Convolutional Codes

Kandidat Projekt

Forfatter: Esben Dissing Bregnballe

Matematik

Aalborg Universitet

Dato: 18.12.2024

AALBORG UNIVERSITET
STUDENTERRAPPORT

Institut for Matematiske Fag
Matematik
Skjernvej 4B
9220 Aalborg
https://math.aau.dk

Title:
Convolutional Codes

Project:
Kandidat Projekt

Project period:
1. September 2024 - 18. December 2024

Author:
Esben Bregnballe

Supervisor:
Matteo Bonini

Abstract:

Error correcting codes are a method of correcting errors in data transmitted over an inaccurate channel. One traditional method, linear block codes, work by dividing the data to be transmitted into messages and additional redundant data is then appended to each message, the message and and redundancy combined being called the codewords. The redundant data can then be used to correct errors that occur in transmission. A disadvantage of this method is that to decode the data, it is necessary to know where one codeword ends and another one starts. If the transmission channel is of a kind that can lose bits without the receiver knowing, the alignment of codeword borders can be lost. This is the problem convolutional codes attempts to solve. Unlike linear block codes, convolutional codes encode data as continous streams. Each symbol is encoded using not only the current data, but also a specified number of preceding symbols.

Contents

Chapter 1

Introduction

Error correcting codes are a method of correcting errors in data transmitted over an inaccurate channel. One traditional method, linear block codes, work by dividing the data to be transmitted into messages and additional redundant data is then appended to each message, the message and and redundancy combined being called the codewords. The redundant data can then be used to correct errors that occur in transmission. A disadvantage of this method is that to decode the data, it is necessary to know where one codeword ends and another one starts. If the transmission channel is of a kind that can lose bits without the receiver knowing, the alignment of codeword borders can be lost. This is the problem convolutional codes attempts to solve. Unlike linear block codes, convolutional codes encode data as continous streams. Each symbol is encoded using not only the current data, but also a specified number of preceding symbols. This still means a missing symbol can cause decoding errors, but it is possible to reestablish decoding afterwards.

Convolutional codes represent their codewords with vectors with entries in the polynomial ring over finite fields. Rings are a generalization of the concept of fields, where there is no requirement of a multiplicative inverse. Many of the known results of finite fields also apply in some way for polynomial rings.

Like with linear block codes, convolutional codes can be represented by generator matrices that, in this case, have polynomials as entries. There is the concept of equivalence, where two generator matrices can be transformed into each other by a unimodular matrix, the module-equivalent of a non-singular matrix. This leads to certain notions of canonical forms, in particular using a so called left prime generator matrix to represent a code.

The usual method of comparing codewords in linear block codes, the Hamming distance, can be extended to convolutional codes, and another method, called the j-column distance, also exists. In the same vein, an analogue of the singleton bound and other bounds that constrain the performance of convolutional codes also have a role to play.

The Viterbi algorithm is a useful minimum distance decoding algorithm for convolutional codes. It works by interpreting the received data as a graph and using a path finding algorithm inspired by Dijkstra's algorithm to find the codeword closest to the received word.

Chapter 2

Foundations of Convolutional Codes

2.1 Modules

In order to describe convolutional codes, it is necessary to use modules, so a short introduction to this construction is given here. A module is a generalization of a vector space, where the scalars are no longer required to be a field, but can instead be rings.

This section relies on [9] for most of the definitions relating to modules, [11] for the theorem about a free module over a PID and [8] for the definition of a unimodular matrix.

Definition 2.1.1 (Module)**.** A (left) module is M over a ring R with multiplicative identity 1_R is an abelian group with operation $+$, and a scalar product

$$\cdot : R \times M \to M,$$

satisfying the following conditions:

- $\alpha(\beta m) = (\alpha\beta)m$
- $(\alpha + \beta)m = \alpha m + \beta m$
- $\alpha(m + n) = \alpha m + \alpha n$
- $\alpha(m + n) = \alpha m + \alpha n$
- $1_R m = m$

This is just a restatement of the axioms of a linear space, where field is replaced with ring.

Unlike linear spaces, not all modules have a basis. For example, if one considers the module of rational numbers $\mathbb{Q}$ over the ring of whole numbers $\mathbb{Z}$. This can be shown by a simple argument. Assume all rationals are in irreducible form. Obviously a single rational number can not be a basis, since any rational not sharing a denominator of the candidate can not be reached by multiplication with a whole number. Then, by induction. Any pair of rational numbers $\frac{a}{b}, \frac{c}{d}$, are linearly dependent, shown by multiplying one by its denominator and the numerator of the second $bc\frac{a}{b} = ac$, giving a multiple of the second rational. Finally, assume for $n \geq 2$ that any set of n rationals are linearly dependent. Then, for a set of $n + 1$ rationals pick one and multiply it to be a multiple of one of the others, as done above, showing

that any $n+1$ rationals are linearly dependent. There exists no basis for the rationals over the whole numbers.

Even though not all modules have a basis, since all vector spaces are modules some obviously do. A module with a basis is called a free module.

Definition 2.1.2 (Free Module)**.** A module M is a free module if and only if it has a basis.

For modules, the equivalent to the dimension of a linear space is called the rank of the module.

Definition 2.1.3 (Rank of Module)**.** The minimum cardinality of a basis spanning a module M is called the rank of M.

Submodules of a free module over a PID are free modules, by the following reasoning.

Theorem 2.1.1. *Let R be a PID and M a finitely generated free module over R with* $\operatorname{rank} m \geq 1$*. Further, let K be a submodule of M. Then, K is a free R-submodule with rank $n \leq m$.*

Proof. The proof is by induction. If $m = 1$, then $M \cong R$ and every submodule is an ideal with rank ≤ 1, and the generator for each ideal works as a basis.

Assume that the theorem holds up to $m-1$: If $K = \{0\}$ then the theorem holds by an empty basis. Otherwise, consider the following. Let $\pi_i \colon M \to R$ be the projection map that gives the ith entry of an element of the module in a basis. Then $\ker \pi_i$ is a free module over R with rank $m-1$. For some $i \in [m]$, $\pi_i(K) \neq \{0\}$. In that case, $\pi_i(K)$ is a nonzero ideal. Because it is an ideal, it is also a free submodule of M with rank 1. Therefore, the intersection $\ker \pi_i \cap K$ is a submodule of $\ker \pi_i$. By the induction hypothesis, the rank of $\ker \pi_i \cap K \leq m-1$.

Let α be a generator for $\pi_i(K)$ and $v \in K$ be an element such that $\pi_i(v) = \alpha$. It is clear that $K = \ker \pi_i \cap K \oplus vR$, since all elements of K can be represented by a linear combination of an element of K that is zero in the ith entry and a multiple of a generator of all the i entries of elements of K. If $\{v_1, \ldots, v_{m-1}\}$ is a basis for $\ker \pi_i \cap K$, then it can be extended to a basis of $K = \ker \pi_i \cap K \oplus vR$ by adding v as v is a generator for vR. Hence, K has a basis and $\operatorname{rank} K \leq m \leq n$. □

If you have a basis for a module or a pair of modules, you can represent linear maps as matrices, and if the ring is abelian matrix multiplication is even associative.

It then makes sense to ask if it is possible to have matrix inverses, which motivates the definition of unimodular matrices.

Definition 2.1.4 (Unimodular matrix)**.** A $k \times k$ matrix $U(z)$ with entries in a ring R is a unimodular matrix if there exists a $k \times k$ matrix $V(z)$ such that

$$U(z)V(z) = V(z)U(z) = I$$

2.2 Convolutional Codes

Conventional error-correcting codes work by dividing the data to be transmitted into blocks and adding redundant information. These block codes have the disadvantage, that in the instance where your channel drops bits and you loose track of the time index of your symbols, for example when communicating with a satellite, it is difficult to find the alignment of your blocks and thus apply the error-correction.

This is the problem solved by convolutional codes. To decode convolutional codes, only the previous couple of symbols are needed. This is relative to the index of the word to be decoded, and does not depend on any global alignment properties. Therefore convolutional codes are suitable for channels with unreliable index of symbols.

This section references [1] and [8] for basic definitions of convolutional codes, and [3] for the existence proofs for different canonical forms of generator matrices.

Definition 2.2.1 (Convolutional Code)**.** Let $k \leq n$ be positive whole numbers. A $(n, k)_q$ convolutional code $\mathcal{C}$ is a $\mathbb{F}_q[z]$-submodule of $\mathbb{F}_q[z]^n$ with rank k. A matrix $G(z)$ whose rows form a basis for the submodule is called a generator matrix of $\mathcal{C}$.

As with linear block codes, there are multiple generator matrices of the same code, although for modular matrices it doesn't follow automatically that two matrices with the same rowspace can be transformed into each other with a linear bijection. It therefore makes sense to consider when two generator matrices of a code are considered equivalent.

Definition 2.2.2 (Equivalence of Generator Matrices)**.** Two generator matrices $G(z), \tilde{G}(z)$ of an $(n, k)_q$ convolutional code $\mathcal{C}$ are called left equivalent if there exists a unimodular matrix $U(z) \in \mathbb{F}_q[z]^{k \times k}$, such that

$$\tilde{G}(z) = U(z)G(z)$$

Similarly, they are called right equivalent if there exists a unimodular matrix $V(z) \in \mathbb{F}_q[z]^{k \times k}$, such that

$$\tilde{G}(z) = G(z)V(z)$$

Finally, in general they are called equivalent if

$$\tilde{G}(z) = U(z)G(z)V(z)$$

The following basic row and column operations can be defined on polynomial matrices

1. Multiplication of a row (column) by a constant c.
2. The adding of one row (column) multiplied by a polynomial $b(z)$ to another row (column).
3. The exchange of any two rows (columns).

Each of these operations can be represented by simple unimodular matrices, namely the result of applying these operations to the identity matrix, and the determinant is nonzero and doesn't depend on z, i.e the determinant is in $\mathbb{F}_q \setminus \{0\}$. For row operations the matrix is applied on the left, for column operations a matrix is applied on the right.

Again, like with linear codes, there exists canonical forms of generator matrices in the equivalence classes of equivalent generator matrices.

It is possible to represent a generator matrix in upper triangular form. This is called column Hermite form.

Theorem 2.2.1 (Column Hermite Form)**.** *Let $G(z) \in \mathbb{F}_q[z]^{k\times n}$, with $k \leq n$. Then there exists a unique unimodular matrix $U(z)$ such that,*

$$U(z)G(z) = \begin{bmatrix} h_{11}(z) & h_{12}(z) & \cdots & h_{1k}(z) & h_{1,k+1}(z) & \cdots & h_{1n}(z) \\ & h_{22}(z) & \cdots & h_{2k}(z) & h_{2,k+1}(z) & \cdots & h_{2n}(z) \\ & & \ddots & \vdots & \vdots & & \vdots \\ & & & h_{kk}(z) & h_{k,k+1}(z) & \cdots & h_{nn}(z) \end{bmatrix},$$

where the h_{ii} are monic polynomials with $\deg h_{ii} > \deg h_{ji}$ for all $i, j \in [k]$ such that $j < i$. This is called the column Hermite form.

Proof. The proof works by repeatedly applying row operations to $G(z)$ until it is in the correct form. Since row operations can be represented by matrices applied from the left, by associativity $U(z)$ is the product of these matrices. The method works similarly to Gaussian elimination.

1. First, assuming that the first column of $G(z)$ is not identically zero, pick an element with minimal degree and exchange so that it is the first row.

2. Divide all entries of the columns a_{i1} with the entry in the first row a_{11} using polynomial division, giving $a_{i1} = a_{11}q_{i1} + r_{i1}$, and for each row beneath the first, subtract q_{i1} times the row.

3. If not all remainders r_{i1} are identically zero, pick a remainder of minimal degree, exchange it to the first row and go to step 2

4. Once the entries beneath a_{11} are identically zero, apply the whole algorithm to the submatrix with the first column and first row removed.

Since the remainder of division is of lower degree than divisor, the process terminates. If the entries on the diagonal of the resulting matrix are not identically zero, one can use row addition to reduce the degree of entries above the diagonal entry, whereas if an entry on the diagonal is identically zero then all the entries above it are zero as well.

All of this is done using row operations, so it can be represented using matrices and the product of these is a $U(z)$ satisfying the theorem. □

It is possible to represent a matrix in lower triangular form. This is called row Hermite form.

Theorem 2.2.2 (Row Hermite Form)**.** *Let $G(z) \in \mathbb{F}_q[z]^{k\times n}$, with $k \leq n$. Then there exists a unique unimodular matrix $V(z)$ such that,*

$$G(z)V(z) = \begin{bmatrix} h_{11}(z) & & & & 0 & \cdots & 0 \\ h_{21}(z) & h_{22} & & & \vdots & & \vdots \\ \vdots & & \ddots & & \vdots & & \vdots \\ h_{k1}(z) & h_{k2} & \cdots & h_{kk} & 0 & \cdots & 0 \end{bmatrix},$$

where the h_{ii} are monic polynomials with $\deg h_{ii} > \deg h_{ij}$ for all $i, j \in [k]$ such that $j < i$. This is called the row Hermite form.

Proof. The proof of this theorem is done using the same method as for the column hermite form, except that column operations are used instead of row operations, and all mentions of columns and rows are transposed. □

A corollary follows from the previous two results. It is already known that row and column operations are invertible linear operations with scalar nonzero determinant, but it turns out that any square matrix with scalar nonzero determinant is equivalent to the identity matrix.

Corollary 2.2.1. *If the determinant of a square polynomial matrix $G(z)$ is in $\mathbb{F}_q \setminus \{0\}$, then it can be written as the product of row (column) operations.*

Proof. The proof will be done for the case of row operations. The argument for column operations is essentially identical, just with columns and rows transposed.

First, put the matrix in column Hermite form $U(z)G(z)$. Since $U(z)$ consists of a product of row operations whose determinants are nonzero scalar values, the determinant of $U(z)G(z)$ is in $\mathbb{F}_q \setminus \{0\}$. Therefore, since $\det U(z)G(z) = h_{11}h_{22}\cdots h_{kk}$ is a nonzero scalar each h_{ii} must also be in $\mathbb{F}_q \setminus \{0\}$, and hence h_{ji} identically zero for all $j < i$. It can therefore be concluded that $U(z)G(z)$ is a diagonal matrix with scalars in the diagonal entries. By applying row scaling this matrix can be turned into the identity matrix. The product of the row scalors and $U(z)$ is a product of row operations that transforms $G(z)$ into the identity matrix, and since row operations have inverses that are also row operations, the inverse of this matrix is a product of row operations that transforms the identity matrix into $G(z)$.

In other words, $G(z)$ is a product of row operations. □

Finally, there is a form where a generator matrix is represented by a diagonal matrix. This would be the analogue of a systematic generator matrix for convolutional codes.

Theorem 2.2.3 (Smith Form)**.** *Let $G(z) \in \mathbb{F}_q[z]^{k\times n}$, with $k \leq n$. Then there exists unique unimodular matrices $U(z)$ and $V(z)$ such that,*

$$U(z)G(z)V(z) = \begin{bmatrix} \gamma_1(z) & & & & 0 & \cdots & 0 \\ & \gamma_2(z) & & & \vdots & & \vdots \\ & & \ddots & & \vdots & & \vdots \\ & & & \gamma_k(z) & 0 & \cdots & 0 \end{bmatrix},$$

where for every $i \in [k]$, γ_i is a monic polynomial with the property that γ_{i+1} divides γ_i. These polynomials are uniquely determined by $G(z)$ and are called the invariant polynomials of $G(z)$.

Proof. Start with $G(z)$, and do the following steps.

1. Choose an entry with least degree that is not identically zero. By row and column exchange, move it to the $(1, 1)$ entry.

2. Do polynomial division of all entries of the first row and first column by g_{11}, giving:

$$g_{i1} = g_{11}q_{i1} + r_{i1}, \quad g_{1j} = g_{11}q_{1j} + r_{1j},$$

for all $i \in [k]$, $j \in [n]$.

3. If all these divisions have zero remainders move to the next step, otherwise do the following.

   If the entry is of the form g_{i1}, then add the first row multiplied by q_{i1}, this will result in $(i, 1)$ having the value r_{i1} which has a lower degree than g_{11}. Exchange the entry with g_{11} and go to the second step.

   If the entry is of the form g_{1j}, then do the same with column addition and multiplication with q_{1j}.

4. When all the remainders are zero, do row addition to zero out the first column and column addition to zero out the first row, except for g_{11}.

5. If any of the entries g_{ij} with $i, j \geq 2$ have nonzero remainders after division by g_{11} then add the jth column to the first and go to the second step.

Since the degree of the $(1, 1)$ entry is strictly decreasing, this process will eventually terminate and all remainders will eventually be zero. The resulting matrix from this process has the correct entry in the upper left entry for the desired diagonal matrix and the lower right part is divisible by this entry. To finish the process, iteratively apply the process to the submatrices with rows and columns corresponding to the finished diagonal entries removed, and apply scalar row multiplication with the leading coefficients of entries to ensure the entries are monic.

The result is a diagonal matrix, where the lower entries are divisible by the higher entries. This is matrix is obtained by row and column operations that can be represented by unimodular matrices applied on the left and right. □

Definition 2.2.3 (Row Degree)**.** Let $G(z)$ be a polynomial matrix. The maximal degree of any entry in the ith row of $G(z)$ is called the ith row degree, and is often denoted ν_i.

Two equivalent generator matrices have equal k×k minors, up to multiplication by a constant, because they differ by multiplication by a unimodular matrix and multiplication by a matrix with determinant in $\mathbb{F}_q \setminus \{0\}$ only changes the determinant by a constant factor. Hence, the following definition makes sense.

Definition 2.2.4 (Degree of Code)**.** Let $\mathcal{C}$ be an $(n, k)_q$ convolutional code. The maximal degree of all the $k \times k$ minors of a generator matrix of $\mathcal{C}$ is called the degree of $\mathcal{C}$, and is denoted by δ. The notation $(n, k, \delta)_q$ is used to denote a code with rank k and degree δ in $\mathbb{F}_q[z]^n$. For every $i \in [k]$, the largest degree of an entry in the ith row of a generator matrix of $\mathcal{C}$, is called the ith row-degree of $\mathcal{C}$.

For a generator matrix $G(z)$ of an $(n, k, \delta)_q$ convolutional code $\mathcal{C}$, with row-degrees $\nu_1, \ldots, \nu_k$, it holds that $\delta \leq \nu_1 + \cdots + \nu_k$. If they are equal, then $G(z)$ is called row-reduced and is a minimal generator matrix for $\mathcal{C}$.

An important propery of polynomial matrices is left-primeness.

Definition 2.2.5 (Left Prime Matrix)**.** A polynomial matrix $G(z) \in \mathbb{F}_q[z]^{k\times n}$ with $k \leq n$, is called left prime if for all factorizations $G(z) = U(z)\tilde{G}(z)$ where $U(z) \in \mathbb{F}_q[z]^{k\times k}$ and $\tilde{G}(z) \in \mathbb{F}_q[z]^{k\times n}$, the left matrix $U(z)$ is unimodular.

The importance of the property can be seen from the following equivalence result, which shows that left-primeness correspond to many properties that, at first glance, one might assume are different.

Theorem 2.2.4 (Equivalence theorem for left prime)**.** *Let $G(z) \in \mathbb{F}_q[z]^{k\times n}$, with $k \leq n$. The following are equivalent*

1. *$G(z)$ is left prime*
2. *The Smith form of $G(z)$ is $[I_k\ 0]$.*
3. *The row Hermite form of $G(z)$ is $[I_k\ 0]$.*
4. *$G(z)$ has a right inverse polynomial matrix*
5. *There exists a $L(z) \in \mathbb{F}_q[z]^{(n-k)\times n}$ such that*

$$\begin{bmatrix} G(z) \\ L(z) \end{bmatrix}$$

   is unimodular.

6. *The ideal generated by all the kth order minors of $G(z)$ is $\mathbb{F}_q[z]$.*
7. *For all $u(z) \in \mathbb{F}_q(z)^k$, where $\mathbb{F}_q(z)$ is the field of rational functions with coefficients in $\mathbb{F}_q$, $u(z)G(z) \in \mathbb{F}_q[z]^n$ implies $u(z) \in \mathbb{F}_q[z]^k$.*
8. *$\operatorname{rank} G(\lambda) = k$ for all $\lambda \in \overline{\mathbb{F}}_q$, where $\overline{\mathbb{F}}_q$ denotes the algebraic closure of $\mathbb{F}_q$.*

If $G(z)$ is a left prime generator matrix for a convolutional code $\mathcal{C}$, then $\mathcal{C}$ is considered a noncatastrophic code. Let $\mathcal{C}$ be a noncatastrophic $(n, k, \delta)_q$ convolutional code and $G(z) \in \mathbb{F}_q[z]^{n\times k}$ be a generator matrix for $\mathcal{C}$. Then there exists a matrix $H \in \mathbb{F}_q[z]^{(n-k)\times n}$ such that

$$Hc(z)^T = 0 \Leftrightarrow c(z) \in \mathcal{C}$$

called the parity check matrix.

2.3 Distances of Convolutional Codes

Distances between codewords are an important way to measure a codes ability to correct errors. A basic assumption in error correction is that some errors are more likely than others, and the distance metric used implicitly reflects the partial order of assumed probabilities. Convolutional codes allows for a version of the Hamming weight, corresponding to the scenarion where all single-symbol errors are considered independently equally likely.

This section uses [1] and [8] for basic definitions of distance metrics and [10] and [4] for the singleton and column distance bounds and necessary prerequisite lemmas and theorems. In addition [4] is also the source for the lemma used in defining MDP, the MDP criterion and the fact that the dual of an MDP convolutional code is MDP.

A generalization of Hamming weight can thus be defined for polynomial vectors.

Definition 2.3.1 (Weight and Distance of Convolutional Code)**.** The weight $\mathrm{wt}(c(z))$ for $c(z) \in \mathbb{F}_q[z]^n$ is defined as

$$\mathrm{wt}(c(z)) = \sum_{i=1}^{\deg(c(z))} \mathrm{wt}(c_i)$$

where $\mathrm{wt}(c_i)$ is the Hamming weight of the ith coefficient of $c(z)$. This, of course, induces a distance $d(c_1(z), c_2(z)) = \mathrm{wt}(c_2(z) - c_1(z))$.

In addition to weight, it is also possible to define an analogue to minimum distance. The free distance of a code is defined as follows.

Definition 2.3.2 (Free Distance)**.** The free distance of a code $\mathcal{C}$ is defined as

$$d_{\mathrm{free}} = \min\{\mathrm{wt}(c(z) \mid c(z) \in \mathcal{C},\ c(z) \neq 0\}$$

The relationship between free distance and error detection and correction capacity is the same as with linear block codes.

Theorem 2.3.1. *If $\mathcal{C}$ is a code with free distance d then the code can always detect up to $d-1$ errors and correct up to $\lfloor \frac{d-1}{2} \rfloor$ errors.*

The proof of this is the same as for linear block codes, as it only depends on the distance one is taking the minimum of being a metric.

Another way to measure distance of codewords exists, the so called column distance. The column distance has relevance to erasure channels, where the location of errors is known, as opposed to general errors where the errors can happen in any of the symbols of the vector. The column distance

is characterized by the generator matrices of codes, but in the case of non-catastrophic codes it can also be characterized by parity check matrices. Therefore, it is assumed all codes are noncatastrophic from hereon.

Definition 2.3.3 (j-truncation)**.** For $c(z) \in R^n$ with $\deg(c(z)) = \gamma$, writing out $c(z) = c_0 + c_1 z + \cdots c_\gamma z^\gamma$. Define the j-truncation of $c(z)$ as $c_{[0,j]}(z) = c_0 + c_1 z + \cdots + c_j z^j$.

Definition 2.3.4 (j-column distance)**.** The j-column distance is defined as

$$d_j^c(\mathcal{C}) = \min_{c(z)\in\mathcal{C}}\{\mathrm{wt}(c_{[0,j]}(z) \mid c_0 \neq 0\}$$

Let $G(z) = \sum_{i=1}^{\mu}$ and $H(z) = \sum_{i=1}^{\nu}$ be a generator matrix and parity check matrix respectively, of a convolutional code $\mathcal{C}$. For $j \in \mathbb{N}$ the truncated sliding generator and parity check matrices are defined as

$$G_j^c = \begin{bmatrix} G_0 & \cdots & G_L \\ & \ddots & \vdots \\ 0 & & G_0 \end{bmatrix} \quad \text{and} \quad H_j^c = \begin{bmatrix} H_0 & & 0 \\ \vdots & \ddots & \\ H_L & \cdots & H_0 \end{bmatrix},$$

respectively. They are named generator matrix and parity check matrix because for any $c(z) = \sum_{i\in\mathbb{N}} c_i z^i \in \mathcal{C}$

$$[c_0 \cdots c_j] = [u_0 \cdots u_j] G_j^c$$

for some $u_0, \ldots, u_j \in \mathbb{F}_q^k$. and

$$H[c_1 \cdots c_j] = 0$$

Since $G(z)$ is left prime, G_0 has full row rank and therefore $c_0 \neq 0$ if and only if $u_0 \neq 0$. Therefore

$$d_j^c(\mathcal{C}) = \min_{u_0\neq 0}\{\mathrm{wt}([u_0 \cdots u_j]G_j^c)\} = \min_{c_0\neq 0}\{\mathrm{wt}(c_{[0,j]}(z)) \mid H_j^c[c_0 \cdots c_j]^T = 0\}$$

From this follows the following theorem.

Theorem 2.3.2. *For $d \in \mathbb{N}$ the following statements are equivalent*

(a) $d_j^c(\mathcal{C}) = d$

(b) None of the first n columns of H_j^c are contained in the span of any other $d-2$ columns and one of the first n columns of H_j^c is in the span of some other $d-1$ columns of this matrix.

There exists upper bounds for distances of convolutional codes, similar to how it works for block codes.

Theorem 2.3.3 (Singleton Bound)**.** *Let $\mathcal{C}$ be an $(n, k, \delta)_q$ convolutional code. Then,*

$$d_{free} \leq (n-k)\left(\left\lfloor \frac{\delta}{k} \right\rfloor + 1\right) + \delta + 1$$

To prove the theorem, the following lemma is needed. Denote by G_∞ the matrix consisting of the high order coefficients of the corresponding entries of $G(z)$.

Lemma 2.3.1. *Let $\mathcal{C}$ be an $(n, k, \delta)_q$ convolutional code. Let $G(z)$ be a generator matrix in with ordered row degrees $\nu_1 \geq \nu_2 \geq \cdots \geq \nu_k$ such that $\operatorname{rank} G_\infty = k$. If this is the case then $\delta = \sum_{i=1}^{k} \nu_i$. Let ℓ be the number of rows with $\nu_i = \nu_k$. Then the following upper bound holds.*

$$d_{free} \leq n(\nu_k + 1) - \ell + 1$$

Proof. By applying column permutations and row addition and scaling, it is possible to change $G(z)$ so that the last ℓ rows of G_∞ form a matrix

$$\begin{bmatrix} I_\ell & M \end{bmatrix}$$

where M is an $\ell \times (n-\ell)$ matrix over $\mathbb{F}_q$. This change can be represented by an invertible matrix acting on the last ℓ rows, that also doesn't change the row degrees $\nu_1, \ldots, \nu_k$.

The last row of the new $G(z)$ will have $\ell - 1$ polynomials with weight, strictly less than ν_k+1, one with weight no less than ν_k+1, and the remaining $n-\ell$ have weight less than or equal to ν_k+1. Therefore the word $(0, 0, \ldots, 1)$ gives a codeword with weight less than or equal to

$$(\ell-1)\nu_k + (\nu_k+1) + (n-\ell)(\nu_k+1) = n(\nu_k+1) - \ell + 1$$

□

It is now possible to prove the singleton bound by maximizing the above bound.

Proof. The upper bound from the previous lemma is the largest if ν_k is maximized and ℓ minimized. The largest possible value for ν_k is $\lfloor \delta/k \rfloor$. Minimizing ℓ results in

$$\nu_1 = \lfloor \delta/k \rfloor + 1, \ldots, \nu_{k-\ell} = \lfloor \delta/k \rfloor + 1, \nu_{k-\ell+1} = \lfloor \delta/k \rfloor, \ldots, \nu_k = \lfloor \delta/k \rfloor$$

substituting $\nu_k = \lfloor \delta/k \rfloor$ and $\ell = k - \delta + k\lfloor \delta/k \rfloor$ into the bound gives the desired result. □

Theorem 2.3.4 (Column Distance Bound)**.** *Let $\mathcal{C}$ be an $(n, k, \delta)_q$ convolutional code. Then,*

$$d_j^c(\mathcal{C}) \leq (n-k)(j+1)+1 \text{ for all } j \in \mathbb{N}$$

Proof. The matrix H_j^c has full row rank, so the $(n-k)(j+1)$ rows are linearly independent. Therefore by Theorem 2.3.2, $d_j^c(\mathcal{C})$ must be less than or equal to $(n-k)(j+1)+1$. □

The first these two bounds is called the generalized singleton bound, and any code that reaches this bound is called Maximum Distance Separable (MDS)

Definition 2.3.5 (MDS Code)**.** An $(n, k, \delta)_q$ convolutional code that satisfies

$$d_{\text{free}} = (n-k)\left(\left\lfloor \frac{\delta}{k} \right\rfloor + 1\right) + \delta + 1$$

is called a maximum distance separable (MDS) code.

A code that reaches the second bound is called Maximum Distance Profile (MDP). A lemma is needed to define it more conveniently.

Lemma 2.3.2. *Let $\mathcal{C}$ be an $(n, k, \delta)_q$ convolutional code. If $d_j^c(\mathcal{C}) = (n-k)(j+1)+1$ for some $j \in [L]$, then $d_i^c(\mathcal{C}) = (n-k)(i+1)+1$ for all $i \leq j$.*

Note that $0 \leq d_0^c(\mathcal{C}) \leq d_1^c(\mathcal{C}) \leq \cdots \leq d_{\text{free}}$, so $d_j^c(\mathcal{C}) \leq (n-k)(\lfloor \frac{\delta}{k} \rfloor + 1) + \delta + 1$ for all $j \in \mathbb{N}$. Because of this guarantee, it is sufficient to establish the column distance bound for all j up to $L = \left\lfloor \frac{\delta}{k} \right\rfloor + \left\lfloor \frac{\delta}{n-k} \right\rfloor$, since inserting this value into the second bound turns the equation into that of the generalized singleton bound, which $d_j^c(\mathcal{C})$ is known to satisfy.

Definition 2.3.6 (MDP)**.** An $(n, k, \delta)_q$ convolutional code that satisfies

$$d_j^c(\mathcal{C}) = (n-k)(j+1)+1 \text{ for } j = 0, \ldots, L = \left\lfloor \frac{\delta}{k} \right\rfloor + \left\lfloor \frac{\delta}{n-k} \right\rfloor$$

is called a maximum distance profile (MDP) code.

Definition 2.3.7 (sMDS)**.** An $(n, k, \delta)_q$ convolutional code is called sMDS if

$$d_M^c(\mathcal{C}) = (n-k)\left(\left\lfloor \frac{\delta}{k} \right\rfloor + 1\right) + \delta + 1, \text{ where } M = \left\lfloor \frac{\delta}{k} \right\rfloor + \left\lceil \frac{\delta}{n-k} \right\rceil$$

Theorem 2.3.5 (MDP Criteria)**.** *Let $\mathcal{C}$ be a convolutional code with generator matrix $G(z) = \sum_{i=1}^{\mu} G_i z^i \in \mathbb{F}_q[z]^{k\times n}$ and with left prime parity check matrix $H(z) = \sum_{i=1}^{\nu} H_i z^i \in \mathbb{F}_q[z]^{(n-k)\times n}$. The following are equivalent*

1. *$\mathcal{C}$ is MDP.*
2. $\mathcal{G}_L = \begin{bmatrix} G_0 & \cdots & G_L \\ & \ddots & \vdots \\ 0 & & G_0 \end{bmatrix}$ *where $G_i = 0$ for $i > \mu$ has the property that every full size minor that is formed by rows with indices $1 \le j_1 < \cdots < j_{(L+1)k} < (L+1)n$ which fulfills $j_{sk} < sn$ for $s = 1, \ldots, L$ is nonzero.*
3. $\mathcal{H}_L = \begin{bmatrix} H_0 & & 0 \\ \vdots & \ddots & \\ H_L & \cdots & H_0 \end{bmatrix}$ *where $H_i = 0$ for $i > \nu$ has the property that every full size minor that is formed by columns with indices $1 \le j_1 < \cdots < j_{(L+1)(n-k)} < (L+1)n$ which fulfills $j_{s(n-k)} < sn$ for $s = 1, \ldots, L$ is nonzero.*

Theorem 2.3.6 (The dual of MDP is MDP)**.** *An $(n, k, \delta)_q$ convolutional code is MDP if and only if its dual, an $(n-k, k, \delta)_q$ convolutional code, is MDP.*

Definition 2.3.8 (Partial Parity Check Matrix)**.** Let $H(z) = \sum_{i=0}^{\nu} H_i z^i$ be a left prime row reduced parity check matrix of an $(n, k, \delta)_q$ convolutional code $\mathcal{C}$ with $H_\nu \neq 0$.

Let $L = \left\lfloor \frac{\delta}{k} \right\rfloor + \left\lfloor \frac{\delta}{n-k} \right\rfloor$. Then,

$$\mathfrak{H} = \begin{bmatrix} H_\nu & \cdots & H_0 & & 0 \\ & \ddots & & \ddots & \\ 0 & & H_\nu & \cdots & H_0 \end{bmatrix} \in \mathbb{F}_q^{(L+1)(n-k)\times(\nu+L+1)n}$$

is called a partial parity check matrix of $\mathcal{C}$. In addition, if every full sized minor of $\mathfrak{H}$ that is formed by columns $j_1, \ldots, j_{(L+1)(n-k)}$ with $j_{(n-k)s+1} > sn$ and $j_{(n-k)s} \le sn + \nu n$ for $s = 1, \ldots, L$ is nonzero, then the code $\mathcal{C}$ is called a complete MDP convolutional code.

Chapter 3

Practical Results of Convolutional Codes

3.1 Construction of MDS Codes

A number of constructions of MDS codes will now be presented. They each have advantages and disadvantages in terms of the allowed parameters and field sizes for the constructions, and therefore they all have their uses.

This section and the next, relies on [8] for for collecting the results in one place. The theorems in this section comes, in order, from [6], [5] and [12].

The first two constructions are for codes with $k = 1$.

Theorem 3.1.1. *For $n \geq 2$ and $q \geq n+1$, set $s_j = \lceil (j-1)(q-1)/n \rceil$ for $j = 2, \ldots, n$ and $\delta = \begin{cases} \lfloor \frac{2}{9}q \rfloor, & n = 2 \\ \lfloor \frac{1}{3}q \rfloor, & 3 \leq n \leq 5 \\ \lfloor \frac{1}{2}q \rfloor, & n \geq 6 \end{cases}$*
In addition, let α be a primitive element of $\mathbb{F}_q$. Set $g_1(x) = \prod_{k=1}^{\delta}(x - \alpha^k)$, $g_j(x) = g_1(x\alpha^{-sj})$. Then $G(z) = [g_1(z), g_2(z), \ldots, g_n(z)]$ is a generator matrix for an $(n, 1, \delta)_q$ MDS convolutional code with free distance $n(\delta + 1)$.

The second construction, like the first, gives a code with $k = 1$ and the same restriction on q. However, it further restricts δ.

Theorem 3.1.2. *Let $q \geq n+1$, $0 \leq \delta \leq n-1$ and α an element of $\mathbb{F}_q$ with order at least n. Then $G(z) = \sum_{i=0}^{\delta} z^i [1, \alpha, \alpha^2, \ldots, \alpha^{(n-1)i}]$ generates an $(n, 1, \delta)_q$ MDS convolutional code.*

And the last construction allows for wider selection of values for n, k and δ, but with a restricted field size.

Theorem 3.1.3. *Let a, r be integers such that $a \geq \lfloor \frac{\delta}{k} \rfloor + 1 + \frac{\delta}{n-k}$ and $an = p^r - 1$ for prime p. Let α be a primitive element of $\mathbb{F}_{p^r}$. Set $N = an$, $K = N - (n-k)(\lfloor \frac{\delta}{k} \rfloor + 1) - \delta$, $g(z) = (z - \alpha^0)(z - \alpha^1) \cdots (z - \alpha^{N-K-1})$ and write $g(z)$ as $g(z) = g_0(z^n) + g_1(z^n)z + \cdots + g_{n-1}(z^n)z^{n-1}$.*

Then

$$G(z) = \begin{bmatrix} g_0(z) & g_1(z) & \cdots & g_{n-1}(z) \\ g_{n-1}(z)z & g_0(z) & \cdots & g_{n-2}(z) \\ \vdots & \ddots & & \vdots \\ g_{n-k}(z)z & \cdots & \cdots & g_{n-k}(z) \end{bmatrix}$$

is the generator matrix of an $(n, k, \delta)_q$ MDS convolutional code.

3.2 Construction of MDP Codes

The constructions for MDP codes depend on finding superregular matrices, and appropriately taking a subset of the rows and columns to form a parity check matrix.

This section uses [8] for the collection of results. The theorems in order of apearance comes from [13], [4] and [2]

Since superregular matrices are used to construct MDP codes it is necessary to define what a superregular matrix is.

Definition 3.2.1 (Superregular matrix)**.** Superregular matrices are defined as follows:

1. Let $l \in \mathbb{N}$ and $A = [a_{ij}]$ be a matrix in $\mathbb{F}_q^{l \times l}$. Define $\overline{a} = [\overline{a}_{ij}]$ where $\overline{a}_{ij} = 0$ if $a_{ij} = 0$ and $\overline{a}_{ij}$ equals the variable x_{ij} otherwise. Consider the determinant of $\overline{A}$ in terms of the ring of polynomials in the variables x_{ij} and coefficients in $\mathbb{F}_q$. The determinant of $\overline{A}$ is considered trivially zero if the determinant of $\overline{A}$ equals the zero polynomial. A is called superregular if all its not trivially zero minors are nonzero.

2. A Toeplitz matrix of the form $\begin{bmatrix} a_1 & & 0 \\ \vdots & \ddots & \\ a_l & \cdots & a_l \end{bmatrix}$ with $a_i \in \mathbb{F}_q$ for $i = 1, \ldots, l$ that is superregular is called a lower triangular superregular matrix.

Using any lower triangular superregular matrix it is then possible to construct a convolutional MDP code with $(n-k)|\delta$ and $k > \delta$.

Theorem 3.2.1. *Let $(n-k)|\delta$, $k > \delta$ and T be an $r \times r$ lower triangular superregular matrix with $r = (L+1)(2n-k-1)$. For $j = 0, \ldots L$ let I_j and J_j be the following sets*

$$I_j = \{(j+1)n + j(n-k-1), \ldots, (j+1)(2n-k-1)\}$$
$$J_j = \{jn + j(n-k-1) + 1, \ldots, (j+1)n + j(n-k-1)\}$$

and let $I = \cup_{j=0}^{L} I_j$ and $J = \cup_{j=0}^{L} J_j$. Form $\mathcal{H}_L =$] [taking the rows and columns of T with indices in I and J respectively. Then $H(z) = \sum_{i=0}^{L} H_i z^i$ is the parity check matrix of an MDP convolutional code.

To construct a lower triangular superregular matrix T for the above theorem, one can use the following theorem.

Theorem 3.2.2. *For every $b \in \mathbb{N}$ the not trivially zero minors of the Toeplitz matrix*

$$\begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ \binom{b}{1} & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \binom{b}{b-1} & & \ddots & \ddots & 0 \\ 1 & \binom{b}{b-1} & \cdots & \binom{b}{1} & 1 \end{bmatrix}$$

are all positive. Therefore, for each $b \in \mathbb{N}$ there exists a smallest prime p_b such that this matrix is superregular over the field $\mathbb{F}_{p_b}$.

Another construction works with a superregular matrix that isn't specifically lower triangular. The matrix is constructed using the following theorem.

Theorem 3.2.3. *Let n, k, δ be given, and $m = \max\{n-k, k\}$. Let α be a primitive element of $\mathbb{F}_{p^N}$ where p is prime and N an integer. Define*

$$T_i = \begin{bmatrix} \alpha^{2^{im}} & \alpha^{2^{im+1}} & \cdots & \alpha^{2^{(i+1)m-1}} \\ \alpha^{2^{im+1}} & \alpha^{2^{im+2}} & & \alpha^{2^{(i+1)m}} \\ \vdots & & \ddots & \vdots \\ \alpha^{2^{(i+1)m-1}} & \alpha^{2^{(i+1)m}} & \cdots & \alpha^{2^{(i+2)m-2}} \end{bmatrix}$$

for $i = 1, \ldots, L = \left\lfloor \frac{\delta}{k} \right\rfloor + \left\lfloor \frac{\delta}{n-k} \right\rfloor$. Define

$$\mathcal{T}(T_0, \ldots, T_L) = \begin{bmatrix} T_0 & & 0 \\ \vdots & \ddots & \\ T_L & \cdots & T_0 \end{bmatrix}$$

If $N \geq 2^{m(L+2)-1}$ then the matrix $\mathcal{T}(T_0, \ldots, T_L)$ is superregular over $\mathbb{F}_{p^N}$.

Using the constructed matrix $\mathcal{T}(T_0, \ldots, T_L)$, it is possible to create an MDP convolutional code according to the following theorem.

Theorem 3.2.4. *Let n, k, δ be integers such that $(n-k)|\delta$ and let $T_l = [t_{ij}^l]$ for $i, j = 1, 2, \ldots, m$ and $l = 0, \ldots, L$ where T_l is a matrix like in the previous theorem. Define the matrix $\overline{H}_l = [t_{ij}^l]$ for $i = 1, \ldots, n-k$, $j = 1, \ldots, k$ and $l = 0, 1, \ldots, L$.*

If $q = p^N$ for $N \in \mathbb{N}^+$ and $q \geq p^{2m(L+1)+n-1}$, then the convolutional code $\mathcal{C} = \ker_R[A(z)B(z)]$ where $A(z) = \sum_{i=0}^{\nu} A_i z^i \in R^{(n-k)\times(n-k)}$ and $B(z) =$

$\sum_{i=0}^{\nu} B_i z^i \in R^{(n-k)\times k}$, *with* $\nu = \frac{\delta}{n-k}, A_0 = I_{n-k}$. *Where* $A(z)$ *and* $B(z)$ *are the matrices obtained by solving the equations*

$$\begin{bmatrix} A_\nu & \cdots & A_1 \end{bmatrix} \begin{bmatrix} \overline{H}_{l-\nu} & \cdots & \overline{H}_1 \\ \vdots & & \vdots \\ \overline{H}_{L-1} & \cdots & \overline{H}_\nu \end{bmatrix} = - \begin{bmatrix} \overline{H}_L & \cdots & \overline{H}_{\nu+1} \end{bmatrix},$$

and

$$B_i = A_0\overline{H}_i + A_1\overline{H}_{i-1} + \cdots + A_i\overline{H}_0, \ i = 0, \ldots, \nu,$$

is an $(n, k, \delta)_q$ *MDP convolutional code.*

3.3 Decoding Convolutional Codes

This section will be about erasure decoding of convolutional codes. It is possible to do conventional error correction on convolutional codes, but erasure decoding without dependence on the time index is where they are particualarly useful.

This section references the sources [8] and [13].

Convolutional codes can be seen as a generalization to linear block codes, where $\delta = 0$, i.e. the polynomials are all of degree 0. The theory for decoding convolutional codes can therefore also be seen as a generalization of the decoding theory for linear block codes.

It is well known that when decoding erasures on linear block codes it is possible to decode up to $d - 1$ erasures, where d is the minimum hamming distance. In particular, d can be decided by the fact that it equals the minimum possible number of columns in a parity check matrix of the code that are linearly dependent.

If $c^{(e)}$ is the components of c that are erased, and $c^{(r)}$ is the components that are recieved. Then, letting $H_0^{(e)}$ be the columns of the parity check matrix corresponding to the indices erased from c and $H_0^{(r)}$ similarly for the received indices, membership of the code decided by $H_0 c^T = 0$ can be rewritten as $H_0^{(e)} c^{(e)} = -H_0^{(r)} c^{(r)}$. Erasure decoding is then solving this set of linear equations.

When erasure correcting an $(n, k, \delta)_q$ noncatastrophic convolutional code $\mathcal{C}$, the performance is better than for an equivalent linear block code due to being able to limit decoding to particular windows of the sequence being decoded.

Writing a left prime parity check matrix as $H(z) = \sum_{i=0}^{\nu} H_i z^i$ and a codeword as $c(z) = \sum_{i\in\mathbb{N}} c_i z^i$, assume that the coefficients $c_0, \ldots, c_{t-1}$ are known and there is at least one erasure in c_t. Then for each $j \in \mathbb{N}$, the

matrix

$$\mathfrak{H}_j = \begin{bmatrix} H_\nu & \cdots & H_0 & & 0 \\ & \ddots & & \ddots & \\ 0 & & H_\nu & \cdots & H_0 \end{bmatrix} \in \mathbb{F}_q^{(j+1)(n-k)\times(\nu+j+1)n}$$

satisfies $\mathfrak{H}_j[c_{t-\nu}\cdots c_{t+j}]^T = 0$, where $c_i = 0$ for all $i \notin \{0,\ldots,\deg c\}$. Let $\mathfrak{H}_j^{(1)}$ denote the matrix consisting of the first νn columns of $\mathfrak{H}_j$. Then it is the case that $\mathfrak{H}_j = [\mathfrak{H}_j^{(1)} H_j^c]$, and so to correct erasures one has to work in the window $[c_{t-\nu}\cdots c_{t+j}]$ and solve the system of equations

$$H_j^{(e)}[c_t^{(e)}\cdots c_{t+j}^{(e)}]^T = -H_j^{(r)}[c_t^{(r)}\cdots c_{t+j}^{(r)}]^T - \mathfrak{H}_j[c_{t-\nu}\cdots c_{t-1}]$$

where $c_i^{(e)}$ and $c_i^{(r)}$ denote the erased and received components of c_i respectively, and $H_j^{(e)}$ and $H_j^{(r)}$ are H_j^c with the corresponding columns erased. Unambiguous erasure correction is possible if and only if the system of equations has a unique solution, that is if $H_j^{(e)}$ has full column rank.

Therefore you have the following theorem:

Theorem 3.3.1 (Erasure Capacity of Convolutional Code)**.** *Let $\mathcal{C}$ be an $(n,k,\delta)_q$ convolutional code. If there is a sliding window of length $(j+1)n$ and no more than $d_j^c(\mathcal{C})-1$ erasures occur, then full left to right error correction is possible.*

Proof. □

The best capacity is obviously achieved when $d_j^c(\mathcal{C})$ reaches the column bound, i.e. when the code is MDP.

If a sequence of symbols with erasures that can't be corrected is encountered, then it is necessary to find a block of νn correct symbols to restart decoding, a so called guard space, that precedes symbols decodable according to Theorem 3.3.1.

Theorem 3.3.2 (Guard Space Theorem)**.** *Let $\mathcal{C}$ be an $(n,k,\delta)_q$ complete MDP convolutional code and $L = \lfloor \frac{\delta}{n} \rfloor + \lfloor \frac{\delta}{n-k} \rfloor$. If in a window size of $(L+\nu+1)n$ there are no more than $(L+1)(n-k)$ erasures, and they are distributed such that between position 1 and sn and between $(L+\nu+1)n$ and $(L+\nu+1)n - sn + 1$ there are no more than $s(n-k)$ erasures, then full correction of these symbols are possible In particular a guard space can be computed.*

3.4 Input-state-output Representation

This section is based on [8].

There is a connection between convolutional codes and discrete-time linear systems. Take linear systems of equations of the form

$$\begin{aligned} x_{t+1} &= x_t A + u_t B \\ y_t &= x_t C + u_t D \\ c_t &= [y_t u_t] \end{aligned} \tag{3.1}$$

with $(A, B, C, D) \in \mathbb{F}_q^{s\times s} \times \mathbb{F}_q^{k\times s} \times \mathbb{F}_q^{s\times n-k} \times \mathbb{F}_q^{k\times n-k}$, $t \in \mathbb{N}$, $s, k, n \in \mathbb{N}^+$ with $n > k$ and the information vector $u_t \in \mathbb{F}_q^k$, the state vector $x_t \in \mathbb{F}_q^s$, and the parity vector $y_t \in \mathbb{F}_q^{n-k}$. The system of equations can be represented by the tuple $\Sigma = (A, B, C, D)$, given that the remaining structure is implicit in the setup. The state of the system over time is then represented by the parameter sequences called the trajectory $\{u_t\}_{t\in\mathbb{N}}, \{x_t\}_{t\in\mathbb{N}}, \{y_t\}$. These parameters represent the input (u_t), the state (x_t), and the output (y_t) of the system. If the initial state is set to $x_0 = 0$, the trajectory of the system can be represented by power series

$$\begin{aligned} u(z) &= \sum_{t\in\mathbb{N}} u_t z^t \\ x(z) &= \sum_{t\in\mathbb{N}} x_t z^t \\ y(z) &= \sum_{t\in\mathbb{N}} y_t z^t \end{aligned}$$

These satisfy the equations (3.1) if and only if

$$\begin{bmatrix} x(z) & u(z) & y(z) \end{bmatrix} E(z) = 0$$

where

$$E(z) = \begin{bmatrix} I_s - Az & -C \\ -Bz & -D \\ 0 & I_{n-k} \end{bmatrix}$$

In order to get a convolutional code from a linear system like (3.1), one only needs the polynomial trajectories $c(z) = [u(z)y(z)]$. The $x(z)$ trajectories with infinite support are discarded since they correspond to an infinite transmission, and therefore the power series are the special case of polynomials. The set of finite support trajectories of the equations form a submodule of $\mathbb{F}_q^n$ with rank k and is therefore an $(n, k)_q$ convolutional code denoted by $\mathcal{C}(A, B, C, D)$. The tuple (A, B, C, D) is then called the input-state-output (ISO) representation of the code. That the trajectories form a convolutional code implies the existence of a generator matrix. Furthermore, $[X(z)G(z)]E = 0$ and $[X(z)G(z)]$ left prime, for $X(z) \in \mathbb{F}_q^{k\times s}$ and $G(z) \in \mathbb{F}_q^{k\times n}$ if and only if $G(z)$ is a generator matrix for the code.

In contrast, for an $(n,k)_q$ convolutional code $\mathcal{C}$, there exists a tuple (A,B,C,D) such that $\mathcal{C} = \mathcal{C}(A,B,C,D)$ up to permutation by the coordinates.

3.5 Viterbi Decoding

Important to convolutional codes is a way to decode them, after all there is no point to encoding information if there is no way to read it back. The Viterbi algorithm is a minimum distance encoder using the convolutional code generalization of hamming distance as the metric. A useful trait of the Viterbi algorithm is that it allows for the decoding of words in small sections at a time, which can be particularly useful in low bandwidth applications.

This section is based on material from [7].

A helpful way to illustrate the Viterbi algorithm is with the aid of trellis diagrams.

The trellis diagram is a graph with each node encoding a state x_t and a time index. The nodes are arranged in columns from left to right corresponding to the time index and rows according to state. So the vertical position of a node denotes the value of the state variable, and the horizontal position corresponds to the time index. Directed edges going from left to right represent possible state transitions, with each edge labeled with the input/output values at the time index of the tail.

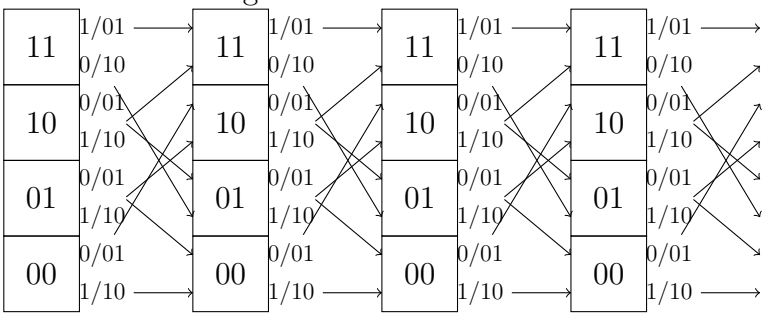
Example 3.5.1. To illustrate the trellis diagrams an example is needed. Take the following system, a good sample code found in [7]. Let $\mathcal{C}$ be an $(2,1)_2$ code, with codewords $y(i) = (y_1(i), y_2(i))$ for all time indices i, where

$$\begin{aligned} y_1(i) &= u(i) + u(i-2) \\ y_2(i) &= u(i) + u(i-1) + u(i-2) \end{aligned}$$

The previous inputs $u(i-1)$ and $u(i-2)$ are encoded in the state, and thus it can be put in input-state-output representation with the matrices

$$\begin{aligned} A &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \\ B &= \begin{bmatrix} 1 & 0 \end{bmatrix} \\ C &= \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \\ D &= \begin{bmatrix} 1 & 1 \end{bmatrix} \end{aligned}$$

The trellis diagram of this code can then be written.



The trellis diagram of a convolutional code is thus used to describe the Viterbi algorithm

Codewords are paths through the trellis diagram of the convolutional code that starts and ends at zero. When a codeword is encoded the transmitted data takes the form $c(z) = u(z)G(z)$. The goal of the decoding process is to find $u(z)$ given a received word $r(z)$ which is $x(z)$ with errors added, i.e. find the path through the trellis diagram corresponding to the most likely codeword. To that end, costs are assigned to all edges of the trellis diagram equal to the distance between the output at that edge and the actual value in the recieved word. The minimum distance decoding is then the minimum cost path through the trellis diagram. Dijkstra's algorithm would be suitable for the case of a general graph, but due to the additional structure given by the trellis, it is possible to simplify the algorithm.

To that end, the following lemma is used.

Lemma 3.5.1 (Viterbi Lemma)**.** *Assuming, for some $i \geq 0$, all the minimal cost paths $\mathcal{P}_x$ from $(0,0)$ to the vertices (x,i) for all $x \in \mathbb{F}_q^s$, are known and have weight $w(x)$. Let $e_1, \ldots, e_r$ be all the edges that end at a given vertex $(x, i+1)$. Let (x_j, i) be the starting vertex of the edge e_j and let w_j be the weight of the edge e_j. Then the minimal cost of a path from $(0,0)$ to $(x, i+1)$ is the minimum w of the sums $w(x_j) + w_j$ and $\mathcal{P}_{x_j} e_j$ is also a minimum cost path if j is chosen such that $w = w(x_j) + w_j$.*

Proof. The construction gives a path of the required cost, so it needs showing that no lower cost can be achieved Assume $\mathcal{P}$ is a path from $(0,0)$ to $(x, i+1)$ satisfying $w(\mathcal{P}) \leq w$. A path ending in $(x, i+1)$ must end in one of the edges e_j, so let e_{j_0} be that edge for $\mathcal{P}$. Then the path can be written $\mathcal{P} = \mathcal{P}' e_{j_0}$ where $\mathcal{P}'$ is the path containing all the edges before e_{j_0}. By assumption $w(\mathcal{P}') \geq w(x_{j_0})$, and therefore $w(\mathcal{P}) = w(\mathcal{P}') + w_{j_0} > w$ which is a contradiction. □

The Viterbi algorithm can be summed up in the following steps.

1. Initialize the variables

$$
\begin{aligned}
&i = 0 \\
&w(0,0) = 0 \\
&w(x,i) = \infty, \text{for all } (x,i) \in \mathbb{F}_q^s \times \mathbb{N}, (x,i) \neq (0,0) \\
&\mathcal{P}(0,0) = \emptyset
\end{aligned}
$$

2. For all states $x \in \mathbb{F}_q^s$ with $w(x,i) < \infty$, iterate over all edges e leaving (x,i) with x' as the end state of e, and do the following: If $w(x,i) + w(e) < w(x', i+1)$, assign the values

$$
\begin{aligned}
\mathcal{P}(x', i+1) &= \mathcal{P}(x,i)e \\
w(x', i+1) &= w(x,i) + w(e)
\end{aligned}
$$

3. $i = i + 1$

4. If $i < N$, go to step 2

5. Output the path $\mathcal{P}(0, N)$

As with Dijkstra's algorithm all nodes are assigned a minimal path cost, the initial node $(0,0)$ being assigned cost 0, the others having infinite cost since a path to them is not yet known. From the set of unvisited nodes a current node is chosen, where all edges connecting from it are traced, and if the sum of the minimal path cost of the current node and the cost of the edge is less than the current minimal path cost of the connecting node that node is updated with the lower value. Unlike Dijkstra's algorithm, where there is no prefered order to the nodes and it therefore becomes necessary to sort the unvisited nodes, in a trellis diagram there is a partial order on the time index. It therefore makes sense to visit the nodes in the order of their time index, low to high.

Bibliography

[1] Gianira Nicoletta Alfarano. *Algebraic, Combinatorial and Geometric Aspects of Some Error-Correcting Codes*. Phd thesis, Mathematisch-naturwissenschaftlichen Fakultät der Universität Zürich, 2022. Available at `https://user.math.uzh.ch/alfarano/PhdThesis.pdf`.

[2] P Almeida, Diego Napp, and Raquel Pinto. A new class of superregular matrices and mdp convolutional codes. *Linear Algebra and its Applications*, 439(7):2145–2157, 2013.

[3] F. R. Gantmacher. *The Theory of Matrices*. Chelsea Publishing Company, 1959.

[4] H. Gluesing-Luerssen, J. Rosenthal, and R. Smarandache. Strongly-mds convolutional codes. *IEEE Transactions on Information Theory*, 52(2): 584–598, 2006. doi: 10.1109/TIT.2005.862100.

[5] Heide Gluesing-Luerssen and Barbara Langfeld. A class of one-dimensional mds convolutional codes. *Journal of Algebra and its Applications*, 5(04):505–520, 2006.

[6] Jørn Justesen. An algebraic construction of rate 1/v-ary codes; algebraic construction (corresp.). *IEEE Transactions on Information Theory*, 21 (5):577–580, 1975.

[7] Jyrki Lahtonen. Convolutional codes. `https://www.karlin.mff.cuni.cz/~holub/soubory/ConvolutionalCodesJyrkiLahtonen.pdf`, March 1998. [Online; accessed 06-December-2024].

[8] Julia Lieb, Raquel Pinto, and Joachim Rosenthal. Convolutional codes, 2020. URL `https://arxiv.org/abs/2001.08281`.

[9] Dylan Poulsen. Modules: An introduction. `http://buzzard.ups.edu/courses/2010spring/projects/poulsen-modules-ups-434-2010.pdf`, April 2010. [Online; accessed 12-November-2024].

[10] Joachim Rosenthal and Roxana Smarandache. Maximum distance separable convolutional codes. *Applicable Algebra in Engineering, Communication and Computing*, 10(1), 1999.

[11] Parvati Shastri. Lectures on modules over principal ideal domains. `https://www.imsc.res.in/~knr/14mayafs/Notes/ps.pdf`, 2014.

[12] Roxana Smarandache, Heide Gluesing-Luerssen, and Joachim Rosenthal. Constructions of mds-convolutional codes. *IEEE Transactions on Information Theory*, 47(5):2045–2049, 2001.

[13] Virtudes Tomás, Joachim Rosenthal, and Roxana Smarandache. Decoding of convolutional codes over the erasure channel. *IEEE Transactions on Information Theory*, 58(1):90–108, 2012.