4-1 Coding for Wireless Channels

Coding allows bit errors introduced by transmission of a modulated signal through a wireless channel to be either detected or corrected by a decoder in the receiver. Coding can be considered as the embedding of signal constellation points in a higher-dimensional signaling space than is needed for communications. By going to a higher-dimensional space, the distance between points can be increased, which provides for better error correction and detection.

In this chapter we describe codes designed for additive white Gaussian noise channels and for fading channels. Codes designed for AWGN channels typically do not work well on fading channels because they cannot correct for long error bursts that occur in deep fading. Codes for fading channels are mainly based on using an AWGN channel code combined with interleaving, but the criterion for the code design changes to provide fading diversity. Other coding techniques to combat performance degradation due to fading include unequal error protection codes and joint source and channel coding.

We first provide an overview of code design in both fading and AWGN, along with basic design parameters such as minimum distance, coding gain, bandwidth expansion, and diversity order. Sections 8.2 and 8.3 provide a basic overview of block and convolutional code designs for AWGN channels. Although these designs are not directly applicable to fading channels, codes for fading channels and other codes used in wireless systems (e.g., spreading codes in CDMA) require background in these fundamental techniques. Concatenated codes and their evolution to turbo codes - as well as low-density parity-check (LDPC) codes - for AWGN channels are also described. These extremely powerful codes exhibit near-capacity performance with reasonable complexity levels. Coded modulation was invented in the late 1970s as a technique to obtain error correction through a joint design of the modulation and coding. We will discuss the basic design principles behind trellis and more general lattice coded modulation along with their performance in AWGN.

Code designs for fading channels are covered in Section 8.8. These designs combine block or convolutional codes designed for AWGN channels with interleaving and then modify the AWGN code design metric to incorpo- rate maximum fading diversity. Diversity gains can also be obtained by combining coded modulation with symbol or bit interleaving, although bit interleaving generally provides much higher diversity gain. Thus, coding combined with interleaving provides diversity gain in the same manner as other forms of diversity, with the diversity order built into the code design. Unequal error protection is an alternative to diversity in fading mitigation. In these codes bits are prioritized, and high-priority bits are encoded with stronger error protection against deep fades. Since bit priorities are part of the source code design, unequal error protection is a special case of joint source and channel coding, which we also describe.

Coding is a very broad and deep subject, with many excellent books devoted solely to this topic. This chapter assumes no background in coding, and thus it provides an in-depth discussion of code designs for AWGN channels

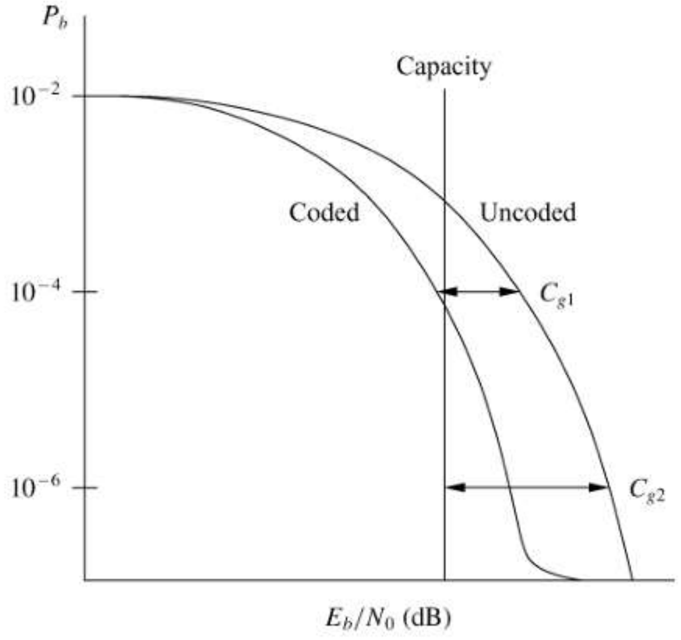

Figure 4.1: Coding gain in AWGN channels.

before designs for wireless systems can be treated. This in-depth discussion can be omitted for a more cursory treatment of coding for wireless channels.

4.2 Overview of Code Design

The main reason to apply error correction coding in a wireless system is to reduce the probability of bit or block error. The bit error probability P, for a coded system is the probability that a bit is decoded in error. The block error probability P, also called the packet error rate, is the probability that one or more bits in a block of coded bits is decoded in error. Block error probability is useful for packet data systems where bits are encoded and transmitted in blocks. The amount of error reduction provided by a given code is typically characterized by its coding gain in AWGN and its diversity gain in fading.

Coding gain in AWGN is defined as the amount that the bit energy or signal-to-noise power ratio can be reduced under the coding technique for a given P_b, or P_{bl}. We illustrate coding gain for P_b, in Figure 4.1. We see in this figure that the gain C_{g1} at $P_b = 10^{-4}$ is less than the gain C_{g2} at $P_b = 10^{-6}$, and there is negligible coding gain at P_b, = 10^{-2}. In fact, codes designed for high-SNR channels can have negative coding gain at low SNRs, since the extra redundancy of the code does not provide sufficient performance gain in P_b, or P_{bl} at low SNRs to compensate for spreading the bit energy over multiple coded bits. Thus, unexpected fluctuations in channel SNR can significantly degrade code performance. The coding gain in AWGN is generally a function of the minimum Euclidean distance of the code, which equals the minimum distance in signal space between code words or error events. Thus, codes designed for AWGN channels maximize their Euclidean distance for good performance.

Error probability with or without coding tends to fall off with SNR as a waterfall shape at low to moderate SNRs. Whereas this waterfall shape holds at all SNRs for uncoded systems and many coded systems, some codes (such as turbo codes) exhibit error floors as SNR grows. The error floor, also shown in Figure 4.1, kicks in at a threshold SNR that depends on the code design. For SNRs above this threshold, the slope of the error probability curve decreases because minimum distance error events dominate code performance in this SNR regime.

Code performance is also commonly measured against channel capacity. The capacity curve is associated with the SNR (E_b/N_0) where Shannon capacity Blog (1+ SNR) equals the data rate of the system. At rates up to capacity, the capacity-achieving code has a probability of error that goes to zero, as indicated by the straight line in Figure 4.1. The capacity curve thus indicates the best possible performance that any practical code can achieve.

For many codes, the error correction capability of a code does not come for free. This performance enhance- ment is paid for by increased complexity and - for block codes, convolutional codes, turbo codes, and LDPC codes -by either a decrease in data rate or an increase in signal bandwidth. Consider a code with n coded bits for every k uncoded bits. This code effectively embeds a k-dimensional subspace into a larger n-dimensional space to provide larger distances between coded symbols. However, if the data rate through the channel is fixed at Rs, then the information rate for a code that uses n coded bits for every k uncoded bits is (k/n)R; that is, coding decreases the data rate by the fraction k/n. We can keep the information rate constant and introduce coding gain by decreasing the bit time by k/n. This typically results in an expanded bandwidth of the transmitted signal by n/k. Coded modulation uses ajoint design of the code and modulation to obtain coding gain without this bandwidth expansion,

Codes designed for AWGN channels do not generally work well in fading owing to bursts of errors that cannot be corrected for. However, good performance in fading can be obtained by combining AWGN channel codes with interleaving and by designing the code to optimize its inherent diversity. The interleave spreads out bursts of errors over time, so it provides a form of time diversity. This diversity is exploited by the inherent diversity in the code. In fact, codes designed in this manner exhibit performance similar to MRC diversity, with diversity order equal to the minimum Hamming distance of the code. Hamming distance is the number of coded symbols that differ between different code words or error events. Thus, coding and interleaving designed for fading channels maximize their Hamming distance for good performance.

4.3 Linear Block Codes

Linear block codes are conceptually simple codes that are basically an extension of single-bit parity-check codes for error detection. A single-bit parity-check code is one of the most common forms of detecting transmission errors. This code uses one extra bit in a block of n data bits to indicate whether the number of l-bits in a block is odd or even. Thus, if a single error occurs, either the parity bit is corrupted or the number of detected 1-bits in the information bit sequence will be different from the number used to compute the parity bit; in either case the parity bit will not correspond to the number of detected 1-bits in the information bit sequence, so the single error is detected. Linear block codes extend this notion by using a larger number of parity bits to either detect more than one error or correct for one or more errors. Unfortunately, linear block codes - along with convolutional codes - trade their error detection or correction capability for either bandwidth expansion or a lower data rate, as we shall discuss in more detail. We will restrict our attention to binary codes, where both the original information and the corresponding code consist of bits taking a value of either 0 or 1.

4.4 Binary Linear Block Codes

A binary block code generates a block of n coded bits from k information bits. We call this an (n, k) binary block code. The coded bits are also called code word symbols. The n code word symbols can take on 2^n possible values corresponding to all possible combinations of the n binary bits. We select 2^k code words from these 2^n possibilities to form the code, where each & bit information block is uniquely mapped to one of these 2^k code words. The rate of the code is R_c = k/n information bits per code word symbol. If we assume that code word symbols are transmitted across the channel at a rate of R, symbols per second, then the information rate associated with an (n, k) block code is $R_b = R_cR_s = (k/n)\ R_s$, bits per second. Thus we see that block coding reduces the data rate compared to what we obtain with uncooked modulation by the code rate R_c

A block code is called a linear code when the mapping of the k information bits to the n code word symbols is a linear mapping. In order to describe this mapping and the corresponding encoding and decoding functions in more detail, we must first discuss properties of the vector space of binary n-tuples and its corresponding subspaces.

The set of all binary n-tuples B_n is a vector space over the binary field, which consists of the two elements 0 and 1. All fields have two operations, addition and multiplication: for the binary field these operations correspond to binary addition (modulo 2 addition) and standard multiplication. A subset S of B_n is called a subspace if it satisfies the following conditions.

1. The all-zero vector is in S.

2. The set S is closed under addition; that is, if $S_i \in S$ and $S_j \in S$ then $S_i + S_j \in S$.

An (n, k) block code is linear if the 2^K length-n code words of the code form a subspace of B_n. Thus, if C_i and C_j, are two code words in an (n, k) linear block code, then $C_i + C_j$ must form another code word of the code.

Example 4.1:

The vector space B3 consists of all binary tuples of length 3:

$$B_3 = \{[000], [001], [010], [011], [100], [101], [110], [111]\}.$$

Note that B3 is a subspace of itself, since it contains the all-zero vector and is closed under addition. Determine which of the following subsets of B3 form a subspace:

- $A_1 = \{[000], [001], [100], [101]\};$
- $A_2 = \{[000], [100], [110], [111]\};$
- $A_3 = \{[001], [100], [101]\}.$

Solution: It is easily verified that A1 is a subspace, since it contains the all-zero vector and the sum of any two tuples in A_1 is also in A_1. A_2 is not a subspace because it is not closed under addition, as $110 + 111 = 001 \in A_2$. A_3 is not a subspace because it is not closed under addition ($001+001 = 000 \in A_3$) and it does not contain the all-zero vector.