r C)[k, l] = \sum_{k=0}^{m-1} \sum_{l=0}^{n-1} A[k, l](B^r \bullet C^{s_{k,l}r}) \\ &= \sum_{k=0}^{m-1} \sum_{l=0}^{n-1} A[k, l](B^{s_{-k,-l}} \bullet C). \end{aligned}$$

Thus, it suffices to show that

$$\sum_{k=0}^{m-1} \sum_{l=0}^{n-1} (A \bullet B^{s_{k,l}r})C[k, l] = \sum_{k=0}^{m-1} \sum_{l=0}^{n-1} A[k, l](B^{s_{-k,-l}} \bullet C). \tag{6.2}$$

We do this by expanding the dot product on the left-hand side of (6.2) into a double sum, and then switching the order of the quadruple sum that ensues. Starting on

the left-hand side of (6.2),

$$\begin{aligned}
\sum_{k=0}^{m-1}\sum_{l=0}^{n-1}(A \bullet B^{s_{k,l}r})C[k,l] &= \sum_{k=0}^{m-1}\sum_{l=0}^{n-1}\left(\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}A[i,j]B^{s_{k,l}r}[i,j]\right)C[k,l] \\
&= \sum_{k=0}^{m-1}\sum_{l=0}^{n-1}\left(\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}A[i,j]B[k-i,l-j]\right)C[k,l] \\
&= \sum_{i=0}^{m-1}\sum_{j=0}^{n-1}A[i,j]\left(\sum_{k=0}^{m-1}\sum_{l=0}^{n-1}B[k-i,l-j]C[k,l]\right) \\
&= \sum_{i=0}^{m-1}\sum_{j=0}^{n-1}A[i,j]\left(\sum_{k=0}^{m-1}\sum_{l=0}^{n-1}B^{s_{-i,-j}}[k,l]C[k,l]\right) \\
&= \sum_{i=0}^{m-1}\sum_{j=0}^{n-1}A[i,j](B^{s_{-i,-j}} \bullet C). \qquad \square
\end{aligned}$$

Theorem 6.13. *Array convolution is associative.*

Proof. If $A, B, C \in \mathbb{A}(m,n)$, then $((AB)C)[k,l] = (AB) \bullet C^{s_{k,l}r}$ and

$$(A(BC))[k,l] = ((BC)A)[k,l] = (BC) \bullet A^{s_{k,l}r} = A^{s_{k,l}r} \bullet (BC).$$

Thus it suffices to show that

$$(AB) \bullet C^{s_{k,l}r} = A^{s_{k,l}r} \bullet (BC).$$

Note that $(AB) \bullet C^{s_{k,l}r} = (AB)^{s_{k,l}r} \bullet C = (A^{s_{k,l}r}B^r) \bullet C = A^{s_{k,l}r} \bullet (BC)$, where the last equality follows from Lemma 6.12. $\square$

Theorem 6.14. *Array convolution is distributive.*

Proof. Let $A, B, C \in \mathbb{A}(m, n)$. Then

$$\begin{aligned}
(A(B+C))[k,l] &= \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} A[i,j](B+C)[k-i,l-j] \\
&= \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} A[i,j](B[k-i,l-j]+C[k-i,l-j]) \\
&= \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} (A[i,j]B[k-i,l-j]) + (A[i,j]C[k-i,l-j]) \\
&= \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} A[i,j]B[k-i,l-j] + \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} A[i,j]C[k-i,l-j] \\
&= (AB)[k,l] + (AC)[k,l] \\
&= (AB+AC)[k,l],
\end{aligned}$$

proving that $A(B+C) = AB + AC$. The case $(A+B)C = AC + BC$ follows from commutativity. □

6.5 Interleaving Two Arrays

Interleaving of sequences of equal length was introduced in [8], and basically is a method of merging two sequences to a new sequence of double the length. Interleaving of arrays can be done in two main ways: by interleaving the rows of each array, or by interleaving the columns of each array. We will use rows rather than columns for this thesis.

Definition 6.15. *For $A, B \in \mathbb{A}(m, n)$ let $A \wr B$ be the array in $\mathbb{A}(2m, n)$ formed by interleaving the rows of A and B, starting with the first row of A.*

In this section, we derive a notation which allows us to express $A \wr B$ in terms of $\mathbb{C}$-algebra operations and index functions. We then give a formula expressing $\mathbf{AC}(A \wr B)$ in terms of $\mathbf{AC}(A)$, $\mathbf{AC}(B)$, $\mathbf{CC}(A, B)$, and $\mathbf{CC}(B, A)$.

Definition 6.16. *For $A \in \mathbb{A}(m,n)$ and $k,l \geq 1$, let $A_{(k,l)} \in \mathbb{A}(km,ln)$ be the array whose (i,j)-entry is $A[i/k, j/l]$ if $k \mid i$ and $l \mid j$, and 0 otherwise.*

Essentially, $A_{(k,l)}$ contains the entries of A spaced out by multiples of k vertically and multiples of l horizontally, with zeroes inserted in between. Alternatively, $A_{(k,l)}$ is the Kronecker product of the $m \times n$ matrix A with the $k \times l$ zero matrix with a 1 in the $(0,0)$-position.

It is immediate that $(A^*)_{(k,l)} = (A_{(k,l)})^*$ (we sometimes write this as $A^*_{(k,l)}$), that $(A + B)_{(k,l)} = A_{(k,l)} + B_{(k,l)}$, and that $(\lambda A)_{(k,l)} = \lambda A_{(k,l)}$ for $\lambda \in \mathbb{C}$. If f is an index function, we will sometimes write $A^f_{(k,l)}$ to mean $(A_{(k,l)})^f$, rather than $(A^f)_{(k,l)}$.

The following theorem explains how $\cdot_{(k,l)}$ behaves with respect to dot products, the index functions $s_{u,v}$ and r, and convolution.

Theorem 6.17. *If $A, B \in \mathbb{A}(m,n)$, $k,l \geq 1$, and $u,v \in \mathbb{Z}$, then*

(1) $A_{(k,l)} \bullet B_{(k,l)} = A \bullet B$,

(2) $(A^{s_{u,v}})_{(k,l)} = (A_{(k,l)})^{s_{uk,vl}}$,

(3) $(A^r)_{(k,l)} = (A_{(k,l)})^r$,

(4) $(AB)_{(k,l)} = A_{(k,l)}B_{(k,l)}$.

Proof. Part (1) follows from

$$\begin{aligned} A_{(k,l)} \bullet B_{(k,l)} &= \sum_{i=0}^{km-1} \sum_{j=0}^{ln-1} A_{(k,l)}[i,j] B_{(k,l)}[i,j] \\ &= \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} A_{(k,l)}[ki, lj] B_{(k,l)}[ki, lj] \\ &= \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} A[i,j] B[i,j] = A \bullet B. \end{aligned}$$

To prove (2), suppose first that $i = ki_0$ and $j = lj_0$. Then

$$\begin{aligned}(A_{(k,l)})^{s_{uk,vl}}[i,j] &= A_{(k,l)}[ki_0 + ku, lj_0 + lv] = A[i_0 + u, j_0 + v] \\ &= A^{s_{u,v}}[i_0, j_0] = (A^{s_{u,v}})_{(k,l)}[i,j].\end{aligned}$$

Now suppose that $k \nmid i$ or $l \nmid j$. Then $k \nmid i + ku$ or $l \nmid j + lv$, or both, thus

$$(A_{(k,l)})^{s_{ku,lv}}[i,j] = A_{(k,l)}[i + ku, j + lv] = 0.$$

Finally, it is immediate that $(A^{s_{u,v}})_{(k,l)}[i,j] = 0$. To prove (3), note that if $k \mid i$ and $l \mid j$, then

$$(A^r)_{(k,l)}[i,j] = A^r[i/k, j/l] = A[(-i)/k, (-j)/l] = A_{(k,l)}[-i,-j] = (A_{(k,l)})^r[i,j].$$

If $k \nmid i$ or $l \nmid j$, then $(A^r)_{(k,l)}[i,j] = 0 = A_{(k,l)}[-i,-j] = (A_{(k,l)})^r[i,j]$.

Finally, we use (1), (2), and (3) to prove (4). Suppose that $i = ki_0$ and $j = lj_0$. Then

$$\begin{aligned}(A_{(k,l)}B_{(k,l)})[i,j] &= (A_{(k,l)}B_{(k,l)})[ki_0, lj_0] = A_{(k,l)} \bullet (B_{(k,l)})^{s_{ki_0,lj_0}{}^r} \\ &= A_{(k,l)} \bullet (B^{s_{i_0,j_0}{}^r})_{(k,l)} = A \bullet B^{s_{i_0,j_0}{}^r} \\ &= (AB)[i_0, j_0] = (AB)_{(k,l)}[i,j].\end{aligned}$$

Now suppose that $k \nmid i$ or $l \nmid j$. Then it is immediate that $(AB)_{(k,l)}[i,j] = 0$. Furthermore,

$$\begin{aligned}(A_{(k,l)}B_{(k,l)})[i,j] &= A_{(k,l)} \bullet (B_{(k,l)})^{s_{i,j}{}^r} \\ &= \sum_{u=0}^{km-1} \sum_{v=0}^{ln-1} A_{(k,l)}[u,v] B_{(k,l)}[i-u, j-v] \\ &= \sum_{u=0}^{m-1} \sum_{v=0}^{n-1} A_{(k,l)}[ku,lv] B_{(k,l)}[i-ku, j-lv].\end{aligned}$$

Since $k \nmid i - ku$ or $l \nmid j - lv$, we have that $B_{(k,l)}[i - ku, j - lv] = 0$ for all u and v. Hence $(A_{(k,l)}B_{(k,l)})[i, j] = 0 = (AB)_{(k,l)}[i, j]$. □

The following lemma expresses the interleaved array $A \wr B$ in terms of A and B by using $\mathbb{C}$-algebra operations and index functions.

Lemma 6.18. *If $A, B \in \mathbb{A}(m, n)$, then $A \wr B = A_{(2,1)} + B^{s_{-1,0}}_{(2,1)}$.*

Proof. The rows of A and B appear as every second row of $A_{(2,1)}$ and $B_{(2,1)}$, with rows of zeroes placed in between. Applying $s_{-1,0}$ to $B_{(2,1)}$ shifts the rows of B down by one place, which puts them directly in line with the inserted zero rows of $A_{(2,1)}$, and hence in between the rows of A. Finally, adding $A_{(2,1)}$ to $B^{s_{-1,0}}_{(2,1)}$ creates a single interleaved array, which is equal to $A \wr B$. □

The final theorem of this section expresses the autocorrelation $\mathbf{AC}(A \wr B)$ in terms of $\mathbf{AC}(A)$, $\mathbf{AC}(B)$, $\mathbf{CC}(A, B)$, and $\mathbf{CC}(B, A)$. This allows us to deal with autocorrelations of an interleaved array in terms of the $\mathbb{C}$-algebra $\mathbb{A}(m, n)$ and index functions.

Theorem 6.19. *If $A, B \in \mathbb{A}(m, n)$, then*

$$\mathbf{AC}(A \wr B) = \mathbf{AC}(A)_{(2,1)} + \mathbf{AC}(B)_{(2,1)} + \mathbf{CC}(A, B)^{s_{-1,0}}_{(2,1)} + \mathbf{CC}(B, A)^{s_{1,0}}_{(2,1)}.$$

Proof. Note that

$$\begin{aligned}
\mathbf{AC}(A \wr B) &= (A_{(2,1)} + B^{s_{-1,0}}_{(2,1)})^r (A_{(2,1)} + B^{s_{-1,0}}_{(2,1)})^* \\
&= (A^r_{(2,1)} + B^{s_{-1,0}r}_{(2,1)})(A^*_{(2,1)} + (B^*)^{s_{-1,0}}_{(2,1)}) \\
&= A^r_{(2,1)}A^*_{(2,1)} + B^{s_{-1,0}r}_{(2,1)}(B^*)^{s_{-1,0}}_{(2,1)} + A^r_{(2,1)}(B^*)^{s_{-1,0}}_{(2,1)} + B^{s_{-1,0}r}_{(2,1)}A^*_{(2,1)}.
\end{aligned}$$

We consider each of the four terms in the last equality separately. First,

$$A^r_{(2,1)}A^*_{(2,1)} = (A^r)_{(2,1)}(A^*)_{(2,1)} = (A^rA^*)_{(2,1)} = \mathbf{AC}(A)_{(2,1)}.$$

Second,

$$B_{(2,1)}^{s_{-1,0}r}(B^*)_{(2,1)}^{s_{-1,0}} = B_{(2,1)}^{rs_{1,0}}(B^*)_{(2,1)}^{s_{-1,0}} = B_{(2,1)}^{r}(B^*)_{(2,1)} = \mathbf{AC}(B)_{(2,1)}.$$

Third,

$$A_{(2,1)}^{r}(B^*)_{(2,1)}^{s_{-1,0}} = (A^rB^*)_{(2,1)}^{s_{-1,0}} = \mathbf{CC}(A,B)_{(2,1)}^{s_{-1,0}}.$$

Finally,

$$B_{(2,1)}^{s_{-1,0}r}A_{(2,1)}^{*} = B_{(2,1)}^{rs_{1,0}}A_{(2,1)}^{*} = (B^rA^*)_{(2,1)}^{s_{1,0}} = \mathbf{CC}(B,A)_{(2,1)}^{s_{1,0}}. \qquad \square$$

Interleaving can be used to construct larger arrays out of smaller ones. Another operation which can also be used along these lines is concatenation, which was defined in Definition 2.9. Both operations will be used in the example in the Section 6.6.

6.6 A Recursive Construction of Almost Perfect Arrays

Let $\mathbf{1} = \mathbf{1}_{m,n}$ be the multiplicative identity of $\mathbb{A}(m,n)$, that is, the $m \times n$ array with a 1 in the $(0,0)$-position and zeroes elsewhere. It is immediate that $\mathbf{1}^{s_{-u,-v}}$ is the $m \times n$ array with a 1 in the (u,v) position and zeroes everywhere else. It is also immediate that $(\mathbf{1}_{m,n})_{(k,l)} = \mathbf{1}_{mk,nl}$ for all k and l.

Let $A \in \mathbb{A}(m,n)$, whose alphabet consists of roots of unity. Then it is clear that A is perfect if and only if $\mathbf{AC}(A) = mn\mathbf{1}$. If m is even, then A is almost perfect if and only if $\mathbf{AC}(A) = mn(\mathbf{1} - \mathbf{1}^{s_{-m/2,0}}) = mn(\mathbf{1} - \mathbf{1}^{s_{m/2,0}})$. (Because $(m/2,0)$ is precisely halfway down the first column, shifting either up or down will reach the same place, so $\mathbf{1}^{s_{-m/2,0}} = \mathbf{1}^{s_{m/2,0}}$.)

If we concatenate two copies of an $m \times n$ array A side-by-side, then the auto-

correlation array of the concatenation will be the concatenation of two copies of $\mathbf{AC}(A)$, with every entry multiplied by two. (A generalisation of this is proved in Lemma 7.9.) It follows that if A is an almost perfect $m \times n$ array (m even) over roots of unity and if B is the $m \times 2n$ array created by concatenating two copies of A side by side, then $\mathbf{AC}(B) = 2mn(\mathbf{1} - \mathbf{1}^{s_{m/2,0}} + \mathbf{1}^{s_{0,n}} - \mathbf{1}^{s_{m/2,n}})$, where this time $\mathbf{1} = \mathbf{1}_{m,2n}$.

The result [1, Lemma 25] takes an almost perfect quaternary array and uses polynomials to construct another almost perfect quaternary array with both the number of rows and columns doubled. Repeating this construction gives an infinite recursive construction of almost perfect quaternary arrays.

In the next theorem, we prove [1, Lemma 25] for arrays over arbitrary roots of unity, using the new algebra presented in this thesis.

Theorem 6.20. *Let m and n be positive integers such that $m/(2n) = w \in \mathbb{Z}$. Let A be an almost perfect $m \times n$ array over arbitrary roots of unity. Define B to be the $m \times 2n$ array created by concatenating two copies of A side-by-side, and let $C = B^{c_w}$. Then the interleaved $2m \times 2n$ array $D = B \wr C$ is almost perfect, and the entries are in the same alphabet as A.*

Proof. Write $\mathbf{1} = \mathbf{1}_{m,2n}$ and note that $\mathbf{AC}(B) = 2mn(\mathbf{1} - \mathbf{1}^{s_{m/2,0}} + \mathbf{1}^{s_{0,n}} - \mathbf{1}^{s_{m/2,n}})$. From Corollary 6.10, it follows that

$$\begin{aligned}\mathbf{AC}(C) = \mathbf{AC}(B)^{c_w} &= 2mn(\mathbf{1} - \mathbf{1}^{s_{m/2,0}} + \mathbf{1}^{s_{0,n}} - \mathbf{1}^{s_{m/2,n}})^{c_w} \\ &= 2mn(\mathbf{1}^{c_w} - \mathbf{1}^{s_{m/2,0}c_w} + \mathbf{1}^{s_{0,n}c_w} - \mathbf{1}^{s_{m/2,n}c_w}).\end{aligned}$$

Now c_w shifts the jth column of an array by jw. It thus fixes the 0th column, and since $\mathbf{1}$ and $\mathbf{1}^{s_{m/2,0}}$ only contain nonzero entries in the 0th column, it follows that $\mathbf{1}^{c_w} = \mathbf{1}$ and $\mathbf{1}^{s_{m/2,0}c_w} = \mathbf{1}^{s_{m/2,0}}$.

Similarly, $\mathbf{1}^{s_{0,n}}$ and $\mathbf{1}^{s_{m/2,n}}$ only contain nonzero entries in the nth column (recall

that we are dealing with $m \times 2n$ arrays). Thus c_w shifts the nth column of these arrays up by $nw = m/2$ places, and does nothing to the other columns. It follows that $\mathbf{1}^{s_{0,n}c_w} = \mathbf{1}^{s_{m/2,n}}$ and $\mathbf{1}^{s_{m/2,n}c_w} = \mathbf{1}^{s_{0,n}}$, so

$$\mathbf{AC}(C) = 2mn(\mathbf{1} - \mathbf{1}^{s_{m/2,0}} - \mathbf{1}^{s_{0,n}} + \mathbf{1}^{s_{m/2,n}}).$$

Let $\mathbf{0}$ denote the array in $\mathbb{A}(m, 2n)$ whose entries are all zeroes. If we set

$$X = \mathbf{1} - \mathbf{1}^{s_{m/2,0}} \qquad \text{and} \qquad Y = \mathbf{1}^{s_{0,n}} - \mathbf{1}^{s_{m/2,n}},$$

then $\mathbf{AC}(B) = 2mn(X + Y)$ and $\mathbf{AC}(C) = 2mn(X - Y)$. Furthermore,

$$\begin{aligned} X^2 = (\mathbf{1} - \mathbf{1}^{s_{m/2,0}})^2 &= \mathbf{11} - 2\mathbf{1}\mathbf{1}^{s_{m/2,0}} + \mathbf{1}^{s_{m/2,0}}\mathbf{1}^{s_{m/2,0}} \\ &= \mathbf{1} - 2\mathbf{1}^{s_{m/2,0}} + \mathbf{1}^{s_{m,0}} = 2\mathbf{1} - 2\mathbf{1}^{s_{m/2,0}} \end{aligned}$$

and

$$\begin{aligned} Y^2 = (\mathbf{1}^{s_{0,n}} - \mathbf{1}^{s_{m/2,n}})^2 &= \mathbf{1}^{s_{0,n}}\mathbf{1}^{s_{0,n}} - 2\mathbf{1}^{s_{0,n}}\mathbf{1}^{s_{m/2,n}} + \mathbf{1}^{s_{m/2,n}}\mathbf{1}^{s_{m/2,n}} \\ &= \mathbf{1}^{s_{0,2n}} - 2\mathbf{1}^{s_{m/2,2n}} + \mathbf{1}^{s_{m,2n}} = 2\mathbf{1} - 2\mathbf{1}^{s_{m/2,0}} = X^2, \end{aligned}$$

so $\mathbf{AC}(B)\mathbf{AC}(C) = 4m^2n^2(X + Y)(X - Y) = 4m^2n^2(X^2 - Y^2) = \mathbf{0}$.

It follows from Theorem 6.11 that $\mathbf{AC}(\mathbf{CC}(B, C)) = \mathbf{0}$, hence $\mathbf{CC}(B, C) = \mathbf{0}$. Theorem 6.11 also implies that $\mathbf{CC}(C, B) = \mathbf{0}$. Finally, from Theorem 6.19 it follows that

$$\begin{aligned} \mathbf{AC}(D) &= \mathbf{AC}(B)_{(2,1)} + \mathbf{AC}(C)_{(2,1)} + \mathbf{CC}(B, C)^{s_{-1,0}}_{(2,1)} + \mathbf{CC}(C, B)^{s_{1,0}}_{(2,1)} \\ &= 2mn(X + Y)_{(2,1)} + 2mn(X - Y)_{(2,1)} + \mathbf{0} + \mathbf{0} \\ &= 4mnX_{(2,1)} = 4mn(\mathbf{1} - \mathbf{1}^{s_{m/2,0}})_{(2,1)} = 4mn(\mathbf{1}_{(2,1)} - \mathbf{1}^{s_{m,0}}_{(2,1)}) \\ &= 4mn(\mathbf{1}_{2m,2n} - \mathbf{1}^{s_{m,0}}_{2m,2n}), \end{aligned}$$

so D is almost perfect. Note that D is constructed from A using nothing but concatenations, column shifts, and interleaving of rows. Since these operations alter only the arrangement of the entries and not the entries themselves, it follows that D is over the same alphabet as A. □

For clarity, we have simplified m, n and w in the above proof from the construction in [1]. For the generalisation, let $m = xu$ and $n = yv$ where x and y are powers of 2 such that $2y$ divides x, and u and v are arbitrary odd positive integers. Finally, set $w = xu/(2y)$. It is straightforward but technical to modify the proof above using these new values. (It may be helpful to keep in mind that since u and v are odd, then $u \pm 1$ and $v \pm 1$ are even.)

We end this section with a small example. Consider the 2×1 binary array

$$A = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

Then, following the construction given in Theorem 6.20, we get

$$B = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} \qquad \text{and} \qquad C = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.$$

Finally, interleaving the rows of B and C gives

$$D = \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ -1 & -1 \\ -1 & 1 \end{bmatrix} \qquad \text{with} \qquad \mathbf{AC}(D) = \begin{bmatrix} 8 & 0 \\ 0 & 0 \\ -8 & 0 \\ 0 & 0 \end{bmatrix},$$

which shows that D is almost perfect. In [1, Lemma 25], it is stated that D will be a quaternary array (a superset of binary arrays), but it is not obvious that the

binary alphabet will be preserved.

The 4×2 array D can be fed into Theorem 6.20 to give the 8×4 binary array

$$E = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \\ -1 & -1 & -1 & -1 \\ -1 & 1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ -1 & 1 & 1 & -1 \end{bmatrix} \quad \text{with} \quad \mathbf{AC}(E) = \begin{bmatrix} 32 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -32 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

which shows that the array is almost perfect. We can feed this array back into Theorem 6.20 continuously, giving a recursive construction of arbitrarily large almost perfect binary arrays. We can do the same thing for arrays over any number of roots of unity, giving an arbitrarily large recursive construction of almost perfect arrays over the same alphabet.

6.7 Discussion

The algebra of arrays in Sections 6.2–6.5 can easily be specialised to arrays of one dimension (that is, sequences), or generalised to arrays of more than two dimensions. This specialisation or generalisation follows in a canonical fashion.

Equations for interleaving more than two arrays are developed in Lemma 7.10 in the next chapter.

The index functions $s_{u,v}$, c_w and r all have representations using polynomial theory. For example, if $A(x, y)$ represents an array A, then $A(x, x^w y)$ represents the column-shifted array A^{c_w}. In the language of polynomials, manipulation of arrays

comes down to manipulation of the indeterminates x and y. Because there are only two laws $(x^m)^n = x^{mn}$ and $x^m x^n = x^{m+n}$, there are only a limited number of array operations that polynomials can describe. On the other hand, any modification of an array can be described as an application of an index function, and, as demonstrated in this chapter, algebraic methods can be used to manipulate such operations effectively.

Chapter 7

A Recursive Construction of n-Perfect Arrays

7.1 Introduction

In Section 6.6, we presented a recursive construction of almost perfect arrays. Recall that this construction involves taking a single almost perfect array, concatenating two copies of it side by side, and then forming two new arrays by taking every possible linear column shift of the concatenation. These column shifted arrays are then interleaved to form a new almost perfect array over the same alphabet as the old one, and with both the horizontal and vertical dimensions doubled. The construction can be applied repeatedly, giving infinitely large almost perfect arrays over a single alphabet.

In this chapter, we extend the concept of an almost perfect array. Recall from Definition 3.10 that an $m \times n$ almost perfect array has an autocorrelation array of all zeroes, except for mn in the $(0,0)$-entry and $-mn$ halfway down the first column. If we note that $\pm mn$ are the second roots of unity ± 1 scaled by mn, we can extend the concept of "almost perfect" by having an autocorrelation array of

all zeroes except for the nth roots of unity spaced evenly down the first column, all scaled by mn. We call an array with such autocorrelation "n-perfect". These arrays are defined formally in Section 7.3, and examples are given.

We also extend the recursive construction given in Section 6.6 to n-perfect arrays. Recall that there are two possible linear column shifts of a concatenation of two copies of an almost perfect array. It turns out that if we concatenate n copies of an n-perfect array side-by-side, then there are n possible linear column shifts of that concatenation. These n column shifted arrays are then interleaved to form one large n-perfect array, over the same alphabet as the original, and with both horizontal and vertical dimensions multiplied by n. The construction can be applied repeatedly to yield arbitrarily large n-perfect arrays, over the same alphabet. The construction itself is given in Section 7.4, along with several examples, and is later proved in Section 7.5.

7.2 Preliminaries

Let $n \geq 1$. Recall that a complex number λ is an *nth root of unity* if $\lambda^n = 1$. Furthermore, if $\lambda^m \neq 1$ for all $1 \leq m < n$ then λ is a *primitive* nth root of unity.

The nth roots of unity are typically written as $e^{2k\pi i/n}$ for $0 \leq k \leq n-1$. We call $e^{2\pi i/n}$ the *principal* nth root of unity. (Note that $e^{2\pi i/n}$ is always primitive.) The following lemma is a well-known result about roots of unity; it appears as Exercise 2.1 in [18].

Lemma 7.1. *Let $n \geq 2$, and let λ be any primitive nth root of unity. If $a \in \mathbb{Z}$, then*

$$\sum_{i=0}^{n-1} \lambda^{ai} = \begin{cases} n & \text{if } n \mid a \\ 0 & \text{if } n \nmid a. \end{cases}$$

Proof. Since $\lambda^k = \lambda^l$ if $k \equiv l \pmod{n}$, we can assume without loss of generality that $a \in \mathbb{Z}_n$. If $a = 0$, the sum is clearly n. Now suppose $a \neq 0$. Then $\lambda^a \neq 1$, since λ is primitive. The roots of $x^n - 1$ are precisely the nth roots of unity and it follows from

$$x^n - 1 = (x-1)(1 + x + \cdots + x^{n-1})$$

that the roots of $1 + x + \cdots + x^{n-1}$ are all the nth roots of unity except 1. Thus λ^a is a root of $1 + x + \cdots + x^{n-1}$, so $1 + (\lambda^a) + \cdots + (\lambda^a)^{n-1} = 0$, proving the sum. □

We write $[a/n]$ for the integral quotient of the division of a by n, where $a \in \mathbb{Z}$ and $n \geq 1$, so that

$$a = [a/n] \cdot n + (a \bmod n).$$

If we write

$$\delta_n(a,b) = \begin{cases} 0 & \text{if } (a \bmod n) + (b \bmod n) \in \mathbb{Z}_n \\ 1 & \text{otherwise} \end{cases}$$

then we get that

$$[(a+b)/n] = [a/n] + [b/n] + \delta_n(a,b)$$

and

$$(a+b) \bmod n = (a \bmod n) + (b \bmod n) - n\delta_n(a,b).$$

Note that $\delta_n(a,b) = \delta_n(c,d)$ if $a \equiv c \pmod{n}$ and $b \equiv d \pmod{n}$.

7.3 n-Perfect Arrays

Definition 7.2. *Let $n, u, v \geq 1$, and $\lambda = \mathrm{e}^{2\pi \mathrm{i}/n}$. An $nu \times v$ array is* n-perfect *if*

$$\mathbf{AC}(A)[i,j] = \begin{cases} nuv\lambda^{i/u} & \text{if } u \mid i \text{ and } j = 0 \\ 0 & \text{otherwise.} \end{cases}$$

An n-perfect array thus has an autocorrelation array with the nth roots of unity in sequence (scaled by the size of the array) equally spaced down the first column, and zeroes everywhere else.

Example 7.3. *The binary array*

$$A = \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ -1 & -1 \\ -1 & 1 \end{bmatrix} \qquad \textit{with} \qquad \mathbf{AC}(A) = \begin{bmatrix} 8 & 0 \\ 0 & 0 \\ -8 & 0 \\ 0 & 0 \end{bmatrix}$$

is 2*-perfect. Note that a* 2*-perfect array is identical to an almost perfect array.*

Example 7.4. *Let* $\lambda = \mathrm{e}^{2\pi \mathrm{i}/3}$ *be the principal* 3*rd root of unity. The array*

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & \lambda^2 & \lambda \\ 1 & \lambda & \lambda^2 \\ \lambda^2 & \lambda^2 & \lambda^2 \\ \lambda^2 & \lambda & 1 \\ \lambda^2 & 1 & \lambda \\ \lambda & \lambda & \lambda \\ \lambda & 1 & \lambda^2 \\ \lambda & \lambda^2 & 1 \end{bmatrix} \qquad \textit{with} \qquad \mathbf{AC}(A) = \begin{bmatrix} 27 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 27\lambda & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 27\lambda^2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

is 3*-perfect.*

7.4 Construction and Examples

Theorem 7.5. *Let $n, u, v \geq 1$ with $v \mid u$. Let A be an n-perfect array of dimensions $nu \times v$. Let B_0 be the $nu \times nv$ array created by concatenating n copies of A horizontally. For each $k = 1, \ldots, n-1$, define $B_k = B_0^{\zeta^{ku/v}}$. Finally, let C be the $n^2u \times nv$ array created by interleaving the rows of $B_0, \ldots, B_{n-1}$. Then C is n-perfect, and over the same alphabet as A.*

We now give examples of Theorem 7.5 for 2-, 3- and 4-perfect arrays.

Example 7.6. *The 2nd roots of unity are $1, -1$. The 2×1 binary array*

$$A = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \qquad \text{with} \qquad \mathbf{AC}(A) = \begin{bmatrix} 2 \\ -2 \end{bmatrix}$$

is 2-perfect. Following Theorem 7.5, we concatenate 2 copies of A side by side to get

$$B_0 = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}.$$

Now $n = 2$, $u = 1$ and $v = 1$ in this case, so the only B_k array to construct is B_1, which is B_0 with the jth column shifted up by $j1u/v = j$ places. This gives

$$B_1 = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.$$

Interleaving the rows of B_0 and B_1, starting with the first row of B_0, gives the 4×2 array

$$C = \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ -1 & -1 \\ -1 & 1 \end{bmatrix} \qquad \text{with} \qquad \mathbf{AC}(C) = \begin{bmatrix} 8 & 0 \\ 0 & 0 \\ -8 & 0 \\ 0 & 0 \end{bmatrix}.$$

Clearly, C is a 2-perfect binary array. We can apply Theorem 7.5 to C to obtain a 2-perfect 8×4 array, and continue in this fashion to obtain an infinite family of 2-perfect binary arrays.

Example 7.7. *Let $\lambda = \mathrm{e}^{2\pi \mathrm{i}/3}$ be the principal 3rd root of unity. Then the 3rd roots of unity are $1, \lambda, \lambda^2$ and the 3×1 array*

$$A = \begin{bmatrix} 1 \\ \lambda^2 \\ \lambda \end{bmatrix} \qquad \textit{with} \qquad \mathbf{AC}(A) = \begin{bmatrix} 3 \\ 3\lambda \\ 3\lambda^2 \end{bmatrix}$$

is 3-perfect. Concatenating 3 copies of A side by side gives

$$B_0 = \begin{bmatrix} 1 & 1 & 1 \\ \lambda^2 & \lambda^2 & \lambda^2 \\ \lambda & \lambda & \lambda \end{bmatrix}.$$

We have $n = 3$ and $u = v = 1$. We thus construct B_1 and B_2, where the jth column of B_k ($k = 1, 2$) is the jth column of B_0 shifted up by $jku/v = jk$ places. Thus

$$B_1 = \begin{bmatrix} 1 & \lambda^2 & \lambda \\ \lambda^2 & \lambda & 1 \\ \lambda & 1 & \lambda^2 \end{bmatrix} \qquad \textit{and} \qquad B_2 = \begin{bmatrix} 1 & \lambda & \lambda^2 \\ \lambda^2 & 1 & \lambda \\ \lambda & \lambda^2 & 1 \end{bmatrix}.$$

Interleaving the rows of B_0, B_1, B_2, starting with the first row of B_0, gives the 9×3 array

$$C = \begin{bmatrix} 1 & 1 & 1 \\ 1 & \lambda^2 & \lambda \\ 1 & \lambda & \lambda^2 \\ \lambda^2 & \lambda^2 & \lambda^2 \\ \lambda^2 & \lambda & 1 \\ \lambda^2 & 1 & \lambda \\ \lambda & \lambda & \lambda \\ \lambda & 1 & \lambda^2 \\ \lambda & \lambda^2 & 1 \end{bmatrix} \quad \textit{with} \quad \mathbf{AC}(C) = \begin{bmatrix} 27 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 27\lambda & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 27\lambda^2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Clearly, C is 3-perfect. We can apply Theorem 7.5 to C to obtain a 3-perfect 27×9 array, and continue in this fashion to obtain an infinite family of 3-perfect arrays, all over 3 roots of unity.

Example 7.8. *The 4th roots of unity are $1, \mathrm{i}, -1, -\mathrm{i}$. The 4×1 quaternary array*

$$A = \begin{bmatrix} 1 \\ -\mathrm{i} \\ -1 \\ \mathrm{i} \end{bmatrix} \quad \textit{with} \quad \mathbf{AC}(A) = \begin{bmatrix} 4 \\ 4\mathrm{i} \\ -4 \\ -4\mathrm{i} \end{bmatrix}$$

is 4-perfect. Concatenating 4 copies of A side by side gives

$$B_0 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -\mathrm{i} & -\mathrm{i} & -\mathrm{i} & -\mathrm{i} \\ -1 & -1 & -1 & -1 \\ \mathrm{i} & \mathrm{i} & \mathrm{i} & \mathrm{i} \end{bmatrix}.$$

We have $n = 4$ and $u = v = 1$. We thus construct B_1, B_2 and B_3, where the jth column of B_k ($k = 1, 2, 3$) is the jth column of B_0 shifted up by $jku/v = jk$ places.

Thus

$$B_1 = \begin{bmatrix} 1 & -\mathrm{i} & -1 & \mathrm{i} \\ -\mathrm{i} & -1 & \mathrm{i} & 1 \\ -1 & \mathrm{i} & 1 & -\mathrm{i} \\ \mathrm{i} & 1 & -\mathrm{i} & -1 \end{bmatrix}, B_2 = \begin{bmatrix} 1 & -1 & 1 & -1 \\ -\mathrm{i} & \mathrm{i} & -\mathrm{i} & \mathrm{i} \\ -1 & 1 & -1 & 1 \\ \mathrm{i} & -\mathrm{i} & \mathrm{i} & -\mathrm{i} \end{bmatrix} \text{ and } B_3 = \begin{bmatrix} 1 & \mathrm{i} & -1 & -\mathrm{i} \\ -\mathrm{i} & 1 & \mathrm{i} & -1 \\ -1 & -\mathrm{i} & 1 & \mathrm{i} \\ \mathrm{i} & -1 & -\mathrm{i} & 1 \end{bmatrix}.$$

Interleaving the rows of B_0, B_1, B_2, B_3*, starting with the first row of* B_0*, gives the* 16×4 *array*

$$C = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -\mathrm{i} & -1 & \mathrm{i} \\ 1 & -1 & 1 & -1 \\ 1 & \mathrm{i} & -1 & -\mathrm{i} \\ -\mathrm{i} & -\mathrm{i} & -\mathrm{i} & -\mathrm{i} \\ -\mathrm{i} & -1 & \mathrm{i} & 1 \\ -\mathrm{i} & \mathrm{i} & -\mathrm{i} & \mathrm{i} \\ -\mathrm{i} & 1 & \mathrm{i} & -1 \\ -1 & -1 & -1 & -1 \\ -1 & \mathrm{i} & 1 & -\mathrm{i} \\ -1 & 1 & -1 & 1 \\ -1 & -\mathrm{i} & 1 & \mathrm{i} \\ \mathrm{i} & \mathrm{i} & \mathrm{i} & \mathrm{i} \\ \mathrm{i} & 1 & -\mathrm{i} & -1 \\ \mathrm{i} & -\mathrm{i} & \mathrm{i} & -\mathrm{i} \\ \mathrm{i} & -1 & -\mathrm{i} & 1 \end{bmatrix} \quad \textit{with} \quad \mathbf{AC}(C) = \begin{bmatrix} 64 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 64\mathrm{i} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -64 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -64\mathrm{i} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Clearly, C *is* 4*-perfect. We can apply Theorem 7.5 to* C *to obtain a* 4*-perfect* 64×16 *array, and continue in this fashion to obtain an infinite family of* 4*-perfect*

quaternary arrays.

It is immediate that Examples 7.6, 7.7 and 7.8 can be generalised in a canonical way for arbitrary nth roots of unity; take $\lambda = e^{2\pi i/n}$ and start with the $n \times 1$ array

$$A = \begin{bmatrix} 1 \\ \lambda^{n-1} \\ \vdots \\ \lambda^1 \end{bmatrix}.$$

It is routine to verify that A is n-perfect.

7.5 Proof of the Construction

The proof of Theorem 7.5 will use several lemmas, which are proved below.

Lemma 7.9. *Let A be an $m \times n$ array, and let $k \geq 1$. If B is the horizontal concatenation of k copies of A, then $\mathbf{AC}(B)$ is the horizontal concatenation of k copies of $\mathbf{AC}(A)$, with every entry multiplied by a factor of k.*

Proof. For $(s,t) \in \mathbb{Z}_m \times \mathbb{Z}_{kn}$, we have $B[s,t] = A[s, t \bmod n]$, so

$$\begin{aligned} \mathbf{AC}(B)[s,t] &= \sum_{i=0}^{m-1} \sum_{j=0}^{kn-1} B[i,j] B^*[i+s, j+t] \\ &= \sum_{i=0}^{m-1} \sum_{j=0}^{kn-1} A[i, j \bmod n] A^*[i+s, (j+t) \bmod n] \\ &= \sum_{i=0}^{m-1} \sum_{j=0}^{kn-1} A[i, j \bmod n] A^*[i+s, j \bmod n + t \bmod n]. \end{aligned}$$

As j goes from 0 to $kn-1$, $j \bmod n$ goes from 0 to $n-1$ a total of k times. Thus

$$\mathbf{AC}(B)[s,t] = k\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} A[i,j]A^*[i+s, j+t \bmod n] = k\mathbf{AC}(A)[s, t \bmod n],$$

proving the lemma. □

Lemma 7.10. *Let $A_0, \ldots, A_{r-1}$ be a family of r arrays ($r \geq 1$), each of dimensions $m \times n$. Let B be the $rm \times n$ array created by interleaving the rows of $A_0, \ldots, A_{r-1}$, starting with the first row of A_0. Then, for $(k,l) \in \mathbb{Z}_{rm} \times \mathbb{Z}_n$,*

$$\mathbf{AC}(B)[k,l] = \sum_{t=0}^{r-1} \mathbf{CC}(A_t, A_{t+k})[[k/r] + \delta_r(t,k), l].$$

(We use the convention that $A_m = A_{m \bmod r}$ for all $m \in \mathbb{Z}$.) In particular, if $i \equiv 0 \pmod r$ then

$$\mathbf{AC}(B)[i,j] = \sum_{t=0}^{r-1} \mathbf{AC}(A_t)[[i/r], j].$$

Proof. Let $(k,l) \in \mathbb{Z}_{rm} \times \mathbb{Z}_n$. Then the (k,l)-entry of B is $A_k[[k/r], l]$ (where $A_k = A_{k \bmod r}$). Thus

$$\begin{aligned}
\mathbf{AC}(B)[k,l] &= \sum_{i=0}^{rm-1}\sum_{j=0}^{n-1} B[i,j]B^*[i+k, j+l] \\
&= \sum_{i=0}^{rm-1}\sum_{j=0}^{n-1} A_i[[i/r], j]A^*_{i+k}[[i+k/r], j+l] \\
&= \sum_{i=0}^{rm-1}\sum_{j=0}^{n-1} A_i[[i/r], j]A^*_{i+k}[[i/r] + [k/r] + \delta_r(i,k), j+l].
\end{aligned}$$

Now the map $(s,t) \mapsto sr+t$ is a bijection from $\mathbb{Z}_m \times \mathbb{Z}_r$ to $\mathbb{Z}_{rm}$. Thus, if we make the substitution $i = sr+t$, then $s = [i/r]$ and $t = i \bmod r$ by division with remainder.

This gives

$$\begin{aligned}\mathbf{AC}(B)[k,l] &= \sum_{t=0}^{r-1}\sum_{s=0}^{m-1}\sum_{j=0}^{n-1} A_{sr+t}[s,j]A^*_{sr+t+k}[s+[k/r]+\delta_r(sr+t,k),j+l] \\ &= \sum_{t=0}^{r-1}\sum_{s=0}^{m-1}\sum_{j=0}^{n-1} A_t[s,j]A^*_{t+k}[s+[k/r]+\delta_r(t,k),j+l] \\ &= \sum_{t=0}^{r-1} \mathbf{CC}(A_t,A_{t+k})[[k/r]+\delta_r(t,k),l]\end{aligned}$$

which proves the lemma. □

We now prove Theorem 7.5. We restate it below, for convenience.

Theorem (Restatement of Theorem 7.5)**.** *Let $n,u,v \geq 1$ with $v \mid u$. Let A be a n-perfect array of dimensions $nu \times v$. Let B_0 be the $nu \times nv$ array created by concatenating n copies of A horizontally. For each $k = 1,\ldots,n-1$, define $B_k = B_0^{Cku/v}$. Finally, let C be the $n^2u \times nv$ array created by interleaving the rows of $B_0,\ldots,B_{n-1}$. Then C is n-perfect, and over the same alphabet as A.*

Proof. Let $\lambda = \mathrm{e}^{2\pi\mathrm{i}/n}$ be the principal nth root of unity. Since A is n-perfect, we have, for $(i,j) \in \mathbb{Z}_{nu} \times \mathbb{Z}_v$, that

$$\mathbf{AC}(A)[i,j] = \begin{cases} nuv\lambda^{i/u} & \text{if } u \mid i \text{ and } j = 0 \\ 0 & \text{otherwise.}\end{cases}$$

Since B_0 is the concatenation of n copies of A side by side, an application of Lemma 7.9 yields, for $(i,j) \in \mathbb{Z}_{nu} \times \mathbb{Z}_{nv}$,

$$\mathbf{AC}(B_0)[i,j] = \begin{cases} n^2uv\lambda^{i/u} & \text{if } u \mid i \text{ and } v \mid j \\ 0 & \text{otherwise.}\end{cases}$$

Now let $k = 1, \ldots, n-1$. Then, by Corollary 6.10,

$$\mathbf{AC}(B_k) = \mathbf{AC}(B_0)^{c_{ku/v}}.$$

Hence, for $(i, j) \in \mathbb{Z}_{nu} \times \mathbb{Z}_{nv}$,

$$\begin{aligned}\mathbf{AC}(B_k)[i, j] &= \mathbf{AC}(B_0)[i + jku/v, j] \\ &= \begin{cases} n^2 uv\lambda^{(i+jku/v)/u} & \text{if } u \mid i + jku/v \text{ and } v \mid j \\ 0 & \text{otherwise.} \end{cases}\end{aligned}$$

Now $u \mid i + jku/v$ if and only if $u \mid i$, because j/v is an integer and jku/v is a multiple of u. Thus

$$\mathbf{AC}(B_k)[i, j] = \begin{cases} n^2 uv\lambda^{i/u+jk/v} & \text{if } u \mid i \text{ and } v \mid j \\ 0 & \text{otherwise.} \end{cases} \tag{7.1}$$

Note that Equation (7.1) is trivially satisfied for $k = 0$, so we will henceforth consider it for all $k \in \mathbb{Z}_n$.

We now prove that, for all $(i, j) \in \mathbb{Z}_{nu} \times \mathbb{Z}_{nv}$,

$$\sum_{k=0}^{n-1} \mathbf{AC}(B_k)[i, j] = \begin{cases} n^3 uv\lambda^{i/u} & \text{if } u \mid i \text{ and } j = 0 \\ 0 & \text{otherwise} \end{cases} \tag{7.2}$$

and also that

$$\mathbf{CC}(B_k, B_l)[i, j] = 0 \tag{7.3}$$

for all $k, l \in \mathbb{Z}_n$ with $k \neq l$.

Let $i, j \in \mathbb{Z}_n$. Then

$$\sum_{k=0}^{n-1} \mathbf{AC}(B_k)[iu, jv] = \sum_{k=0}^{n-1} n^2 uv\lambda^{i+jk} = n^2 uv\lambda^i \sum_{k=0}^{n-1} \lambda^{jk}.$$

Applying Lemma 7.1 gives

$$\sum_{k=0}^{n-1} \mathbf{AC}(B_k)[iu, jv] = \begin{cases} n^3 uv\lambda^i & \text{if } j = 0 \\ 0 & \text{otherwise.} \end{cases}$$

The above equation proves Equation (7.2) for all $(i, j) \in \mathbb{Z}_{nu} \times \mathbb{Z}_{nv}$ where $u \mid i$ and $v \mid j$; the other case follows from the fact that $\mathbf{AC}(B_k)[i, j] = 0$ for all other values of i and j with $k \in \mathbb{Z}_n$.

We now turn to Equation (7.3). Let $k, l \in \mathbb{Z}_n$, and assume that $k \neq l$. We consider the convolution $\mathbf{AC}(B_k) \circledast \mathbf{AC}(B_l)$. Now, for $(s, t) \in \mathbb{Z}_{nu} \times \mathbb{Z}_{nv}$,

$$(\mathbf{AC}(B_k) \circledast \mathbf{AC}(B_l))[s, t] = \sum_{i=0}^{nu-1} \sum_{j=0}^{nv-1} \mathbf{AC}(B_k)[i, j]\mathbf{AC}(B_l)[s - i, t - j].$$

Note, from Equation (7.1), that $(\mathbf{AC}(B_k) \circledast \mathbf{AC}(B_l))[s, t] = 0$ for all s, t where $u \nmid s$ or $v \nmid t$. Now let $s = s_0 u$ and $t = t_0 v$. Then we can write

$$\begin{aligned}
(\mathbf{AC}(B_k) \circledast \mathbf{AC}(B_l))[s, t] &= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \mathbf{AC}(B_k)[iu, jv]\mathbf{AC}(B_l)[s_0 u - iu, t_0 v - jv] \\
&= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (n^2 uv\lambda^{i+jk})(n^2 uv\lambda^{(s_0-i)+(t_0-j)l}) \\
&= n^4 u^2 v^2 \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \lambda^{(s_0+t_0 l)+j(k-l)} \\
&= n^5 u^2 v^2 \lambda^{s_0+t_0 l} \sum_{j=0}^{n-1} \lambda^{j(k-l)}
\end{aligned}$$

which is zero by Lemma 7.1. In conclusion,

$$\mathbf{AC}(B_k) \circledast \mathbf{AC}(B_l) = \mathbf{0},$$

where the $\mathbf{0}$ on the right represents the $nu \times nv$ array of zeroes. An application of Theorem 6.11 yields $\mathbf{AC}(\mathbf{CC}(B_k, B_l)) = \mathbf{0}$, which implies that $\mathbf{CC}(B_k, B_l) = \mathbf{0}$ by Lemma 3.6. This proves Equation (7.3).

Now C is the $n^2u \times nv$ array created by interleaving the rows of $B_0, \ldots, B_{n-1}$. It thus follows from Lemma 7.10 that, for $(i, j) \in \mathbb{Z}_{n^2u} \times \mathbb{Z}_{nv}$,

$$\mathbf{AC}(C)[i, j] = \sum_{t=0}^{n-1} \mathbf{CC}(B_t, B_{t+i})[[i/n] + \delta_n(t, i), j]; \tag{7.4}$$

again, recall that $B_t = B_{t \bmod n}$. If $n \nmid i$ then, for any $t = 0, \ldots, n-1$, the arrays B_t and B_{t+i} are distinct, so

$$\mathbf{CC}(B_t, B_{t+i})[[i/n] + \delta_n(t, i), j] = 0$$

in this case, giving $\mathbf{AC}(C)[i, j] = 0$.

Now suppose that $n \mid i$. Then we can write $i = i_0 n$ (so that $i_0 \in \mathbb{Z}_{nu}$ and $[i/n] = i_0$) and we have from Lemma 7.10 and Equation (7.2) that

$$\begin{aligned}
\mathbf{AC}(C)[i, j] &= \sum_{t=0}^{n-1} \mathbf{AC}(B_t)[[i/n], j] \\
&= \sum_{t=0}^{n-1} \mathbf{AC}(B_t)[i_0, j] \\
&= \begin{cases} n^3 uv \lambda^{i_0/u} & \text{if } u \mid i_0 \text{ and } j = 0 \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

where the last equality follows from Equation (7.2).

Note that $n \nmid i$ implies $nu \nmid i$, and that $i = i_0 n$ coupled with $u \mid i_0$ yields $nu \mid i$. With these facts in mind, the above two paragraphs imply that, for $(i, j) \in \mathbb{Z}_{n^2u} \times \mathbb{Z}_{nv}$,

$$\mathbf{AC}(C)[i, j] = \begin{cases} n^3 uv\lambda^{i/nu} & \text{if } nu \mid i \text{ and } j = 0 \\ 0 & \text{otherwise} \end{cases}$$

so C is n-perfect.

Finally, C was created from A using only the operations of concatenation, column shifting, and row interleaving. Since these operations alter the positioning of elements in an array but do not alter the elements themselves, it follows that C is of the same alphabet as A. □

7.6 Discussion

The elements of an n-perfect array do not have to be nth roots of unity; they can be anything which gives nth roots of unity in the autocorrelation array. All examples of n-perfect arrays given in this chapter have been over nth roots of unity. However, Arasu and de Launey's almost perfect arrays are in fact 2-perfect, by our definitions, and in [1], examples of almost perfect arrays over the quaternary (4th roots) alphabet are given.

Example 7.8 in Section 7.4 gives an example of a 4-perfect array over the quaternary alphabet. This array (and all larger ones generated recursively) could be used in applications, and it is not obvious how to construct these arrays using the method in [1].

Chapter 8

Conclusion

In this research we studied the algebraic nature of array manipulations such as correlation and convolution, and we also studied the existence of arrays whose autocorrelation is as close to perfect as possible (that is, having as few nonzero off-peak autocorrelation values as possible). We were able to devise a succinct algebraic method of computing correlation of arrays and sequences, and we used this method to prove a powerful recursive construction of almost perfect arrays. We were also able to generalise that construction, proving the existence of infinitely large "n-perfect" arrays. Finally, and in conjunction with our colleague Barerra Acevedo, we were able to modify an algorithm of Arasu and de Launey, thereby proving the existence of infinitely large perfect arrays over the basic quaternions.

8.1 Summary of Contributions

In this research we

(1) Translated and proved the Arasu and de Launey inflation of perfect quaternary arrays construction from the language of polynomials into the language of arrays, thereby generalising the alphabet to any subset of $\mathbb{C}$ and

also making the algorithm more accessible.

(2) Modified the algorithm of (1), enabling it to inflate perfect arrays over basic quaternion alphabet, and also proving the existence of infinitely large perfect arrays over the basic quaternions.

(3) Created a new algebra of arrays which can succinctly state and prove array constructions which involve correlation and convolution.

(4) Used the algebra of arrays in (3) to restate and prove a recursive construction of Arasu and de Launey yielding arbitrarily large almost perfect arrays. Our construction also makes it clear that the alphabet can be over any subset of $\mathbb{C}$, rather than the quaternary alphabet used by Arasu and de Launey.

(5) Created a new class of arrays ("n-perfect"), which have been based on the almost perfect arrays of Arasu and de Launey.

(6) Generalised ideas of Arasu and de Launey to state and prove a new recursive construction of n-perfect arrays over any alphabet, thereby yielding arbitrarily large n-perfect arrays.

Bibliography

[1] K. T. Arasu and Warwick de Launey, "Two-Dimensional Perfect Quaternary Arrays", *IEEE Transactions on Information Theory*, Vol. 47, No. 4, May 2001, pp. 1482–1493.

[2] Santiago Barerra Acevedo, "Perfect Sequences and Arrays of Unbounded Lengths and Sizes over the Basic Quaternions", PhD Thesis, Monash University, 2013.

[3] Santiago Barerra Acevedo and T. E. Hall, "Perfect Sequences of Unbounded Lengths over the Basic Quaternions", *Lecture Notes in Computer Science*, SETA2012, 2012, pp. 159–167.

[4] Santiago Barerra Acevedo and Nathan Jolly, "Perfect Arrays of Unbounded Sizes over the Basic Quaternions", *Cryptography and Communications*, June 2013, DOI 10.1007/s12095-013-0086-x.

[5] L.D. Baumert, *Difference Sets, Lecture Notes in Mathematics*, ed, Vol. 182, Springer-Verlag Press, 1971.

[6] D. Calabro and J. K. Wolf, "On The Synthesis of Two-Dimensional Arrays With Desirable Correlation Properties", *Information and Control*, Vol. 11, 1968, pp. 537–560.

[7] R. A. Games, "Crosscorrelation of M-sequences and GMW-sequences With The Same Primitive Polynomial", *Discrete Applied Mathematics*, Vol. 12, 1985, pp. 139–146.

[8] Guang Gong, "Theory and Applications of q-ary Interleaved Sequences", *IEEE Transactions on Information Theory*, Vol. 41, No. 2, March 1995, pp. 400–411.

[9] D. Everett, "Periodic Digital Sequences with Pseudonoise Properties", *G.E.C. Journal*, Vol. 33, No. 3, 1966, pp. 115–126.

[10] T. E. Hall, C. F. Osborne and A. Z. Tirkel, "Families of Matrices with Good Auto and Cross-Correlation", *Ars Combinatoria*, Vol. 61, 2001, pp. 187–196.

[11] F. Hartung and M. Kutter, "Multimedia Watermarking Techniques", *Proc. IEEE*, 1999, **87**, (7), pp. 1079–1107.

[12] R. C. Heimiller, "Phase Shift Pulse Codes With Good Peeriodic Correlation Properties", *IRE Transactions on Information Theory IT -7*, 1961, pp. 254–257.

[13] Nathan Jolly, "An Algebra of Arrays and Almost Perfect Watermarks", *Cryptography and Communications*, Vol. 7, No. 2, 2015, DOI 10.1007/s12095-015-0123-z.

[14] O. Kuznetsov, "Perfect Sequences over the Real Quaternions", *Signal Design and Its Applications in Communications*, IWSDA '09, Fourth International Workshop, Vol. 1, 2010, pp. 17–20.

[15] C. E. Lee, "Perfect q-ary Sequences From Multiplicative Characters Over GF(p)", *Electronic Letters*, Vol. 28, 1992, pp. 833–835.

[16] J. Lindner, "Binary Sequences Up To Length 40 With Best Possible Autocorrelation Function", *Electron. Lett.*, Vol. 11, p. 507, 1975.

[17] F. J. MacWilliams and N. J. Sloane, "Pseudo-Random Sequences and Arrays", *Proceedings of the IEEE*, Vol. 64, No. 12, pp. 1715–1729, 1976.

[18] Paul J. McCarthy, Introduction to Arithmetical Functions, Springer-Verlag, 1986.

[19] O. Moreno and S. V. Maric, "A New Family of Frequency Hop Codes", *IEEE Trans. Commun.*, Vol. 48, No. 8, pp. 1241–1244, Aug. 2000.

[20] H. J. Nussbaumer, "Fast Fourier Transform and Convolution Algorithms", Springer-Verlag, 1981.

[21] M. R. Schroeder, "Number Theory in Science and Communication", 3rd ed., Springer-Verlag, 1997.

[22] R. A. Scholtz, P. V. Kumar and C. J. Corrada-Bravo, "Signal Design for Ultra-Wideband Radio, Sequences and Their Applications", SETA '01, May, 2001.

[23] E. S. Shivaleela, K. N. Sivarajan and A. Selvarajan, "Design of a New Family of Two-Dimensional Codes for Fiber-Optic CDMA Networks", *J. Lightwave Technol.*, Vol. 16, No. 4, pp. 501–508, 1998.

[24] A. Z. Tirkel, G. A. Rankin, R. M. van Schyndel, W. J. Ho, N. R. A. Mee and C. F. Osborne, "Electronic Water Mark", DICTA 1993, pp. 666–673, Maquarie University, Sydney, 1993.

[25] A. Tirkel and T. E. Hall, "Matrix Construction Using Cyclic Shifts of a Column", *International Symposium on Information Theory*, 2005.

[26] R. Turyn, “Sequences With Small Correlation”, in *Error Correcting Codes*, H. B. Mann, Ed. New York: Wiley, 1969, pp. 195–228.

[27] L. J. Weng, “Decomposition of M-sequences and Its Applications”, *IEEE Transactions on Information Theory*, Vol. 17, 1971, pp. 457–463.