EE 387 course information

- Instructor: John Gill, Packard 266
- Textbook: *Algebraic Codes for Data Transmission* by Richard Blahut
- Weekly homework, including occasional programming problems
- Midterm exam, in class, Monday, October 31, 9:00–10:20am
- Final exam, in class, Tuesday, December 8, 8:30–11:30am
- Class web page: http://www.stanford.edu/class/ee387
Textbook coverage

Chapters 1 through 8 and 12 of *Algebraic Codes for Data Transmission*:

1. Introduction to error control codes
2. Introduction to algebra: groups, rings, vector spaces, finite fields
3. Linear block codes: matrices, syndromes, weight distributions
4. Galois field structure and arithmetic
5. Cyclic codes: generator polynomials, shift registers
6. BCH and Reed-Solomon codes: definitions and properties
7. BCH and Reed-Solomon codes: decoders for errors and erasures
8. Implementation of Galois field arithmetic and shift register decoders
12. Product codes; coding gain
Communication systems

The familiar communication system block diagram is shown below.

EE 387 covers only one part of communication systems: channel coding

- Design and analysis of block codes for error protection
- Algebraic rather than probabilistic techniques for decoding block codes

Trellis, turbo, and LDPC codes are not covered; see EE 388.

In communications theory, the source output and channel output are probabilistic.
Other communication systems courses

➢ Source coding

Data compression of digital data, quantization of analog data.

➢ Lossless coding: Huffman codes, arithmetic coding, Lempel-Ziv codes

➢ Lossy coding: vector quantization, transform coding and motion estimation used in compressing images and video

Relevant courses: EE 398A, EE 376C, Music 422

➢ Encryption

Private key vs. public key.

Encryption works best with nonredundant plaintext, so data compression improves security

The output of a good encryption unit appears random; channel errors cause error propagation, so good channel coding is very important

Relevant courses: CS 255
Modulation

Modulator generates waveforms suitable for transmission over a physical channel.

- AM: \( x(t) \rightarrow (1 + ax(t)) \sin 2\pi f_c t \) : EE 279
- CD-ROMs, disk drives: 8-to-14 and RLL(1,7) codes satisfy run-length constraints (“symbolic dynamics”): EE 392P (not offered)
- Modems: FSK, PSK, QAM, GMSK, and even fancier signal constellations: EE 359, EE 379

Advances in modulation schemes have resulted in significant improvements in communication rates and storage capacity.

However, channel errors that the demodulator does not correct can result in \textit{error propagation} and must be corrected by the channel decoder.
Relevant results from EE 376A, Information Theory

The Source and Channel Coding Theorems imply the following:

◮ Reliable communication can be achieved at any rate below the capacity of the communication channel — in the limit for large amounts of data.

◮ The communication problem can be broken down into the separate components shown in the above diagram, without loss of reliability or efficiency — in principle.

In EE 387 we will study how to achieve reliable communication with reasonable block sizes and feasible decoding complexity.

◮ We add controlled redundancy to data transmitted over the channel.

◮ This redundancy lowers the raw data rate, but reduces the error rate when the redundancy is used to correct errors.

◮ The net effect is to increase the rate at which reliable data is delivered.
Error control classification

The error control problem can be classified in several ways.

- Type of errors: how much clustering—random, burst, catastrophic
- Type of modulator output: digital ("hard") vs. analog ("soft")
- Type of error control coding: detection vs. correction
- Type of codes: block vs. convolutional
- Type of decoder: algebraic vs. probabilistic

The first two classifications are used to select a coding scheme according to the last three classifications.
Error rate

Consider the channel + modulator portion of the communication diagram.

The raw error rate is the fraction of incorrect output symbols.\
\[
\lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} \Pr(x_i \neq y_i)
\]

(This definition assumes that the demodulator output is hard data.)

Important special case: i.i.d. channel has raw error rate \( \Pr(x_i \neq y_i) \).

Noise is the difference between received and transmitted symbols:
\[
n_i = y_i - x_i \Rightarrow y_i = x_i + n_i.
\]
Noise clustering

Channel noise can be classified by the dependence between noise events.

1. Random: independent noise symbols, perhaps i.i.d. or Bernoulli.
   - Each noise event affects isolated symbols.
   - Symbol error probability may change over time.

2. Burst: noise event causes a contiguous sequence of unreliable symbols.
   - Some causes of burst noise:
     - Noise event is larger in duration or physical size than one symbol
     - Error propagation may occur because of demodulator design. Example: when a (17,16) code is used, a single raw error may affect 16 data bits.

3. Catastrophic: channel becomes unusable for a period of time comparable to or longer than a data packet. Example: ethernet collisions.
   - Retransmission, perhaps at a much later time, is needed because packets are hopelessly corrupted.
Decoding complexity for three types of error clustering

For raw error rate $10^{-4}$, we expect 100 errors per $10^6$ bits.

- An i.i.d. error distribution requires a random error correcting code. For example, a $(1000,960)$ code with 40 check bits for every 1000-bit codeword can correct four errors per block. After decoding, the bit error is reduced to about $10^{-9}$. Redundancy: 4%.

- If burst errors have length 10, that is, one burst every $10^5$ bits, then we can use a single burst error correcting code, e.g., parameters $(2000,1970)$. Redundancy: 1.5%.

- One catastrophe of length 100 or more. Encode packets with 25-bit checksum, say $(5000,4975)$. We lose (and require retransmission of) one packet in 200. Redundancy: 1% (0.5% for retransmission, 0.5% for checksums).

For a fixed bit error rate, the cost of error protection decreases as errors change from random to burst.
Types of error protection

- **Error detection**
  
  Goal: avoid accepting faulty data.
  
  Lost data may be unfortunate; wrong data may be disastrous.
  
  Solution: *checksums* are included in messages (packets, frames, sectors). If any part of the message is altered, then the checksum is *not* valid (with high probability).

- **(Forward) error correction (FEC or ECC).**
  
  Use redundancy in encoded message to estimate from the received data (*senseword*) what message was actually sent.
  
  Optimal estimate is message that is most probable given what is received (MAP, *maximum a posteriori*). The best estimate is typically the message that is “closest” to the senseword.

Error correction is more complex than error detection.

“Proof”: ECC can be used for detection as follows: reject a message if any correction is needed.
Types of error protecting codes

- **Block codes**

  Data is *blocked* into \( k \)-vectors of information digits, then encoded into \( n \)-digit codewords \( (n \geq k) \) by adding \( p = n - k \) redundant check digits.

  There is no memory between blocks. The encoding of each data block is independent of past and future blocks.

  An encoding in which the information digits appear unchanged in the codewords is called *systematic*. 
Convolutional codes

Time-invariant encoding scheme: each \( n \)-bit codeword block depends on current information digits and on the past \( m \) \( n \)-bit information blocks.

The parameter \( m \) is called the *memory order*. The *constraint length* is \((m + 1)n\); this is the number of bits that the decoder must consider.

Here is the simplest convolutional code.

\[
\begin{align*}
m_i & \quad \rightarrow \quad m_{i-1} \\
\downarrow & \quad \quad \downarrow \\
\rightarrow & \quad \quad \rightarrow \\
c^1_i &= m_i \\
\downarrow & \quad \downarrow \\
\rightarrow & \quad \rightarrow \\
c^2_i &= m_i + m_{i-1}
\end{align*}
\]

For this rate 1/2 convolutional code, \( m = 1 \) and \( n = 2 \).

*Exercise*: determine the error correction ability and a decoding method for the above convolutional code.
Error control applications

- Planetary probes (Gaussian noise; ground-based decoders).
- Memory subsystems (9-bit SIMMs for detection, 72-bit DIMMs for correction).
- Computer buses (high speed, short blocklength). In many systems, the possibility of errors is ignored.
- Modems. V.32/V.90 use trellis codes. V.42 uses error detection with retransmission.
- Datacomm networks (Ethernet, FDDI, WAN, WiFi, Bluetooth) (usually error detection only).
- Magnetic disks and tapes (detection for soft errors, correction for burst errors).
- CDs, DATs, minidisks, DVDs. Digital sound needs ECC!
- Satellite TV.
Coding schemes

The above examples exhibit a range of data rates, block sizes, error rates.

- No single error protection scheme works for all applications.

- Choosing a good coding scheme may be difficult because error characteristics are not known.
  
  Common solution: use methods that correct multiple classes of errors.
  Disadvantage: may not be “optimal” for any particular environment.

- Some applications use multiple coding techniques.
  
  - Packet or frame has checksum (CRC) in addition to FEC redundancy.
  
  - A common combination uses an *inner* convolutional code and an *outer* Reed-Solomon block code.

  The R-S code corrects burst errors introduced by the convolutional code.
  One approach to combining codes is call *concatenated codes*. 
The outer code protects 400 24-bit symbol using 4 check symbols. This \((404, 400)\) code can correct up to two symbol errors.

The inner code corrects random bit errors using 5 check bits for each 24 information bits. This \((29, 24)\) binary block code can correct a single error in a 29-bit inner channel codeword.

The outer code corrects short burst errors as well as most miscorrections made by the inner code.

Exercise: analyze the performance of this concatenated code.