Hadamard Matrices and Hadamard Codes

Definition: A *Hadamard matrix H* of order n is an $n \times n$ matrix of 1s and -1s in which $HH^{\mathrm{T}} = nI_n$. (I_n is the $n \times n$ identify matrix.)

Equivalently, a Hadamard matrix is an $n \times n$ matrix of 1s and -1s in which any two distinct rows agree in exactly $n/2$ positions (and thus disagree in exactly $n/2$ positions.)

With this definition, the entries of the matrix don't need to be 1s and -1s. They could be chosen from $\{\mathbf{red}, \mathbf{green}\}$ or $\{\mathbf{0}, \mathbf{1}\}$.

PROP. A Hadamard matrix can exist only if n is 1, 2, or a multiple of 4.

It has been conjectured that Hadamard matrices exist for any n that is a multiple of 4.

This is probably true but has not been proven. (It has been verified at least through $n = 664$.)

We will be interested primarily in the case where n is a power of 2, in which case Hadamard matrices are known to exist.

This will follow from the proposition below once we demonstrate a Hadamard matrix of order 2.

PROP. If H is a Hadamard matrix of order n, then $\begin{bmatrix} H & H \\ H & -H \end{bmatrix}$ is a Hadamard matrix of order $2n$.

Examples of Hadamard Matrices:

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$H_4 = \left[\begin{array}{cc|cc} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ \hline 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{array}\right]$$

$$H_8 = \left[\begin{array}{cccc|cccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ \hline 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{array}\right]$$

$$H_{16} = \left[\begin{array}{cccccccc|cccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \end{array}\right]$$

If H is a Hadamard matrix of order n, consider the $2n \times n$ matrix

$$C = \begin{pmatrix} H \\ -H \end{pmatrix}$$

Let $\mathbf{c}_i$ be the i^{th} row vector of C.

PROP. If $1 \le i \le j \le 2n$, rows $\mathbf{c}_i$ and $\mathbf{c}_j$ of C agree in

i) n positions if $j = i$,

ii) 0 positions if $j = i+n$,

iii) $n/2$ positions otherwise.

Let $\mathbf{v}$ be a vector or 1s and -1s of length n.

a) If for some i, $\mathbf{v}$ differs from $\mathbf{c}_i$ in at most $n/4-1$ positions, then it differs from $\mathbf{c}_j$ in at least $n/4+1$ positions, whenever $j \neq$ i.

b) If for some i, $\mathbf{v}$ differs from $\mathbf{c}_i$ in $n/4$ positions, then it differs from $\mathbf{c}_j$ in at least $n/4$ positions.

Note this tells us

a') If $\mathbf{v}$ differs from $\mathbf{c}_i$ in at most $n/4-1$ positions, then $\mathbf{v}$ is strictly "closer" to row $\mathbf{c}_i$ than to any other row of C. ("Closest" will mean differing in the fewest number of positions.)

b') If $\mathbf{v}$ differs from $\mathbf{c}_i$ in $n/4$ positions, $\mathbf{c}_i$ is at least tied for "closest" to $\mathbf{v}$, among the rows of C.

We can use C to encode a text over an alphabet $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_{2n}\}$ as follows:

To encode: $\sigma_i \rightarrow \mathbf{c}_i$.

To decode: If we receive the n-vector $\mathbf{v}$, we search for a row $\mathbf{c}_i$ of C that differs from $\mathbf{v}$ in at most $n/4 - 1$ positions.

i) If $\mathbf{c}_i$ exists, then we decode to σ_i. (Assuming at most $n/4$ errors occur, this is always correct.)

ii) Otherwise we detect an error. (Assuming at most $n/4$ errors occur, an error that is not corrected will always be detected.)

This code has $2n$ codewords of length n.

The minimum distance between any distinct codewords is $n/2$.

Under "minimal distance decoding", we can always correct $n/4 - 1$ errors in an n-bit encoded block and, in addition, detect $n/4$ errors.

It is more convenient to change the encoding alphabet from $\{-1, 1\}$ to $\{0, 1\}$, which we treat as Z_2.

Even then, the code C is not in general linear over Z_2. (The sum of codewords is not in general a codeword.)

In fact, it can't be linear if n is not a power of 2, as the number of codewords ($2n$) is not a power of 2.

However, if C is derived from one of the codes H_2, H_4, H_8, H_{16}, ..., then the code is linear. Call this code C_i. Here are the codewords of C_{16}.

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
1	0	1	0	0	1	0	1	1	0	1	0	0	1	0	1
1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1
1	0	0	1	0	1	1	0	1	0	0	1	0	1	1	0
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
1	0	1	0	1	0	1	0	0	1	0	1	0	1	0	1
1	1	0	0	1	1	0	0	0	0	1	1	0	0	1	1
1	0	0	1	1	0	0	1	0	1	1	0	0	1	1	0
1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1
1	0	1	0	0	1	0	1	0	1	0	1	1	0	1	0
1	1	0	0	0	0	1	1	0	0	1	1	1	1	0	0
1	0	0	1	0	1	1	0	0	1	1	0	1	0	0	1

(remaining 16 vectors obtained from above by flipping 0 and 1)

Rows 1, 2, 3, 5, and 9 form a basis for the code. Listing them in the order 1, 9, 5, 3, 2 we obtain a generator matrix for the code.

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

This code encodes blocks of length 5 to blocks of length 16.

The rate is 5/16, or about 0.31 — not very good.

But it can correct 3 errors in any 16-bit encoded block, and detect a fourth.

If we move on to the Hadamard code based on H_{32}, the generator matrix will be a 6×32.

This code has an even worse rate: 6/32, or about 0.19.

But it can correct 7 errors in any 32-bit encoded block, and detect an eighth.

This code was used on a Mariner spacecraft in 1969, to broadcast pictures back to earth.

In general, the Hadamard code based on the Hadamard matrix H_n, where $n = 2^k$, has a generator matrix that is $(k+1) \times 2^k$.

The rate is $(k+1)/2^k$ — terrible, especially as k increases.

The code can correct $2^{k-2} - 1$ errors in a 2^k-bit encoded block, and in addition detect one more error — excellent.

Both the Hamming codes and the Hadamard codes are actually special cases of a more general class of codes: *Reed-Muller* codes.

If $\mathbf{a} = (a_1, a_2, ..., a_n)$ and $\mathbf{b} = (b_1, b_2, ..., b_n)$ are vectors over Z_2, define

$$\mathbf{ab} = (a_1 b_2, a_2 b_2, ..., a_n b_n).$$ [not the usual definition of product]

$\mathbf{g}_0 =$	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
$\mathbf{g}_1 =$	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$\mathbf{g}_2 =$	1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
$\mathbf{g}_3 =$	1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0
$\mathbf{g}_4 =$	1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0
$\mathbf{g}_5 =$	1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
$\mathbf{g}_1\mathbf{g}_2 =$	1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$\mathbf{g}_1\mathbf{g}_3 =$	1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$\mathbf{g}_1\mathbf{g}_5 =$	1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$\mathbf{g}_1\mathbf{g}_5 =$	1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$\mathbf{g}_2\mathbf{g}_3 =$	1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
$\mathbf{g}_2\mathbf{g}_4 =$	1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0
$\mathbf{g}_2\mathbf{g}_5 =$	1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0
$\mathbf{g}_3\mathbf{g}_4 =$	1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0
$\mathbf{g}_3\mathbf{g}_5 =$	1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0
$\mathbf{g}_4\mathbf{g}_5 =$	1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0
$\mathbf{g}_1\mathbf{g}_2\mathbf{g}_3 =$	1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$\mathbf{g}_1\mathbf{g}_2\mathbf{g}_4 =$	1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$\mathbf{g}_1\mathbf{g}_2\mathbf{g}_5 =$	1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$\mathbf{g}_1\mathbf{g}_3\mathbf{g}_4 =$	1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$\mathbf{g}_1\mathbf{g}_3\mathbf{g}_5 =$	1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$\mathbf{g}_1\mathbf{g}_4\mathbf{g}_5 =$	1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$\mathbf{g}_2\mathbf{g}_3\mathbf{g}_4 =$	1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$\mathbf{g}_2\mathbf{g}_3\mathbf{g}_5 =$	1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
$\mathbf{g}_2\mathbf{g}_4\mathbf{g}_5 =$	1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
$\mathbf{g}_3\mathbf{g}_4\mathbf{g}_5 =$	1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
$\mathbf{g}_1\mathbf{g}_2\mathbf{g}_3\mathbf{g}_4 =$	1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$\mathbf{g}_1\mathbf{g}_2\mathbf{g}_3\mathbf{g}_5 =$	1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$\mathbf{g}_1\mathbf{g}_2\mathbf{g}_4\mathbf{g}_5 =$	1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$\mathbf{g}_1\mathbf{g}_3\mathbf{g}_4\mathbf{g}_5 =$	1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$\mathbf{g}_2\mathbf{g}_3\mathbf{g}_4\mathbf{g}_5 =$	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Notice there are 31 vectors divided into 5 groups, having 1, 5, 10, 10, and 5 vectors.

The first group (1 vector) is the generator matrix of the repetition code R_{32}.

It is a [32,1,32] code that can correct 15 errors in any 32-bit encoded block.

The first two groups (6 vectors) generate a Hadamard code.

It is a [32,6,16] code that can correct 7 errors in any 32-bit encoded block.

The first three groups (16 vectors) generate a certain Reed-Muller code

It is a [32,16,8] code that can correct 3 errors in any 32-bit encoded block.

The first four groups (26 vectors) generate a "extended" Hamming code. (Deleting a column gives a Hamming code.)

It is a [32,26,4] code that can correct 1 error in any 32-bit encoded block.

All five groups (31 vectors) generate a [32,31,2] code.

It is a simple parity check code, and can correct no errors.

In addition to correcting the number of errors specified above, each code can detect one additional error.