Searching for Minimum Storage Regenerating Codes

Daniel Cullina
Professor Tracey Ho
Department of Electrical Engineering
California Institute of Technology

Acknowledgments

I would like to thank my thesis advisor, Professor Tracey Ho for introducing me to this topic and guiding me throughout the year. I would also like to thank Dr. Georgios Dimakis for helping me to understand his results regarding the problem and for sending me in fruitful directions. Special thanks to Sherwin Doroudi for his help during a variety of discussions throughout the year, to Mason Smith for his help with linear algebra, and to Shengbo Xu for his help during the writing process.

Abstract

Regenerating codes allow distributed storage systems to recover from the loss of a storage node while transmitting the minimum possible amount of data across the network. We search for examples of Minimum Storage Regenerating Codes. To exhaustively search the space of potential codes, we reduce the potential search space in several ways. We impose an additional symmetry condition on codes that we consider. We specify codes in a simple alternative way, using additional recovered coefficients rather than transmission coefficients. We place codes into equivalence classes to avoid redundant checking. We find MSR codes for the parameters $n = 5$ and $k = 3$ in various fields. We demonstrate that it is possible for such codes to be composed of vectors in general position.

Contents

1 Introduction

Distributed storage systems allow data to be stored with much higher reliability than that of the individual components. They do this by recovering from the failure of individual components without the loss of data. When a node is lost, the network must provide a replacement node with the data that was stored on the original node. There are a variety of storage and recovery schemes that can be used to implement distributed storage. Data replication is the simplest such scheme. When a node is lost, nodes elsewhere in the network can transmit their copies of the lost data to a replacement node. The amount of data transmitted is equal to the size of the lost node.

Erasure codes can match the level of reliability achieved by replication while improving on the amount of storage space used. In an Maximum Distance Separable (MDS) code, any k nodes contain enough information to recreate all of the source data. When a node is lost, the simplest procedure to is to transmit the contents of k nodes to the replacement node. The replacement node can rebuild all of the source data and use it to construct the contents of the lost node. An amount of data equal to the size of the source must be transmitted across the network during this procedure.

However, there are codes that can recover a lost node while transmitting less data than this [Dimakis et al., 2007]. Regenerating codes require storage nodes to transmit a linear combination of different parts of their contents to a replacement node during the recovery process. This allows the information theoretic lower bound of network bandwidth to be achieved. Minimum Storage Regenerating (MSR) codes are MDS codes that use the minimal network repair bandwidth.

We are interested in finding examples of MSR codes. To exhaustively search the space of potential codes in feasible amounts of time, we reduce the search space in several ways. We impose an additional condition that restrict the type of codes that we consider. This allows us to consider only highly symmetric codes that can be more concisely specified. We specify a code in a simple alternative way, using additional recovered coefficients rather than transmission coefficients. The space of codes can be searched more easily and efficiently when codes are specified this way. Finally, we use linear transformations to relate codes to each other and place them into equivalence classes. This allows us to check only one code from each equivalence class

2 Definitions and Notation

The storage networks that we are concerned with contain n equivalent storage nodes. We wish to store $\mathcal{M}$ bits of data in the network, where $\mathcal{M}$ is k times the size of one of the storage nodes. Because of this, we say that the network has k source nodes.

2.1 Lower bound on recovery bandwidth

During the recovery process, $\frac{\mathcal{M}d}{k(d-k+1)}$ bits of data must be transmitted, where d is the number of nodes providing data [Dimakis et al., 2007]. We are interested in the case where $d = n-1$, so this bound becomes $\frac{\mathcal{M}(n-1)}{k(n-k)}$. There are $n-1$ nodes that each contain $\frac{\mathcal{M}}{k}$, so each node is transmitting $\frac{1}{n-k}$ of its contents. Because of this, we store $n-k$ packets of data in each storage node. We break the source data up into packets of the same size and each storage packet will be some linear combination of the $k(n-k)$ packets of source data.

2.2 Notation

We use several matrices to represent the data and coefficients used in an MSR code.

$\mathbf{A}_i$	$(n-k)$	$\times$	$k(n-k)$	matrix of storage coefficients
$\mathbf{B}_{i,j}$	1	$\times$	$(n-k)$	row vector of transmission coefficients
$\mathbf{C}_i$	$(n-k)$	$\times$	$(n-1)$	matrix used to rebuild storage node i
$\mathbf{D}$	$k(n-k)$	$\times$	x	matrix of source data

The ith storage node contains $\mathbf{A}_i\mathbf{D}$, the original data multiplied by the storage coefficients for that node.

2.3 Independence

The storage nodes of the code are independent if any k nodes can reproduce the original data. That is, for all combinations of k storage nodes, there is a matrix $\mathbf{M}$ such that

$$\mathbf{D} = \mathbf{M}\begin{pmatrix} \mathbf{A}_{c(1)} \\ \mathbf{A}_{c(2)} \\ \vdots \\ \mathbf{A}_{c(k)} \end{pmatrix}\mathbf{D} \quad \text{or equivalently} \quad \det\begin{pmatrix} \mathbf{A}_{c(1)} \\ \mathbf{A}_{c(2)} \\ \vdots \\ \mathbf{A}_{c(k)} \end{pmatrix} \neq 0$$

Each combination of nodes must produce a full rank matrix.

2.4 Recovery

When node j fails, the ith node transmits $\mathbf{B}_{i,j}\mathbf{A}_i\mathbf{D}$. The code allows the recovery of node j if there is a matrix $\mathbf{C}_i$ that recreates the lost node from the transmitted vectors:

$$\mathbf{A}_j\mathbf{D} = \mathbf{C}_i\begin{pmatrix} \mathbf{B}_{i,j}\mathbf{A}_1 \\ \vdots \\ \mathbf{B}_{i,j}\mathbf{A}_{j-1} \\ \mathbf{B}_{i,j}\mathbf{A}_{j+1} \\ \vdots \\ \mathbf{B}_{i,j}\mathbf{A}_n \end{pmatrix}\mathbf{D} \quad \text{or equivalently} \quad \mathbf{A}_j = \mathbf{C}_j\begin{pmatrix} \mathbf{B}_{1,j}\mathbf{A}_1 \\ \vdots \\ \mathbf{B}_{j-1,j}\mathbf{A}_{j-1} \\ \mathbf{B}_{j+1,j}\mathbf{A}_{j+1} \\ \vdots \\ \mathbf{B}_{n,j}\mathbf{A}_n \end{pmatrix}$$

$\mathbf{D}$ drops out of both the independence and recovery conditions. We can thus talk about the coefficients only. We can also ignore the $\mathbf{C}_i$ matrices. From the recovery condition, we can see that in a working code the $\mathbf{C}_i$ matrices are fully specified by the $\mathbf{A}_i$ and $\mathbf{B}_{i,j}$ matrices. With these two conditions, we can determine whether a set of $\mathbf{A}_i$ and $\mathbf{B}_{i,j}$ matrices form a code.

2.5 General Position

A stronger version of the independence condition is also interesting. A collection of n-dimensional vectors is in general condition if every combination of n vectors is full rank. To apply this condition to a code, we consider the rows of the $\mathbf{A}_i$ matrices as vectors. In order to satisfy the independence condition there are $\begin{pmatrix} n \\ k \end{pmatrix}$ combinations of vectors that must be full rank, while there are $\begin{pmatrix} n(n-k) \\ k(n-k) \end{pmatrix}$ that must be full rank for the vectors of the code to be in general position.

3 Rotationally Symmetric Codes

To reduce the total number of coefficients, we consider codes whose $\mathbf{A}_i$ matrices are related to each other by a simple transformation.
Let $\mathbf{R}$ be an $k(n-k) \times k(n-k)$ matrix such that $\mathbf{R}^n = \mathbf{I}$, and let $\mathbf{A}_i = \mathbf{A}\mathbf{R}^i$.
For a discussion of the $\mathbf{R}$ matrices themselves, see the appendix. This reduces the number of storage coefficients needed to specify a code by a factor of n, reducing the search space exponentially.

3.1 Recovery Condition

This makes the recovery condition

$$\begin{aligned}
\mathbf{A}\mathbf{R}^j &= \mathbf{C}_j \begin{pmatrix} \mathbf{B}_{1,j}\mathbf{A}\mathbf{R}^1 \\ \vdots \\ \mathbf{B}_{j-1,j}\mathbf{A}\mathbf{R}^{j-1} \\ \mathbf{B}_{j+1,j}\mathbf{A}\mathbf{R}^{j+1} \\ \vdots \\ \mathbf{B}_{n,j}\mathbf{A}\mathbf{R}^n \end{pmatrix} \\
\mathbf{A} &= \mathbf{C}_j \begin{pmatrix} \mathbf{B}_{1,j}\mathbf{A}\mathbf{R}^1 \\ \vdots \\ \mathbf{B}_{j-1,j}\mathbf{A}\mathbf{R}^{j-1} \\ \mathbf{B}_{j+1,j}\mathbf{A}\mathbf{R}^{j+1} \\ \vdots \\ \mathbf{B}_{n,j}\mathbf{A}\mathbf{R}^n \end{pmatrix} \mathbf{R}^{n-j} \\
&= \mathbf{C}_j \begin{pmatrix} \mathbf{B}_{1,j}\mathbf{A}\mathbf{R}^{n-j+1} \\ \vdots \\ \mathbf{B}_{j-1,j}\mathbf{A}\mathbf{R}^{n-1} \\ \mathbf{B}_{j+1,j}\mathbf{A}\mathbf{R}^1 \\ \vdots \\ \mathbf{B}_{n,j}\mathbf{A}\mathbf{R}^{n-j} \end{pmatrix}
\end{aligned}$$

We can replace $\mathbf{B}_{i,j}$ with $\mathbf{B}_{i-j}$, reorder the rows of the transmitted coefficient matrix, and replace $\mathbf{C}_j$ with $\mathbf{C}$. Now there is only one recovery condition.

$$\mathbf{A} = \mathbf{C}\begin{pmatrix} \mathbf{B}_1\mathbf{A}\mathbf{R}^1 \\ \vdots \\ \mathbf{B}_{n-1}\mathbf{A}\mathbf{R}^{n-1} \end{pmatrix}$$

This is an improvement of a factor of n.

3.2 Independence Condition

Similarly, when checking independence, we only need to check combinations that include the first node.

$$\det\begin{pmatrix} \mathbf{A}\mathbf{R}^{c(1)} \\ \mathbf{A}\mathbf{R}^{c(2)} \\ \vdots \\ \mathbf{A}\mathbf{R}^{c(k)} \end{pmatrix} = \det\begin{pmatrix} \mathbf{A}\mathbf{R}^1 \\ \mathbf{A}\mathbf{R}^{c(2)-c(1)+1} \\ \vdots \\ \mathbf{A}\mathbf{R}^{c(k)-c(1)+1} \end{pmatrix}\det\mathbf{R}^{c(1)}$$

This reduces the number of conditions from $\begin{pmatrix} n \\ k \end{pmatrix}$ to $\begin{pmatrix} n-1 \\ k-1 \end{pmatrix}$. This is an improvement of a factor of $\frac{n}{k}$.

3.3 Example

$$\mathbf{R} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \quad \mathbf{A}_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix} \quad \mathbf{A}_3 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{A}_2 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \quad \mathbf{A}_4 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

The $\mathbf{B}_i$ matrices gives us the transmitted vectors.

$$\mathbf{B}_1\mathbf{A}_2 = \begin{pmatrix} 1 & 0 \end{pmatrix}\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \end{pmatrix}$$

$$\mathbf{B}_2\mathbf{A}_3 = \begin{pmatrix} 1 & 0 \end{pmatrix}\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\mathbf{B}_3\mathbf{A}_4 = \begin{pmatrix} 0 & 1 \end{pmatrix}\begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 \end{pmatrix}$$

From these we can complete the code by calculating $\mathbf{C}$.

$$\mathbf{A}_1 = \mathbf{C}\begin{pmatrix} \mathbf{B}_1\mathbf{A}_2 \\ \mathbf{B}_2\mathbf{A}_3 \\ \mathbf{B}_3\mathbf{A}_4 \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix} = \mathbf{C}_1 \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \qquad \mathbf{C}_1 = \begin{pmatrix} -1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

4 Additional Recovered Coefficients

The $\mathbf{B}_{i,j}$ matrices cannot be eliminated in a similarly simple manner, but their contribution to the code to be represented in alternative way. During recovery $n-1$ vectors are transmitted to the lost node, but the original $\mathbf{A}_i$ matrix has only $n-k$ rows. Thus $k-1$ additional vectors of coefficients are recovered. Specifying these vectors allows the $\mathbf{B}_{i,j}$ matrices to be determined.

Let $\mathbf{Z}_j$ be the $k-1 \times k(n-k)$ matrix that contains the additional rows recovered when node j is lost. Let $\mathbf{X}_j = \begin{pmatrix} \mathbf{Z}_j \\ \mathbf{A}_j \end{pmatrix}$ be the $n-1 \times k(n-k)$ matrix that contains all of the rows recovered when node j is lost. Then $\mathbf{X}_j^T(\mathbf{X}_j\mathbf{X}_j^T)^{-1}\mathbf{X}_j$ projects vectors into $\operatorname{span}\mathbf{X}_j$. A row vector $\mathbf{v}$ is in $\operatorname{span}\mathbf{X}_j$ if the projection does not change the vector, or $\mathbf{v}(\mathbf{X}_j^T(\mathbf{X}_j\mathbf{X}_j^T)^{-1}\mathbf{X}_j) = \mathbf{v}$. This can be rewritten as $\mathbf{v}(\mathbf{I} - \mathbf{X}_j^T(\mathbf{X}_j\mathbf{X}_j^T)^{-1}\mathbf{X}_j) = \mathbf{0}$.

$\mathbf{I} - \mathbf{X}_j^T(\mathbf{X}_j\mathbf{X}_j^T)^{-1}\mathbf{X}_j$ gives the difference between the original vector and the projection. This is a projection to the $(k-1)(n-k-1)$-dimensional space $\mathbb{F}^n/\operatorname{span}\mathbf{X}_j$. The only potentially useful vectors to transmit during recovery are those in $\operatorname{span}\mathbf{X}_j$, so we need to ensure that the transmitted vector $\mathbf{B}_{i,j}\mathbf{A}_i$ must satisfy $\mathbf{B}_{i,j}\mathbf{A}_i(\mathbf{I} - \mathbf{X}_j^T(\mathbf{X}_j\mathbf{X}_j^T)^{-1}\mathbf{X}_j) = \mathbf{0}$. Thus the choices for $\mathbf{B}_{i,j}$ are the vectors in the nullspace of $\mathbf{A}_i(\mathbf{I} - \mathbf{X}_j^T(\mathbf{X}_j\mathbf{X}_j^T)^{-1}\mathbf{X}_j)$.

4.1 Unrecovered Coefficients

Let $\mathbf{Y}_j$ refer to a basis that spans $\mathbb{F}^n/\operatorname{span}\mathbf{X}_j$. Now we can rewrite the projection as $\mathbf{I} - \mathbf{X}_j^T(\mathbf{X}_j\mathbf{X}_j^T)^{-1}\mathbf{X}_j = \mathbf{Y}_j^T(\mathbf{Y}_j\mathbf{Y}_j^T)^{-1}\mathbf{Y}_j$. Now we can say $\mathbf{B}_{i,j}\mathbf{A}_i$ must satisfy $\mathbf{B}_{i,j}\mathbf{A}_i\mathbf{Y}_j^T(\mathbf{Y}_j\mathbf{Y}_j^T)^{-1}\mathbf{Y}_j = \mathbf{0}$, which reduces to $\mathbf{B}_{i,j}\mathbf{A}_i\mathbf{Y}_j^T = 0$. Thus the null space of $\mathbf{A}_i(\mathbf{I} - \mathbf{X}_j^T(\mathbf{X}_j\mathbf{X}_j^T)^{-1}\mathbf{X}_j)$ is the same as the nullspace of $\mathbf{A}_i\mathbf{Y}_j^T$. $\mathbf{A}_i\mathbf{Y}_j^T$ is a $(n-k) \times (k-1)(n-k-1)$ matrix, so its nullity is at least $(n-k) - (k-1)(n-k-1)$ or $1 + (n-k-1)(2-k)$. However, if $2 < k < n-1$, this bound does not force the nullity to be positive. This bound does explain why it is so easy to find codes when $k = 2$.

4.2 Example: Obtaining B from Y

Now we can see how the $\mathbf{B}_{i,j}$ vectors were discovered in the previous example. Let $\mathbf{Y}_1 = \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}$. Note that $\mathbf{A}_1\mathbf{Y}_1^T = 0$ as required. We apply $\mathbf{Y}_1^T$ to the other $\mathbf{A}_i$ matrices and find the $\mathbf{B}_{i,j}$ vectors that satisfy $\mathbf{B}_{i,j}\mathbf{A}_i\mathbf{Y}_j^T = 0$.

$$\begin{aligned}
\mathbf{A}_2\mathbf{Y}_1^T &= \begin{pmatrix} 0 \\ 1 \end{pmatrix} & \mathbf{B}_1 &= \begin{pmatrix} 1 & 0 \end{pmatrix} \\
\mathbf{A}_3\mathbf{Y}_1^T &= \begin{pmatrix} 0 \\ 1 \end{pmatrix} & \mathbf{B}_2 &= \begin{pmatrix} 1 & 0 \end{pmatrix} \\
\mathbf{A}_4\mathbf{Y}_1^T &= \begin{pmatrix} 1 \\ 0 \end{pmatrix} & \mathbf{B}_3 &= \begin{pmatrix} 0 & 1 \end{pmatrix}
\end{aligned}$$

Let $\mathbf{Y}_1 = (\ 0 \quad 1 \quad -1 \quad 0\ )$ instead, another vector that satisfies $\mathbf{A}_1\mathbf{Y}_1^T = 0$.

$$\begin{aligned}
\mathbf{A}_2\mathbf{Y}_1^T &= \begin{pmatrix} 1 \\ -1 \end{pmatrix} & \mathbf{B}_1 &= \begin{pmatrix} 1 & 1 \end{pmatrix} & \mathbf{B}_1\mathbf{A}_2 &= \begin{pmatrix} 0 & 1 & 1 & 1 \end{pmatrix} \\
\mathbf{A}_3\mathbf{Y}_1^T &= \begin{pmatrix} -1 \\ 0 \end{pmatrix} & \mathbf{B}_2 &= \begin{pmatrix} 0 & 1 \end{pmatrix} & \mathbf{B}_2\mathbf{A}_3 &= \begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix} \\
\mathbf{A}_4\mathbf{Y}_1^T &= \begin{pmatrix} 0 \\ 1 \end{pmatrix} & \mathbf{B}_3 &= \begin{pmatrix} 1 & 0 \end{pmatrix} & \mathbf{B}_3\mathbf{A}_4 &= \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}
\end{aligned}$$

$$\mathbf{A}_1 = \mathbf{C}\begin{pmatrix} \mathbf{B}_1\mathbf{A}_2 \\ \mathbf{B}_2\mathbf{A}_3 \\ \mathbf{B}_3\mathbf{A}_4 \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix} = \mathbf{C}\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \mathbf{C} = \begin{pmatrix} 0 & 1 & -1 \\ 1 & 0 & -1 \end{pmatrix}$$

We obtain a new unique code. For the $n = 4$, $k = 2$ case, nearly all choices for $\mathbf{Y}_j$ produce a working code. This is not the case for larger coefficients. For $n = 4$, $k = 2$, the $\mathbf{Y}_j$ vectors that produce working codes form a vector space. The proof of this is in the appendix.

5 Transformations of codes and equivalence classes

5.1 Row transformations

Suppose we have an invertible $(n-k) \times (n-k)$ matrix $\mathbf{T}$ and a working code defined by $\mathbf{A}_i$ and $\mathbf{B}_{i,j}$ matrices. Then the matrices $\mathbf{TA}_i$ and $\mathbf{B}_{i,j}\mathbf{T}^{-1}$ also define a working code. For recoverability we have

$$\mathbf{TA}_j = \mathbf{TC}_j \begin{pmatrix} \mathbf{B}_{1,j}\mathbf{T}^{-1}\mathbf{TA}_1 \\ \vdots \\ \mathbf{B}_{j-1,j}\mathbf{T}^{-1}\mathbf{TA}_{j-1} \\ \mathbf{B}_{j+1,j}\mathbf{T}^{-1}\mathbf{TA}_{j+1} \\ \vdots \\ \mathbf{B}_{n,j}\mathbf{T}^{-1}\mathbf{TA}_n \end{pmatrix} = \mathbf{TC}_j \begin{pmatrix} \mathbf{B}_{1,j}\mathbf{A}_1 \\ \vdots \\ \mathbf{B}_{j-1,j}\mathbf{A}_{j-1} \\ \mathbf{B}_{j+1,j}\mathbf{A}_{j+1} \\ \vdots \\ \mathbf{B}_{n,j}\mathbf{A}_n \end{pmatrix}$$

and for independence we have

$$\det \begin{pmatrix} \mathbf{TA}_{c(1)} \\ \mathbf{TA}_{c(2)} \\ \vdots \\ \mathbf{TA}_{c(k)} \end{pmatrix} = \det \begin{pmatrix} \mathbf{T} & 0 & \dots & 0 \\ 0 & \mathbf{T} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \mathbf{T} \end{pmatrix} \det \begin{pmatrix} \mathbf{A}_{c(1)} \\ \mathbf{A}_{c(2)} \\ \vdots \\ \mathbf{A}_{c(k)} \end{pmatrix} = (\det \mathbf{T})^k \det \begin{pmatrix} \mathbf{A}_{c(1)} \\ \mathbf{A}_{c(2)} \\ \vdots \\ \mathbf{A}_{c(k)} \end{pmatrix} \neq 0$$

The row transformation is applied to the $\mathbf{A}$ matrices from the left and the rotation matrix in a rotationally symmetric code is applied from the right. Thus, applying the transformation to a rotationally symmetric code results in another rotationally symmetric code that uses the same rotation matrix. We can define codes to be equivalent if they are related by a row transformation. Testing only one code from each equivalence class reduces the search space by k^2 dimensions.

5.2 Column transformations

The same technique can be applied to the columns. If we have an invertible $k(n-k) \times k(n-k)$ matrix $\mathbf{T}$ and a working code defined by $\mathbf{A}_i$ and $\mathbf{B}_{i,j}$ matrices, then the matrices $\mathbf{A}_i\mathbf{T}$ and $\mathbf{B}_{i,j}$ also define a working code. For recoverability we have

$$\mathbf{A}_j\mathbf{T} = \mathbf{C}_j \begin{pmatrix} \mathbf{B}_{1,j}\mathbf{A}_1\mathbf{T} \\ \vdots \\ \mathbf{B}_{j-1,j}\mathbf{A}_{j-1}\mathbf{T} \\ \mathbf{B}_{j+1,j}\mathbf{A}_{j+1}\mathbf{T} \\ \vdots \\ \mathbf{B}_{n,j}\mathbf{A}_n\mathbf{T} \end{pmatrix} = \mathbf{C}_j \begin{pmatrix} \mathbf{B}_{1,j}\mathbf{A}_1 \\ \vdots \\ \mathbf{B}_{j-1,j}\mathbf{A}_{j-1} \\ \mathbf{B}_{j+1,j}\mathbf{A}_{j+1} \\ \vdots \\ \mathbf{B}_{n,j}\mathbf{A}_n \end{pmatrix} \mathbf{T}$$

and for independence we have

$$\det \begin{pmatrix} \mathbf{A}_{c(1)}\mathbf{T} \\ \mathbf{A}_{c(2)}\mathbf{T} \\ \vdots \\ \mathbf{A}_{c(k)}\mathbf{T} \end{pmatrix} = \det \begin{pmatrix} \mathbf{A}_{c(1)} \\ \mathbf{A}_{c(2)} \\ \vdots \\ \mathbf{A}_{c(k)} \end{pmatrix} \det \mathbf{T} \neq 0$$

In a rotationally symmetric code, the column transformation and the rotation are both applied from the right, so they interact.

$$\mathbf{A}_i\mathbf{T} = \mathbf{A}\mathbf{R}^i\mathbf{T} = \mathbf{A}\mathbf{T}\mathbf{T}^{-1}\mathbf{R}^i\mathbf{T} = \mathbf{A}\mathbf{T}(\mathbf{T}^{-1}\mathbf{R}\mathbf{T})^i$$

So the new code is rotationally symmetric with a different rotation matrix, $\mathbf{T}^{-1}\mathbf{R}\mathbf{T}$. This means that we can use a simple rotation matrix when searching for codes and simultaneously check all rotationally symmetric codes that use similar rotation matrices.

This also makes it possible to put any rotationally symmetric code into systematic form. When a code is in systematic form, the first k storage matrices can be stacked to form an identity matrix.

$$\begin{pmatrix} \mathbf{A}_1\mathbf{T} \\ \vdots \\ \mathbf{A}_k\mathbf{T} \end{pmatrix} = \mathbf{I}$$

Finding the transformation that puts a code into systematic form is simple. It is simply the inverse of the stack of first k storage matrices.

$$\mathbf{T} = \begin{pmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_k \end{pmatrix}^{-1}$$

5.3 Example: Systematic Form

$$\begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \qquad \mathbf{T} = \begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & 1 & 1 \end{pmatrix}$$

The same $\mathbf{B}$ vectors as before will work for recovery.

$$\begin{aligned}
\mathbf{B}_1\mathbf{A}_2\mathbf{T} &= \begin{pmatrix} 1 & 0 \end{pmatrix}\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix} \\
\mathbf{B}_2\mathbf{A}_3\mathbf{T} &= \begin{pmatrix} 1 & 0 \end{pmatrix}\begin{pmatrix} 0 & 1 & -1 & 0 \\ 1 & -1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & -1 & 0 \end{pmatrix} \\
\mathbf{B}_3\mathbf{A}_4\mathbf{T} &= \begin{pmatrix} 0 & 1 \end{pmatrix}\begin{pmatrix} 0 & -1 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \end{pmatrix}
\end{aligned}$$

The same $\mathbf{C}$ matrix as before will also work.

$$\mathbf{A}_1\mathbf{T} = \mathbf{C}\begin{pmatrix} \mathbf{B}_1\mathbf{A}_2 \\ \mathbf{B}_2\mathbf{A}_3 \\ \cdot\,\mathbf{B}_3\mathbf{A}_4 \end{pmatrix}\mathbf{T} \qquad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} -1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

6 Search Procedure

When searching for codes of a given n and k over a finite field, this procedure was used. Iterate over $\mathbf{A}$ matrices in a way that ensures that exactly one matrix from each row transformation equivalence class is produced. For each $\mathbf{A}$ matrix, produce the collection of n $\mathbf{A}_i$ matrices using a single simple column rotation matrix. Then test the independence condition. Test it before the recovery condition because it requires only $\mathbf{A}_i$ matrices. If the independence condition is met, iterate over the space of potential additional recovered coefficients. For each $\mathbf{X}_j$ matrix produced by this process, check the recovery condition. If the condition is met, this is a code.

7 Search results

7.1 $n = 4$, $k = 2$

These coefficients are small enough to all several fields to be searched exhaustively. We have searched the prime fields up to $GF(13)$. There are no codes in $GF(2)$, but in all larger fields codes are extremely easy to find. In all of these fields, nearly all of the potential codes that satisfy the independence condition also satisfy the recovery condition. As the field size increases, larger and larger fractions of the potential codes satisfy the independence condition. In $GF(3)$, 22% of potential codes satisfy the independence condition, and of these all satisfy the recovery condition. In $GF(13)$, 78% of potential codes satisfy the independence condition and of these 92% also satisfy the recovery condition.

7.2 $n = 5$, $k = 3$

For these coefficients, codes were not previously known. We have exhaustively searched $GF(2)$, $GF(3)$, $GF(4)$, and $GF(5)$ and randomly searched in larger fields. We found codes in $GF(3)$, $GF(4)$, $GF(7)$, and larger fields, but none in $GF(2)$ or $GF(5)$. While the codes we have found in smaller fields are not composed of vectors in general position, we found a code in $GF(17)$ that is. Several of these codes are given in the appendix.

7.3 $n = 6$, $k = 3$

For these coefficients, I have yet to find any codes. In $GF(3)$, only about 1% of potential codes satisfy the independence condition. In $GF(4)$ this number is about 14% and in $GF(5)$ it is about 30%.

References

[Dimakis et al., 2007] Dimakis, A., Godfrey, P., Wainwright, M., and Ramchandran, K. (2007). Network coding for distributed storage systems. In *Proc. of IEEE INFOCOM.*

8 Appendix

8.1 Rotation Matrices

The simplest way to construct a matrix $\mathbf{R}$ with period n is to have it rotate n of columns of any matrix it is applied to. If the dimension is also n, this matrix is

$$\mathbf{R}_n = \begin{pmatrix} \mathbf{0} & \mathbf{I}_n \\ 1 & \mathbf{0} \end{pmatrix}$$

Call this kind of rotation matrix a simple rotation matrix. If the dimension of the matrix is larger than the period, we can build the matrix by stringing multiple basic rotation matrices together along the diagonal.

$$\mathbf{R} = \begin{pmatrix} \mathbf{R}_a & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_b & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \dots & \mathbf{0} & \mathbf{R}_c \end{pmatrix}$$

The period of the entire matrix will be the least common multiple of the periods of the basic rotation matrices (a, b, and c).

If a matrix satisfies $\mathbf{R}^n = \mathbf{I}$, then it satisfies $(\det \mathbf{R})^n = 1$. We are only interested in real matrices so $\det \mathbf{R} = 1$ for odd n and $\det \mathbf{R} = \pm 1$ for even n. Thus its eigenvalues satisfy $\prod \lambda_i = \pm 1$. Additionally, $\operatorname{trace} \mathbf{R} \in \mathbb{Q}$ which implies that $(\sum \lambda_i) \in \mathbb{Q}$. Thus all of the eigenvalues must be roots of unity and they must occur in sets that sum to a rational number. This greatly restricts the number of classes of rotation matrices.

The eigenvalues of a simple rotation matrix of period n are the n roots of unity. When a rotation matrix is built from simple rotation matrices, each simple matrix contributes one eigenvalue equal to 1.

8.2 Rotation Matrices with too many eigenvalues equal to 1

If more than k of the eigenvalues of the rotation matrix $\mathbf{R}$ are equal to 1, it is impossible to use the $\mathbf{R}$ to construct a code that satisfies the independence condition. Suppose j of the eigenvalues of $\mathbf{R}$ are equal to 1. Decompose $\mathbf{R}$ as $\mathbf{R} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^*$ where the columns of $\mathbf{U}$ are the eigenvectors of $\mathbf{R}$ and $\mathbf{\Lambda}$ is a diagonal matrix of the eigenvalues with all of the 1's in the upper left. Thus $\mathbf{\Lambda}$ has the form $\mathbf{\Lambda} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Lambda}' \end{pmatrix}$.

Let $\mathbf{A} = \mathbf{A}'\mathbf{U}^*$. Now $\mathbf{A}_i = \mathbf{A}\mathbf{R}^i = \mathbf{A}'\mathbf{U}^*(\mathbf{U}\mathbf{\Lambda}\mathbf{U}^*)^i = \mathbf{A}'\mathbf{\Lambda}^i\mathbf{U}^*$. Now let $\mathbf{A}' = \begin{pmatrix} \mathbf{A}'' & \mathbf{A}''' \end{pmatrix}$ where $\mathbf{A}''$ contributes the left j columns and $\mathbf{A}'''$ contributes the right $k(n-k) - j$ columns. Thus $\mathbf{A}'\mathbf{\Lambda}^i =$

$\begin{pmatrix} \mathbf{A}'' & \mathbf{A}'''\mathbf{\Lambda}'^i \end{pmatrix}$. The matrix obtained by stacking $\mathbf{A}_1$ through $\mathbf{A}_k$ is

$$\begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_k \end{pmatrix} = \begin{pmatrix} \mathbf{A}'\mathbf{\Lambda}\mathbf{U}^* \\ \mathbf{A}'\mathbf{\Lambda}^2\mathbf{U}^* \\ \vdots \\ \mathbf{A}'\mathbf{\Lambda}^k\mathbf{U}^* \end{pmatrix} = \begin{pmatrix} \mathbf{A}'\mathbf{\Lambda} \\ \mathbf{A}'\mathbf{\Lambda}^2 \\ \vdots \\ \mathbf{A}'\mathbf{\Lambda}^k \end{pmatrix} \mathbf{U}^* = \begin{pmatrix} \mathbf{A}'' & \mathbf{A}'''\mathbf{\Lambda}' \\ \mathbf{A}'' & \mathbf{A}'''\mathbf{\Lambda}'^2 \\ \vdots & \vdots \\ \mathbf{A}'' & \mathbf{A}'''\mathbf{\Lambda}'^k \end{pmatrix} \mathbf{U}^*$$

This matrix must be full rank for the independence condition to be satisfied. $\mathbf{U}^*$ is full rank so we can ignore it. The other part of the matrix can be transformed by row operations into

$$\begin{pmatrix} \mathbf{A}'' & \mathbf{A}'''\mathbf{\Lambda}' \\ 0 & \mathbf{A}'''(\mathbf{\Lambda}'^2 - \mathbf{\Lambda}') \\ \vdots & \vdots \\ 0 & \mathbf{A}'''(\mathbf{\Lambda}'^k - \mathbf{\Lambda}') \end{pmatrix}$$

If $j > k$, the block of zeros in the lower left includes at least one entry on the diagonal. Then further row operations can be performed on the upper left and lower right blocks independently to make the whole matrix upper triangular. At this point, the determinant of the matrix is the product of the values on the diagonal. At least one of these values is 0, so the determinant is 0. This means that the independence condition cannot be satisfied if $j > k$ and no codes can be constructed using such a $\mathbf{R}$ matrix.

As a result of this restriction, there are pairs of n and k for which it is impossible to construct a workable $\mathbf{R}$ matrix by putting together simple rotation matrices. $n = 7$ and $k = 4$ are such a pair. 7 is prime so there are no shorter cycles that can be used in the construction of the matrix. $k(n - k) = 12$ is the dimension of $\mathbf{R}$, do that leaves 5 entries on the diagonal that can only be filled with 1's. Thus any $\mathbf{R}$ matrix will will have at least 6 eigenvalues equal to 1.

8.3 Unrecovered Coefficients when $n = 4$, $k = 2$

Suppose that you have $\mathbf{A}_i$ along with $\mathbf{B}_{i,j}$ that recovers $\mathbb{F}^n/\operatorname{span}\mathbf{Y}_j$ and $\mathbf{B}'_{i,j}$ that recovers $\mathbb{F}^n/\operatorname{span}\mathbf{Y}'_j$. Then we know that $\mathbf{B}_{i,j}\mathbf{A}_i\mathbf{Y}_j = 0$ and $\mathbf{B}'_{i,j}\mathbf{A}_i\mathbf{Y}'_j = 0$. If we let $\mathbf{Y}''_j = \alpha\mathbf{Y}_j + \beta\mathbf{Y}'_j$ with $\alpha, \beta \in \mathbb{F}$, then the set of coefficients $\mathbf{B}''_{i,j} = \alpha\mathbf{B}'_{i,j}\mathbf{A}_i\mathbf{Y}_j\mathbf{B}_{i,j} - \beta\mathbf{B}_{i,j}\mathbf{A}_i\mathbf{Y}'_j\mathbf{B}'_{i,j}$ recovers $\mathbb{F}^n/\operatorname{span}\mathbf{Y}''_j$:

$$\begin{aligned}
\mathbf{B}''_{i,j}\mathbf{A}_i\mathbf{Y}''_j &= (\alpha\mathbf{B}'_{i,j}\mathbf{A}_i\mathbf{Y}_j\mathbf{B}_{i,j} - \beta\mathbf{B}_{i,j}\mathbf{A}_i\mathbf{Y}'_j\mathbf{B}'_{i,j})\mathbf{A}_i(\alpha\mathbf{Y}_j + \beta\mathbf{Y}'_j) \\
&= \alpha^2(\mathbf{B}'_{i,j}\mathbf{A}_i\mathbf{Y}_j)(\mathbf{B}_{i,j}\mathbf{A}_i\mathbf{Y}_j) + \alpha\beta(\mathbf{B}'_{i,j}\mathbf{A}_i\mathbf{Y}_j)(\mathbf{B}_{i,j}\mathbf{A}_i\mathbf{Y}'_j) \\
&\quad -\alpha\beta(\mathbf{B}_{i,j}\mathbf{A}_i\mathbf{Y}'_j)(\mathbf{B}'_{i,j}\mathbf{A}_i\mathbf{Y}_j) - \beta^2(\mathbf{B}_{i,j}\mathbf{A}_i\mathbf{Y}'_j)(\mathbf{B}'_{i,j}\mathbf{A}_i\mathbf{Y}'_j) \\
&= \alpha^2(\mathbf{B}'_{i,j}\mathbf{A}_i\mathbf{Y}_j)(0) + \alpha\beta(\mathbf{B}'_{i,j}\mathbf{A}_i\mathbf{Y}_j)(\mathbf{B}_{i,j}\mathbf{A}_i\mathbf{Y}'_j) \\
&\quad -\alpha\beta(\mathbf{B}'_{i,j}\mathbf{A}_i\mathbf{Y}_j)(\mathbf{B}_{i,j}\mathbf{A}_i\mathbf{Y}'_j) - \beta^2(\mathbf{B}_{i,j}\mathbf{A}_i\mathbf{Y}'_j)(0) \\
&= 0
\end{aligned}$$

Thus for a given $\mathbf{A}_i$, the $\mathbf{Y}_j$ vectors that create a working code form a vector space.

8.4 $(5,3)$ code over $GF(3)$

$$\mathbf{A}_1 = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{Z}_1 = \begin{pmatrix} 2 & 0 & 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 & 1 & 2 \end{pmatrix}$$

$$\mathbf{B}_{2,1}\mathbf{A}_2 = \begin{pmatrix} 2 & 1 \end{pmatrix}\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 2 & 1 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{B}_{3,1}\mathbf{A}_3 = \begin{pmatrix} 1 & 0 \end{pmatrix}\begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

$$\mathbf{B}_{4,1}\mathbf{A}_4 = \begin{pmatrix} 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

$$\mathbf{B}_{5,1}\mathbf{A}_5 = \begin{pmatrix} 1 & 1 \end{pmatrix}\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 0 & 0 & 1 & 1 \end{pmatrix}$$

$$\mathbf{A}_1 = \mathbf{C}\begin{pmatrix} \mathbf{B}_{2,1}\mathbf{A}_2 \\ \mathbf{B}_{3,1}\mathbf{A}_3 \\ \mathbf{B}_{4,1}\mathbf{A}_4 \\ \mathbf{B}_{5,1}\mathbf{A}_5 \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 2 & 0 & 1 \\ 1 & 0 & 2 & 1 \end{pmatrix}\begin{pmatrix} 0 & 2 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 2 & 0 & 0 & 1 & 1 \end{pmatrix}$$

8.5 $(5,3)$ code over $GF(7)$

$$\mathbf{A}_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 2 & 0 \end{pmatrix}$$

$$\mathbf{Z}_1 = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 6 \\ 4 & 3 & 0 & 1 & 2 & 3 \end{pmatrix}$$

$$\mathbf{B}_{2,1}\mathbf{A}_2 = \begin{pmatrix} 0 & 1 \end{pmatrix}\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 2 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{B}_{3,1}\mathbf{A}_3 = \begin{pmatrix} 2 & 1 \end{pmatrix}\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 2 & 0 & 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 2 & 2 & 1 & 0 & 2 \end{pmatrix}$$

$$\mathbf{B}_{4,1}\mathbf{A}_4 = \begin{pmatrix} 5 & 1 \end{pmatrix}\begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 2 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 2 & 5 & 1 & 5 \end{pmatrix}$$

$$\mathbf{B}_{5,1}\mathbf{A}_5 = \begin{pmatrix} 6 & 1 \end{pmatrix}\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 2 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 2 & 6 & 6 \end{pmatrix}$$

$$\mathbf{A}_1 = \mathbf{C}_1 \begin{pmatrix} \mathbf{B}_{2,1}\mathbf{A}_2 \\ \mathbf{B}_{3,1}\mathbf{A}_3 \\ \mathbf{B}_{4,1}\mathbf{A}_4 \\ \mathbf{B}_{5,1}\mathbf{A}_5 \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 2 & 0 \end{pmatrix} = \begin{pmatrix} 3 & 0 & 2 & 2 \\ 4 & 4 & 1 & 6 \end{pmatrix}\begin{pmatrix} 2 & 0 & 1 & 0 & 0 & 0 \\ 0 & 2 & 2 & 1 & 0 & 2 \\ 0 & 0 & 2 & 5 & 1 & 5 \\ 1 & 0 & 0 & 2 & 6 & 6 \end{pmatrix}$$

8.6 General Position $(5,3)$ code over $GF(17)$

$$\mathbf{A}_1 = \begin{pmatrix} 1 & 11 & 0 & 13 & 9 & 16 \\ 15 & 1 & 6 & 1 & 5 & 11 \end{pmatrix}$$

$$\mathbf{Z}_1 = \begin{pmatrix} 3 & 14 & 1 & 0 & 5 & 15 \\ 1 & 4 & 0 & 1 & 11 & 4 \end{pmatrix}$$

$$\mathbf{B}_1\mathbf{A}_2 = \begin{pmatrix} 1 & 1 \end{pmatrix}\begin{pmatrix} 9 & 1 & 11 & 0 & 13 & 16 \\ 5 & 15 & 1 & 6 & 1 & 11 \end{pmatrix} = \begin{pmatrix} 14 & 16 & 12 & 6 & 14 & 10 \end{pmatrix}$$

$$\mathbf{B}_2\mathbf{A}_3 = \begin{pmatrix} 9 & 1 \end{pmatrix}\begin{pmatrix} 13 & 9 & 1 & 11 & 0 & 16 \\ 1 & 5 & 15 & 1 & 6 & 11 \end{pmatrix} = \begin{pmatrix} 16 & 1 & 7 & 15 & 6 & 2 \end{pmatrix}$$

$$\mathbf{B}_3\mathbf{A}_4 = \begin{pmatrix} 12 & 1 \end{pmatrix}\begin{pmatrix} 0 & 13 & 9 & 1 & 11 & 16 \\ 6 & 1 & 5 & 15 & 1 & 11 \end{pmatrix} = \begin{pmatrix} 6 & 4 & 11 & 10 & 14 & 16 \end{pmatrix}$$

$$\mathbf{B}_4\mathbf{A}_5 = \begin{pmatrix} 4 & 1 \end{pmatrix}\begin{pmatrix} 11 & 0 & 13 & 9 & 1 & 16 \\ 1 & 6 & 1 & 5 & 15 & 11 \end{pmatrix} = \begin{pmatrix} 11 & 6 & 2 & 7 & 2 & 7 \end{pmatrix}$$

$$\mathbf{A}_1 = \mathbf{C}_1 \begin{pmatrix} \mathbf{B}_{2,1}\mathbf{A}_2 \\ \mathbf{B}_{3,1}\mathbf{A}_3 \\ \mathbf{B}_{4,1}\mathbf{A}_4 \\ \mathbf{B}_{5,1}\mathbf{A}_5 \end{pmatrix} \qquad \begin{pmatrix} 1 & 11 & 0 & 13 & 9 & 16 \\ 15 & 1 & 6 & 1 & 5 & 11 \end{pmatrix} = \begin{pmatrix} 11 & 6 & 9 & 8 \\ 2 & 2 & 16 & 15 \end{pmatrix} \begin{pmatrix} 14 & 16 & 12 & 6 & 14 & 10 \\ 16 & 1 & 7 & 15 & 6 & 2 \\ 6 & 4 & 11 & 10 & 14 & 16 \\ 11 & 6 & 2 & 7 & 2 & 7 \end{pmatrix}$$