1
Introduction

The goal of this book is to provide algebraic tools for the design of pseudo-random sequences. It is meant to be both a text book and a reference book. We present a unified approach based on algebraic methods, which allows us to simultaneously treat linear feedback shift registers, feedback with carry shift registers, and many other analogous classes of sequence generators. The requisite algebraic tools are developed in Appendices A through D.

1.1 Pseudo-random sequences

Applications of random numbers became so widespread in the early 1950s that eventually a table of one million random digits was generated and published [178] by the Rand corporation. It was soon found necessary to generate "random numbers" in real time using a computer algorithm. Sequences of numbers generated in this way are referred to as pseudo-random, see Section 8.1. Pseudo-random sequences have become ubiquitous in modern electronics and information technology. They are used, for example, as spreading codes in communications systems (such as cellular telephones and GPS signals), as components for generating keystreams for stream ciphers and other cryptographic applications, as sampling data for simulations and Monte Carlo integration, for timing measurements in radar and sonar signals and in GPS systems, as error correcting codes in satellite and other communications, as randomizers of digital signals to eliminate spectral lines, as counters in field programmable gate arrays, and in power on self tests.

In all cases speed in generating the sequences is important — time spent generating sequences affects the speed of the whole system, and thus affects throughput, or accuracy, or some other important characteristic. In most cases it is also important to be able to reproduce the sequence (for example, so a receiver can undo the encoding from a transmitter). Many desirable characteristics of sequences are specific to the application. In cryptography we want sequences that are difficult to predict from short segments. In CDMA we want families of sequences with low pairwise correlations. Error correcting codes require families of sequences with large pairwise Hamming distance and efficient decoding algorithms. Many applications require sequences with uniform distributions of fixed size patterns. Some, such as Monte Carlo methods, require many other statistical properties.

1.2 LFSR sequences

For all of these applications there are solutions that rely on sequences produced by *linear feedback shift registers* (LFSRs). They are easily implemented in hardware, resulting in high speed generation of the sequences. There is a rich algebraic theory of the design and analysis of LFSR sequences, which is used in the selection of those generators having the most appropriate properties for a given application. Some applications rely on other types of sequence generators such as *feedback with carry shift registers* (FCSRs). These can also be implemented in hardware for high speed, and they have an algebraic theory parallel to that of LFSRs.

Although they were preceded by many important papers and technical reports [47, 48, 60, 93, 136, 190, 210, 211], the publication of “Error correcting codes” [172] by W. W. Peterson and “Shift Register Sequences” [61] by S. Golomb were milestones in the development of LFSR techniques for the generation of pseudo-random sequences. These books explained and exploited the deep connection between the architecture of the shift register and the mathematics of Galois theory in a way that makes for exciting reading, even today, 45 years later. In Golomb’s book, each “cell” of the shift register is a vacuum tube that can be either ON or OFF, and the output of the shift register is a pseudo-random sequence of zeroes and ones. Reading this book, one is tempted to run out and buy the parts to build one of these machines and watch it run.

One of the most fascinating aspects of this theory concerns the design of shift registers that produce maximal length sequences, or *m-sequences*, and the remarkable statistical and correlation properties of these sequences, which we describe in Chapter 10: besides having maximal length, each m-sequence $\mathbf{a}$ of rank k is also a (punctured) de Bruijn sequence[1] and its autocorrelation function is optimal. It is an amazing fact that the design problem can be completely solved using the Galois theory of finite fields. Although this fact was known already to L. E. Dickson [41] (in a slightly different language, of course, since electronic shift registers did not exist in 1919), it was rediscovered in the 1950s by the engineering community, and it remains one of the most compelling illustrations of how an abstract mathematical theory can unexpectedly become the key to understanding a complex physical system. Similarly, the distribution and correlation properties of m-sequences turn

[1] Each subsequence of length k, except the all-zero subsequence, occurs exactly once in each period of $\mathbf{a}$.

out to be related to a variety of mathematical abstractions including finite fields, difference sets (a subject that developed independently, but around the same time [80]), and even elliptic curves. See Section 10.6.4.

The explosive development of code division multiple access (CDMA) communications, especially with cellular telephones, has created considerable interest in finding families of pseudo-random sequences with low cross-correlation between distinct sequences in a given family. In Chapter 11 we discuss some of the more common families, including Gold codes, Kasami sequences, GMW sequences, and Legendre sequences, as well as several more exotic variations on these themes. Our analysis is more "geometric" than the standard approach. We have only scratched the surface of this fascinating topic and the reader interested in a more complete study of correlation questions may wish to consult Golomb and Gong's recent book [63].

In the years since the publication of Golomb's book [61], the basic design of shift registers has been enhanced in several different directions. The "Galois" and "Fibonacci" modes were developed [173], many ways of interconnecting shift registers were analyzed, and perhaps most significantly, the binary state vacuum tubes were eventually replaced by cells with many possible states. Engineers were led, for example, to consider N-ary shift registers, whose cell contents are taken from the integers modulo N, or from a finite Galois field, or more generally from an algebraic ring. It turns out that much of the analysis of shift register sequences goes through in this more general setting. In Chapters 3, 10, and 11 we present this general analysis. Although the material is not new, it is derived from many disparate sources.

1.3 FCSR sequences

It is possible to enhance the basic shift register architecture in yet another way, by the addition of a small amount of *memory*. The memory is used as a "carry" in the calculations. For example, when two sequences of ones and zeroes are added, they can be added as elements of $\mathbb{Z}/(2)$ (or XOR addition) in which $1 + 1 = 0$, or they can be added as "integers", in which 1+1 = 2 = 0 + a carry of 1. The difference is illustrated by the following example, where carries go to the right (which is the opposite convention to the usual place value notation):

$$\begin{array}{cccccccccc}
1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
\hline
0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\
\multicolumn{10}{c}{\text{mod 2}}
\end{array}
\qquad
\begin{array}{cccccccccc}
1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
\hline
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
\multicolumn{10}{c}{\text{with carry.}}
\end{array}$$

In fact, the *summation combiner* [182] does exactly the latter: it combines two binary sequences into a third, using addition with carry. It was originally proposed as a method for creating a difficult-to-predict bit stream from two relatively easy-to-predict bit streams, for cryptographic applications.

In an effort to analyze the summation combiner, the authors decided to incorporate this addition with carry into the architecture of a linear feedback shift register. The result was the *feedback with carry* shift register (FCSR) [114, 115, 119], described in Chapters 4, 13, and 16.

Around the same time a similar idea began circulating in the random number generation community, perhaps initiated by G. Marsaglia [144] and A. Zaman [146], and more fully developed by R. Couture and P. l'Écuyer [31, 32] and others, where the method became known as "add with carry" and "multiply with carry" (MWC) random number generators. These approaches appeared to be different at first because the add with carry generators involved only two "cells", each of which stores a very large integer, while the FCSR used many cells, each of which stores only a single bit. But in later papers, architectures were considered which involve (possibly) many cells, each storing (possibly) large integers, and in this setting the two methods are seen to be identical. The generation and analysis of FCSR and MWC sequences is covered in Chapters 4 and 13. Galois and Fibonacci versions of FCSR generators also exist. See Chapter 7.

As in the case of LFSRs, the FCSR architecture can be enhanced by considering cell contents taken from $\mathbb{Z}/(p)$, or a finite field, or even an arbitrary ring. But in these cases, the analysis becomes considerably more difficult (and interesting). The natural setting for all these architectures is the *algebraic feedback shift register* or AFSR, for which the general theory is developed in Chapter 5. In this generality, the theory of AFSRs includes both that of FCSRs (with cell contents in an arbitrary ring) and LFSRs. But the AFSR architecture also contains a number of new and interesting special cases as well, some of which are studied in some detail in Chapters 5, 6, 12, 14, and 17.

For example, another way in which the FCSR architecture can be enhanced is to delay the "carry" by a certain number of steps. Here is an example of addition in which the carry is delayed by one step:

$$\begin{array}{cccccccccc}
1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
\hline
0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\
\multicolumn{10}{c}{\text{carry delay } +1}
\end{array} \tag{1.1}$$

It is possible to build hardware FCSR generators that implement this sort of addition; we call them d-FCSRs[2] and they are described in Chapter 6. The d-FCSR is a special case of an AFSR, and although the analysis of the d-FCSR is more

[2] Our original intention was that the d in d-FCSR was an integer indicating the degree, but it has been since claimed that d stands for "delay".

difficult than that of the FCSR, it is somewhat simpler than the general AFSR and is still surprisingly complete.

For another example, the cells of an FCSR, or of a d-FCSR, could be polynomials. This gives rise to the function field FCSR, as described in Section 5.4.2 and Chapter 12. There appears to be an almost unlimited number of variations on this theme.

As in the case of LFSR sequences, the maximal length FCSR sequences (which we refer to as ℓ-sequences) have several remarkable properties. Such a sequence is as equi-distributed as possible, given its period. Although the autocorrelation function of such a sequence is not known in general, it nevertheless has perfect *arithmetic autocorrelation* (a function that is the direct arithmetic analog of the usual autocorrelation function). These properties are investigated in Chapter 13. In Chapter 14 the analogous questions for maximal length d-FCSR sequences are considered. The case of AFSR sequences based on a function field is especially surprising: the ones of maximal length turn out to be punctured de Bruijn sequences with ideal auto-correlations, but (in the non-binary case) they are not necessarily m-sequences. See Chapter 12.

1.4 Register synthesis

In Part III of the book we change gears and consider the “synthesis” problem: given a periodic sequence **a**, how can we construct a device (such as an LFSR) that will generate the sequence? The length of the smallest such LFSR is called the *linear span* or *linear complexity* of the sequence **a**. The optimal result in this direction is the *Berlekamp–Massey algorithm* which predicts later terms of the sequence using the minimum possible amount of data, and it does so very efficiently. It is well known that this algorithm is “essentially” the same as the continued fraction expansion in the field of formal power series. In Section 15.2.4 the exact relation between these two procedures is made explicit.

The Berlekamp–Massey algorithm therefore provides a possible technique for uncovering the keystream in a stream cipher, based only on the knowledge of the plaintext. It is therefore desirable that such a keystream should have an enormous linear span. Several standard techniques for constructing sequences with large linear span are described in Section 15.5, but the reader should be aware that this is a rapidly changing field and none of these techniques is considered “secure” by modern standards.

The continued fractions approach to FCSR synthesis (that is, the construction of an FCSR that produces a given sequence) simply does not work. However, two successful approaches are described in Section 16.3. The first is based on the mathematical theory of *lattice approximations*, and the second is based on the extended

Euclidean algorithm. These techniques were used to "break" the summation cipher described above. The problem of AFSR synthesis is even more difficult, but in Chapter 17 we describe Xu's algorithm which converges in many cases.

The mathematics behind all this material can be a daunting obstacle. Although we have included the appropriate mathematical background in the appendices of this book, most of the chapters in the main part of the book have been written so as to be independent of this material, as much as possible. The reader is invited to start in the middle, on a topic of interest, and to read as far as possible until it becomes necessary to refer to the appendices, using the index as a guide. It may even be possible to delay the retreat to the appendices indefinitely, by skipping the proofs of the theorems and propositions.

1.5 Applications of pseudo-random sequences

Although pseudo-random sequences are ubiquitous in modern technology, each application requires a particular type of pseudo-random sequence which must be optimized so as to have particular statistical properties. For this reason, the subject of pseudo-random sequences is enormous and varied. In the following paragraphs we only give an outline of some of the most common applications.

1.5.1 Frequency hopping spread spectrum

In a frequency hopping (FH) wireless communication system, the transmitter and receiver jump, in synchrony, from one frequency to another, as directed by a (periodic) pseudo-random sequence $\mathbf{a} = a_0, a_1, \cdots$ that they share. The power spectrum of the transmitted signal is therefore "spread" out among these various frequencies.

This technique may be used to minimize interference from natural or man-made causes: an agent that attempts to jam the transmission is forced to pursue the costly strategy of jamming all the frequencies unless he also knows the pseudo-random sequence. Sequences designed for this application must therefore be difficult to predict based on partial knowledge of the sequence. FH can also be used as part of an encryption scheme: an unwanted listener must monitor all the frequencies in order to detect the full message. FH is used in multiple-access systems in which several communications occur simultaneously. In this case, each transmitter is assigned its own (periodic) *signature* pseudo-random sequence of frequencies. These sequences must be designed so as to exhibit low *Hamming correlation*. That is, the design should minimize the chances of a *collision* (when two or more transmitters attempt to transmit on the same frequency at the same time). See Section 11.9. Recovery from occasional collisions can be effected through the use of an error correcting code.

The first patent for an FH system was obtained by actress Hedy Lamarr (Hedy Marky) and composer George Antheil in 1944. See Section 11.9 and references [193, 200]. FH is widely implemented and is currently part of the Bluetooth specification. Some brands of 2.4 GHz transmitters for radio control make use of FH.

1.5.2 Code division multiple access

In a *code division multiple access* (CDMA) communication system, many users wish to communicate on the same channel, for example on a single radio frequency or on a single optical fiber. Each transmitter (A, B, C, etc.) is pre-assigned a (relatively short) signature sequence (s_A, s_B, etc.). Often it is a binary sequence. The signature sequence s_A must also be known to the receiver if it is attempting to receive a message from A. Let us say that transmitter A wishes to send a message. To send the information bit “1”, transmitter A broadcasts its full signature sequence. To send the information bit “0” it broadcasts the complementary sequence. Typically many transmitters operate simultaneously on the same base frequency. If the family of sequences is chosen so as to have low *cross-correlation* then the receiver is able to determine the information bits from transmitter A by correlating the total received signal with the signature sequence s_A. Since the signals from the other transmitters have low correlation with this signature sequence, these other signals will appear as noise at the output of the receiver’s correlator. Provided the different signature sequences are *orthogonal* (meaning that their cross-correlations are small), and provided there are not too many simultaneous users, this noise will not swamp the positive correlations from A. The signature sequence for transmitter A is also chosen so as to have low out-of-phase correlation with itself, so that the correlator will only register a high (positive or negative) correlation when it is synchronized with the start of the transmitted signature sequence.

Usually, each transmitter is designed so as to broadcast a sine wave at a single base frequency F_0. The signature sequence is typically transmitted using *phase shift keying*: to alternate between a “0” and a “1” the phase of the transmission might be shifted by 180 degrees. The rate at which these shifts occur is called the *chip rate*. Dividing the chip rate by the length of the signature sequence gives the rate at which the information bits are processed, or the *information rate*. The power spectrum of the transmission appears as a spike at the frequency F_0 with sidelobe spikes separated from F_0 by integer multiples of the chip rate and the information rate. Consequently, the spectrum is “spread” over a frequency band, and, just as with an FH system, the transmission is somewhat immune to jamming by natural or man-made interference.

CDMA systems therefore require families of pseudo-random sequences with low auto- and cross-correlation. Further desirable properties may include cryptographic strength of the signature sequences, error correcting capabilities, etc. These desiderata tend to be mutually incompatible and one of the challenges of modern communication theory is the design of optimal CDMA families.

CDMA is widely used in cellular telephone and GPS systems. See the exercises in Section 11.12 where the C/A (coarse acquisition) code for the current USA civilian GPS system is described.

1.5.3 Optical CDMA

In an optical fiber (or similar channel) there is only "room" for a single signal at a time (meaning that the fiber is either active with light, or it is dark). Such a channel can be used for CDMA as above, with each user assigned his own signature sequence. In order to send a binary "1" the user sends his signature sequence. To send a binary "0" the user sends nothing. When r users are transmitting simultaneously, their signals $a_1, a_2, \cdots, a_r$ are combined according to the "maximum" rule, $a = \max(a_1, a_2, \cdots, a_r) \in \{0, 1\}$. Such a system will work provided each individual signature sequence has many zeroes and few ones, thereby leaving as much "dark" time as possible for the other users.

As in the case of wireless CDMA, the individual messages are recovered using a correlation receiver. However the definition of "correlation" is slightly different for the optical channel than for the wireless channel. See Section 11.10.

1.5.4 Synchronization and radar

In many applications (such as radar) it is necessary to measure the time lag between two signals (for example, a transmitted and a reflected signal). For example one might transmit a single, high powered, short duration burst and measure the elapsed time for the reflected signal. But there are many problems with this approach. Such bursts are difficult to produce, and due to propagation and other effects, the received signal may fail to look like a "burst": it may be distributed over a relatively large time segment.

Another approach is to transmit a periodic pseudo-random sequence with optimal autocorrelation (meaning that the out of phase correlation of the sequence with itself is zero, or almost zero). In the simplest systems, a correlator compares the received signal to all possible (cyclic) shifts of the transmitted signal. The output of the correlator will exhibit a sharp peak for exactly one shift which therefore translates into a time lag (which will be a multiple M_0 of the chip period T_0). However this measurement carries an indeterminacy that is an integral multiple M_1 of the period T_1 of the sequence, meaning that the sequence went through M_1 full periods before returning. The time lag M_1T_1 is easier to measure because the period T_1 is typically much greater than T_0.

This technique was used by the Jet Propulsion Laboratory in 1961 to bounce a radar signal off the surface of Venus. This exciting story, and its consequences for the accurate determination of the astronomical unit, are described in [22]. Modern radar systems are much more complex than described here, since we have not even addressed the issue of Doppler shift due to the motion of the target, and pseudo-random sequence design for radar is an enormous but fascinating subject in its own right.

1.5.5 Stream ciphers

In a stream cipher, a *keystream generator* produces a pseudo-random sequence of bits (or of elements in an Abelian group). This sequence is added, symbol by symbol, to the plaintext. The result is called the *ciphertext* which is then broadcast. The receiver, who has an identical copy of the keystream generator, subtracts its output, symbol by symbol, from the ciphertext, and thereby recovers the plaintext.

Stream ciphers are used for transmitting large amounts of data. They are extremely fast and are often implemented in hardware for added speed. Consequently the keystream generator must be designed to produce a pseudo-random sequence of enormous period using a relatively simple algorithm. In a *known plaintext attack* it is assumed that the enemy (by some nefarious technique or by negligence on the part of the user) has knowledge of a relatively small section of the plaintext. This allows the enemy to recover that section of the keystream. A stream cipher will therefore be secure against a plaintext attack provided it is difficult to predict the key sequence from a knowledge of a (relatively small) segment of the sequence. This is the main requirement for a “strong” key sequence. In some situations one even assumes that the attacker has knowledge of the hardware that is used to generate the key sequence, but does not have knowledge of its initial state.

Many techniques exist for predicting the full key sequence, based on a knowledge of a segment, some of which are covered in Section 15.5 of this book. One technique is a *correlation attack* in which one attempts to correlate the known segment with the state of the keystream generator. Other techniques include statistical tests and the Berlekamp–Massey algorithm. See Section 15.2.

These requirements (simple algorithm, large period, difficult to predict from short segments) are incompatible in the complexity-theoretic sense, but not in the practical sense, and a tremendous amount of research has been invested in the design and analysis of cryptographically secure keystream generators. Stream ciphers are in wide use, perhaps the most common being RC4, which is used in WEP (for secure WiFi communication) and SSL (for secure internet transactions).

1.5.6 Pseudo-random arrays

Imagine a huge $m \times n$ (bi-periodic) array of zeroes and ones such that every non-zero nine by nine block appears exactly once in a single period. There are $2^{81} - 1$ possible blocks, so the array might have $m = 2^{27} - 1$ and n $= (2^{81} - 1)/(2^{27} - 1)$, which is approximately 2^{54}. Now imagine that an agent is lost in this array, but from its location it can see the square it resides on as well as the 80 neighboring squares that are within four steps (in any direction) from its present location. The agent looks up in a table (or preferably, does a small computation) and it is then able to determine that it is at location (x, y), which is uniquely determined by the nine by nine pattern of bits.

Such arrays are used on the floors of warehouses to help robots determine their position. Another example is a widely marketed "digital pen" that writes on "digital paper" that is printed with a background array of microscopic dots. The pen has a tiny camera that reads a small region on the paper, from which software determines the position of the pen on the paper, as well as which piece of paper is being used, and which pad it came from. The background array of dots is an example of a pseudo-random array which has been designed so that any nine by nine region will uniquely determine the position within the array. At 300 dots per inch an array of 2^{27} by 2^{54} dots is approximately 8 miles by 1 000 000 miles, a surface area large enough that every piece of digital paper printed by the company can come from a different region. Pseudo-random arrays are discussed in Section 10.4.

1.5.7 Monte Carlo

Let $\overline{I}^s = [0, 1]^s$ and suppose $f : \overline{I}^s \to \mathbb{R}$ is a continuous function. We wish to calculate $\int_{\overline{I}^s} f(u)du$. If $s = 1$ the integral can be approximated using the trapezoidal rule or Riemann sums: let $0 = x_1, \cdots, x_N - 1$ be N equally spaced points in [0, 1] and form

$$\frac{1}{N}\sum_{i=1}^{n} f(x_i). \tag{1.2}$$

If s is large this becomes impractical because it will involve N^s evaluations. The Monte Carlo method consists of choosing points $x_1, \cdots, x_N \in \overline{I}^s$ at random (independent and uniformly distributed) for use in equation (1.2). It is known that the *expected* error in this estimate varies as $O\left(1/\sqrt{N}\right)$. Typically these computations are performed for large values of N, so an efficient source of independent