UNIT II

Encoding of Convolutional codes

Convolutional codes are a type of error-correcting code used in information and coding theory, particularly in digital communication systems. They are often employed in scenarios where a continuous stream of data needs to be transmitted and errors may occur during transmission. The encoding process for convolutional codes involves the use of shift registers.

Convolutional codes are used in various communication systems, including wireless communication, satellite communication, and digital broadcasting. Their advantage lies in their ability to provide error correction while operating on continuous data streams. The decoding of convolutional codes often involves the use of algorithms such as the Viterbi algorithm, which efficiently finds the most likely sequence of transmitted bits given the received sequence and the characteristics of the code.

It includes ,

* Shift Register Structure
* Connection Polynomial
* Encoder Operation
* Trellis diagram
* Termination

**Generator Matrix in the Time Domain:**

A generator matrix is commonly associated with linear block codes. A generator matrix defines a linear block code by specifying a set of basis vectors for the code. These basis vectors are used to generate codewords from the input message.

In the time domain, the generator matrix is often represented as a matrix where each row corresponds to a basis vector. Let's consider a linear block code with parameters

(n,k), where

n is the length of the codeword and

k is the dimension of the code (the number of information symbols).

The generator matrix

**G=k\*n**

The generator matrix in the time domain specifies how to transform a message vector into a corresponding code word vector. This process is essential for encoding information in a systematic and structured way, particularly in the design and implementation of linear block codes used for error control in communication systems.

In time domain, generation process are

Message Input

Matrix Multiplication

Codeword Output

**State Diagram:**

States: Each circle or node in the diagram represents a state of the system. States capture the current condition or mode of operation of the system.

Transitions: Directed arrows, labeled with input symbols, connect states and represent possible transitions between states. The transition from one state to another occurs based on the input symbol received.

Input Symbols: Input symbols are the signals or symbols that trigger state transitions. The labels on the arrows indicate the specific input symbol that causes a transition from one state to another.

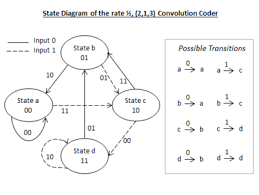
Output Symbols: Some state diagrams include information about the output symbols associated with each state or transition. In coding systems, these symbols could represent the encoded bits or symbols generated during the encoding process.

Initial State: One state is typically designated as the initial state, representing the starting point of the system or process.

Final States or Acceptance States: In some systems, specific states may be marked as final or acceptance states. These states represent the end of a process or successful completion.

Cycles: State diagrams may contain cycles, indicating that the system can return to a previous state under certain conditions.

State diagrams are powerful tools for understanding and designing finite-state machines. They provide a visual representation of the system's behavior, making it easier to analyze and optimize the performance of encoding and decoding processes in communication systems.



**Code Termination**

In theory, the code sequences of convolutional codes are of infinite length, but for practical applications we usually employ finite sequences. In order to obtain a terminated code sequence, we start encoding in the all-zero encoder state and we ensure that, after the encoding process, all memory elements contain zeros again. In the case of an encoder without feedback this can be done by adding k · m zero bits to the information sequence. Let L denote the number of information blocks, i.e. we encode information bits.

**Truncation:** We stop encoding after a certain number of bits without any additional effort. This leads to high error probabilities for the last bits in a sequence.

**Termination:** We add some tail bits to the code sequence in order to ensure a predefined end state of the encoder, which leads to low error probabilities for the last bits in a sequence.

**Tail-biting:** We choose a starting state that ensures that the starting and end states are the same. This leads to equal error protection.

Puncturing :

Puncturing is often used in the context of error-correcting codes, such as convolutional codes or Reed-Solomon codes, to achieve different code rates or to adapt the code for specific applications.

**Code Rate Adjustment:** Puncturing is commonly employed to adjust the code rate, which is the ratio of the information bits to the total number of bits in the encoded sequence. By puncturing certain bits, the code rate can be increased or decreased

**Error-Correcting Capability**: Puncturing affects the error-correcting capability of a code. In some scenarios, sacrificing some redundancy (puncturing) may be acceptable if it leads to a higher code rate, allowing for a higher data transfer rate.

**Bandwidth Efficiency**: Puncturing can be used to optimize the bandwidth efficiency of a communication system.

**Compatibility with Standards:** In practical applications, puncturing is often employed to adapt a code to meet specific standards or requirements. Standards for communication systems may specify certain code rates, and puncturing provides a way to achieve those rates by selectively removing bits.

**Iterative Decoding**: In some iterative decoding algorithms, such as Turbo codes or LDPC (Low-Density Parity-Check) codes, puncturing is used to create irregular codes. This irregularity can improve the performance of iterative decoding algorithms.

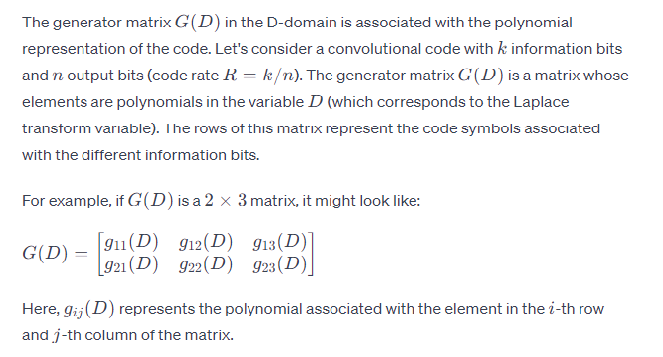
The process of puncturing involves systematically removing bits from the encoded sequence based on a predetermined pattern or algorithm. The receiver must be aware of the puncturing pattern used by the transmitter to correctly decode the received sequence.

Puncturing is a technique used to balance factors such as code rate, error correction capability, and bandwidth efficiency, depending on the specific requirements of a communication system.

**Generator Matrix in the D-Domain**

The concept of a "generator matrix in the D-domain" is typically associated with linear block codes and their representation in terms of polynomial codes, often used in digital communication and coding theory. The D-domain refers to the Laplace transform domain, and it is commonly used when dealing with convolutional codes.

In the context of convolutional codes, the generator matrix is often represented using polynomials in the D-domain. Convolutional codes are a class of error-correcting codes where encoding is performed by passing the information through shift registers and combining the outputs



The encoding process involves multiplying the information vector by this generator matrix in the D-domain to produce the encoded output. The resulting polynomial is then transformed back to the time domain to obtain the transmitted codeword.

It's important to note that the D-domain representation is particularly useful when analyzing convolutional codes and their properties, as it allows for a concise and convenient representation of the encoding process in terms of polynomials.

**Encoder Properties**

Encoders play a crucial role in transforming input data into a coded format for transmission or storage. The properties of encoders are essential considerations in designing reliable and efficient communication systems. Here are some key properties of encoders:

**Linearity:**

A linear encoder satisfies the principle of superposition, meaning that the encoding operation distributes over additions and scalar multiplications.

**Systematic Encoding:**

Systematic encoding is a desirable property where a portion of the encoded sequence remains unchanged and is identical to the original message. This facilitates easier decoding and error correction.

**Error-Correction Capability:**

Some encoders, especially in the context of error-correcting codes, are designed to introduce redundancy into the encoded message to enable the detection and correction of errors at the receiver.

**Efficiency and Code Rate:**

The efficiency of an encoder is often measured by its code rate, which is the ratio of the number of information bits to the total number of bits in the encoded sequence. Higher code rates generally result in more efficient use of bandwidth but may come at the expense of error correction capability.

**Memoryless Property:**

Memoryless encoding means that the encoding of a symbol is independent of the context or the symbols that came before it. In other words, the encoding of a symbol is determined solely by the value of that symbol.

**Block or Convolutional Encoding:**

Encoders can be categorized as block encoders or convolutional encoders. Block encoders process fixed-size blocks of input symbols, while convolutional encoders operate on a continuous stream of input symbols. The choice between these types depends on the specific requirements of the communication system.

**Parallelism:**

Some encoders can operate in parallel, encoding multiple symbols simultaneously. This property can be advantageous in terms of speed and efficiency.

**Adaptability:**

In certain applications, encoders may need to adapt to changing channel conditions or requirements. The ability to adjust parameters or coding schemes to optimize performance is an important property.

The specific properties of an encoder depend on its intended application, the characteristics of the communication channel, and the requirements of the overall system. Designing an encoder with the right combination of properties is crucial for **achieving reliable and efficient communication**.

**Examples :**

* Hamming codes
* Reed Solomon codes

Understanding and analyzing the minimum distance of a code is crucial for selecting an appropriate code for a specific application. A balance must be struck between achieving a high code rate and maintaining a sufficiently large minimum distance to meet the error detection and correction requirements of the system.

**Trellis Diagram**

A trellis diagram is a graphical representation used in information theory, particularly in the context of convolutional codes and trellis-coded modulation. It provides a visual way to understand the encoding and decoding processes of a convolutional code or a trellis-coded modulation scheme. Trellis diagrams are especially helpful for illustrating the state transitions in the encoder and the paths taken during decoding.

**States:**

In a trellis diagram, each vertical column represents a state. States represent the internal memory of the encoder at a given time. The number of states corresponds to the memory of the convolutional code.

**Nodes:**

Each state in the trellis diagram is associated with multiple nodes, each representing a possible input symbol. The number of nodes per state is equal to the number of possible input symbols.

**Branches:**

Branches connect nodes between adjacent states. Each branch corresponds to a specific transition from one state to another based on the input symbol. The direction of the branch indicates the transition, and the label on the branch indicates the input symbol that causes the transition.

**Path:**

A path through the trellis diagram corresponds to a sequence of input symbols and the resulting sequence of output symbols. Paths represent possible code sequences generated by the convolutional code.

**Termination:**

The trellis diagram typically extends over time, with a starting state and an ending state. The termination state represents the completion of the encoding or decoding process.

Trellis diagrams provide a compact and visual representation of the operation of convolutional codes and are instrumental in understanding decoding algorithms like the Viterbi algorithm. They are also used in trellis-coded modulation for signal constellations.

**Viterbi algorithm**

The Viterbi algorithm is a dynamic programming algorithm used for decoding convolutional codes in information theory. It was developed by Andrew Viterbi and has become a widely used method for finding the most likely sequence of states in the presence of noise and errors.

The algorithm is particularly effective in decoding convolutional codes, which are a type of error-correcting code commonly used in digital communication.

**Importance of Viterbi Algorithms**

1. Convolutional Code Representation:

A convolutional code is represented by a trellis diagram. The trellis diagram illustrates the transitions between different states based on the input symbols. Each path through the trellis corresponds to a possible sequence of input symbols and the resulting encoded output.

2. Initialization:

The algorithm begins with an initialization step. The initial state metrics and path metrics are set based on the received signal. The state metric represents the likelihood of being in a particular state, and the path metric represents the likelihood of the received sequence given a particular path through the trellis.

3. Recursion:

The algorithm then proceeds with a recursive step, moving through each time step in the trellis. At each time step, the algorithm calculates the metrics for each possible transition to a state by considering both the current state and the previous state.

4. Path Metric Update:

For each possible transition, the algorithm updates the path metric by combining the previous path metric and the branch metric (a measure of the similarity between the received symbol and the expected symbol based on the current state). The transition with the lowest path metric is selected as the survivor path.

5. Traceback:

After processing all time steps, the algorithm performs a traceback operation to find the most likely sequence of states and hence the most likely transmitted message. This is done by backtracking through the trellis, selecting the transitions with the lowest path metrics at each time step.

6. Output:

The final output is the decoded message obtained from the traceback operation.

**Key Concepts:**

Branch Metric: A measure of the similarity between the received symbol and the expected symbol based on the current state.

Path Metric: The accumulated metric for a particular path through the trellis, representing the likelihood of the received sequence given that path.

State Metric: The metric representing the likelihood of being in a particular state at a given time step.

The Viterbi algorithm is widely used in communication systems, including wireless communication, satellite communication, and digital storage systems, where convolutional codes are employed for error correction. It is known for its efficiency and optimality in finding the most likely transmitted sequence in the presence of noise and errors.

**Free Distance:**

The term "free distance" typically refers to the free distance of a linear code. The free distance is an important parameter that characterizes the error-correcting capability of a linear code.

**Free Distance of a Linear Code:**

Definition:

The free distance, denoted as df of a linear code is the minimum Hamming distance between any two distinct codewords in the code, excluding the all-zero codeword. In other words, it is the minimum number of bit positions in which two codewords can differ while still being considered distinct.

Importance:

The free distance is a crucial parameter in determining the error correction capability of a linear code. A larger free distance generally implies better error-correcting properties. It indicates the ability of the code to correct a certain number of errors or detect a greater number of errors in a received codeword.

**Error Correction:**

The free distance is related to the code's ability to correct errors. Specifically, if the free distance is

**Relationship with Minimum Distance:**

The free distance is always less than or equal to the minimum distance

The minimum distance includes the possibility of the all-zero codeword, while the free distance specifically excludes it.

**Applications:**

Understanding the free distance is crucial in the design and analysis of error-correcting codes. It helps code designers make informed decisions about the code's parameters, such as code rate and minimum distance, based on the desired error correction capabilities for a specific communication channel.

The free distance of a linear code is a measure of the minimum Hamming distance between non-zero codewords, providing valuable insights into the code's ability to correct errors in a noisy communication environment.

**Active Distance:**

In information theory and coding theory, the commonly used terms are "Hamming distance," "weight," and "distance."

Hamming Distance:

The Hamming distance between two equal-length strings of symbols is the number of positions at which the corresponding symbols differ. In the context of linear codes, the minimum Hamming distance is the smallest Hamming distance between any two distinct codewords in the code. It is a key parameter indicating the error-detecting and error-correcting capabilities of a code.

Weight of a Codeword:

The weight of a codeword in a linear code is the number of nonzero elements (symbols) in the codeword. There are different types of weights, such as Hamming weight and Manhattan weight, depending on the context. In Hamming weight, the weight is the number of non-zero components in the codeword.

Weight Distance:

The term "weight distance" is not a standard term in information theory or coding theory. However, there is a concept related to weights called "minimum weight," which refers to the smallest weight among all nonzero codewords in a linear code. The minimum weight is closely related to the minimum Hamming distance. The Hamming distance and weights of codewords are essential concepts. The minimum Hamming distance is a key parameter for error correction, and the weight of a codeword provides information about its structure.

weight enumerators and path enumerators are tools used to analyze and describe the properties of codes, especially linear block codes.

**Weight Enumerator**:

Definition:

The weight enumerator is a generating function that counts the number of codewords in a linear block code according to their weights. The weight of a codeword is defined as the number of non-zero elements (symbols) in the codeword.

Function:

The weight enumerator is represented as a polynomial, where each term corresponds to the number of codewords with a specific weight. For a code of length

n, the weight enumerator is often denoted as W(x1,x2,x3.............xn) where xi corresponds to the weight i.

Use:

The weight enumerator is useful in studying the error-correcting capabilities of a code. By analyzing the weight distribution of codewords, one can gain insights into the code's ability to detect and correct errors.

**Path Enumerator:**

Definition:

The path enumerator is related to trellis-based decoding algorithms, particularly in the context of convolutional codes. It counts the number of paths through the trellis that correspond to different error patterns.

Function:

Similar to the weight enumerator, the path enumerator is represented as a polynomial. Each term in the polynomial corresponds to the number of paths with a specific error pattern.

Use:

The path enumerator is used in the analysis of convolutional codes, especially in understanding the performance of algorithms like the Viterbi algorithm. It helps in studying the probability of different error patterns and aids in optimizing the design of convolutional codes.

Pairwise Error Probabilities

the pairwise error probability (PEP) is a measure that quantifies the probability of making an error when decoding a specific pair of codewords in a communication system. It is particularly relevant in scenarios where error-correcting codes are used, and the decoder must distinguish between different codewords.

Importance of Pairwise Error Probabilities

The pairwise error probability is the probability that the received signal is decoded as one codeword when it was actually transmitted as a different codeword. It is a measure of the likelihood of confusion between pairs of codewords.

Mathematically, the pairwise error probability is often denoted as Pe(Ci,Cj)

Ci Cj as codewords

​ It can be expressed as the probability of making an error in distinguishing between two specific codewords.

Understanding the pairwise error probability is crucial for evaluating the performance of error-correcting codes. A lower pairwise error probability indicates a better ability of the code to discriminate between codewords, leading to improved error correction.

Code Design:

Code designers often aim to design codes with low pairwise error probabilities, especially for codes used in communication systems. A well-designed code should have a high minimum distance and low probability of confusion between codewords.

Relation to Minimum Distance:

The pairwise error probability is closely related to the minimum distance of a code. In general, codes with larger minimum distances tend to have lower pairwise error probabilities.

Trade-off with Code Rate:

There is often a trade-off between the rate of a code (the ratio of information bits to total bits) and the pairwise error probability. Increasing the code rate may lead to a decrease in minimum distance and an increase in the pairwise error probability.

Performance Analysis:

Analyzing the pairwise error probability provides insights into the performance of a code under various channel conditions. It helps in assessing the effectiveness of the code in correcting errors and maintaining reliable communication.

The pairwise error probability is a key metric in the evaluation of error-correcting codes. Code designers strive to design codes with low pairwise error probabilities to enhance the code's error correction capabilities in communication systems.

**Viterbi Bond**

The Viterbi Bound is a theoretical bound on the performance of maximum likelihood decoding, and it is often used in the context of convolutional codes. Specifically, it provides an upper bound on the probability of error when using the Viterbi algorithm for decoding.

**Importance of Viterbi Bound**

**Viterbi Algorithm:**

The Viterbi Algorithm is a maximum likelihood decoding algorithm commonly used for decoding convolutional codes.

**Viterbi Bound:**

The Viterbi Bound is an upper bound on the probability of error for the Viterbi Algorithm. It provides an estimate of the decoding performance achievable by the algorithm.

**Performance Limit:**

The Viterbi Bound is a theoretical limit that helps assess the potential performance of the Viterbi Algorithm. It sets a benchmark for the algorithm's ability to correct errors in the presence of noise.

**Trade-off with Complexity:**

While the Viterbi Algorithm is powerful in achieving near-optimal decoding performance, it comes with increased computational complexity. The Viterbi Bound helps in understanding the trade-off between performance and computational complexity.

**Use in Code Design:**

The Viterbi Bound is often used in the design and analysis of convolutional codes. It provides insights into the achievable error correction capabilities and aids in optimizing parameters such as code rate and constraint length.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*