# 1 4-Point Hadamard Transform: Detailed Explanation and Computation

The Hadamard Transform is widely used in signal processing due to its simplicity and efficiency. Here, we compute the 4-point Hadamard Transform step by step.

## 1.1 Definition of the Hadamard Matrix

For a 4-point transform, the Hadamard matrix ($H_4$) is:

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}.$$

This matrix is symmetric and orthogonal, with elements $+1$ and $-1$. The Hadamard transform is computed as:

$$Y = H_4 \cdot X,$$

where $X = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \end{bmatrix}^T$ is the input vector, and $Y = \begin{bmatrix} y_0 & y_1 & y_2 & y_3 \end{bmatrix}^T$ is the output vector.

## 1.2 Step-by-Step Computation

The output components $y_0, y_1, y_2, y_3$ are computed as:

$$\begin{aligned} y_0 &= (x_0 + x_1 + x_2 + x_3), \\ y_1 &= (x_0 - x_1 + x_2 - x_3), \\ y_2 &= (x_0 + x_1 - x_2 - x_3), \\ y_3 &= (x_0 - x_1 - x_2 + x_3). \end{aligned}$$

## 1.3 Intermediate Computation (Decomposed into Two Stages)

To compute the output efficiently, we divide the computation into two stages:

**Stage 1: Compute Intermediate Sums and Differences.**

$$\begin{aligned} a_0 &= x_0 + x_1, & a_1 &= x_0 - x_1, \\ a_2 &= x_2 + x_3, & a_3 &= x_2 - x_3. \end{aligned}$$

**Stage 2: Combine Intermediate Results.** Using the intermediate values $a_0, a_1, a_2, a_3$, we calculate:

$$\begin{aligned} y_0 &= a_0 + a_2, & y_1 &= a_1 + a_3, \\ y_2 &= a_0 - a_2, & y_3 &= a_1 - a_3. \end{aligned}$$

## 1.4 Matrix Multiplication Representation

The transform can also be represented as explicit matrix multiplication:

$$Y = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}.$$

This results in:

$$\begin{aligned} y_0 &= x_0 + x_1 + x_2 + x_3, \\ y_1 &= x_0 - x_1 + x_2 - x_3, \\ y_2 &= x_0 + x_1 - x_2 - x_3, \\ y_3 &= x_0 - x_1 - x_2 + x_3. \end{aligned}$$

# 2 Number of Adders for Hadamard Transform

## 2.1 General Formula for $N$-Point Hadamard Transform

For an $N$-point Hadamard Transform, the total number of stages is:

$$\log_2(N),$$

and each stage requires:

$$\frac{N}{2} \text{ additions or subtractions.}$$

Thus, the total number of adders required is:

$$\text{Total Adders} = \frac{N}{2} \cdot \log_2(N).$$

## 2.2 Examples of Total Adders

1. 4-Point Hadamard Transform ($N = 4$):

$$\text{Total Adders} = \frac{4}{2} \cdot \log_2(4) = 2 \cdot 2 = 8 \, \text{Adders.}$$

2. 8-Point Hadamard Transform ($N = 8$):

$$\text{Total Adders} = \frac{8}{2} \cdot \log_2(8) = 4 \cdot 3 = 12 \, \text{Adders.}$$

3. 16-Point Hadamard Transform ($N = 16$):

$$\text{Total Adders} = \frac{16}{2} \cdot \log_2(16) = 8 \cdot 4 = 32 \, \text{Adders.}$$

## 2.3 Efficiency of the Hadamard Transform

The Hadamard transform is computationally efficient because: - It uses only additions and subtractions. - No multiplications are required because the Hadamard matrix consists of only $+1$ and $-1$.

## 2.4 Key Properties of the Transform

### 2.4.1 Energy Conservation:

The transform preserves the total energy of the input. Specifically:

$$\|Y\|^2 = \|X\|^2,$$

where $\|X\|$ and $\|Y\|$ are the Euclidean norms of the input and output vectors, respectively.

### 2.4.2 Simplicity:

Only additions and subtractions are used; no multiplications are required, making the Hadamard Transform computationally efficient.

### 2.4.3 Symmetry:

The structure of $H_4$ ensures that it captures both symmetric and anti-symmetric components of the input.

## 2.5 Applications of the Hadamard Transform

### 2.5.1 Signal Compression:

The Hadamard Transform is often used for efficient data compression.

#### 2.5.2 Error Detection and Correction:

Due to its orthogonality, the Hadamard Transform helps identify errors in transmitted data.

#### 2.5.3 Image Processing:

Used in binary image analysis and filtering.

#### 2.5.4 Quantum Computing:

A key component in quantum algorithms, particularly for generating superpositions.