

ÁLGEBRA LINEAR APLICADA

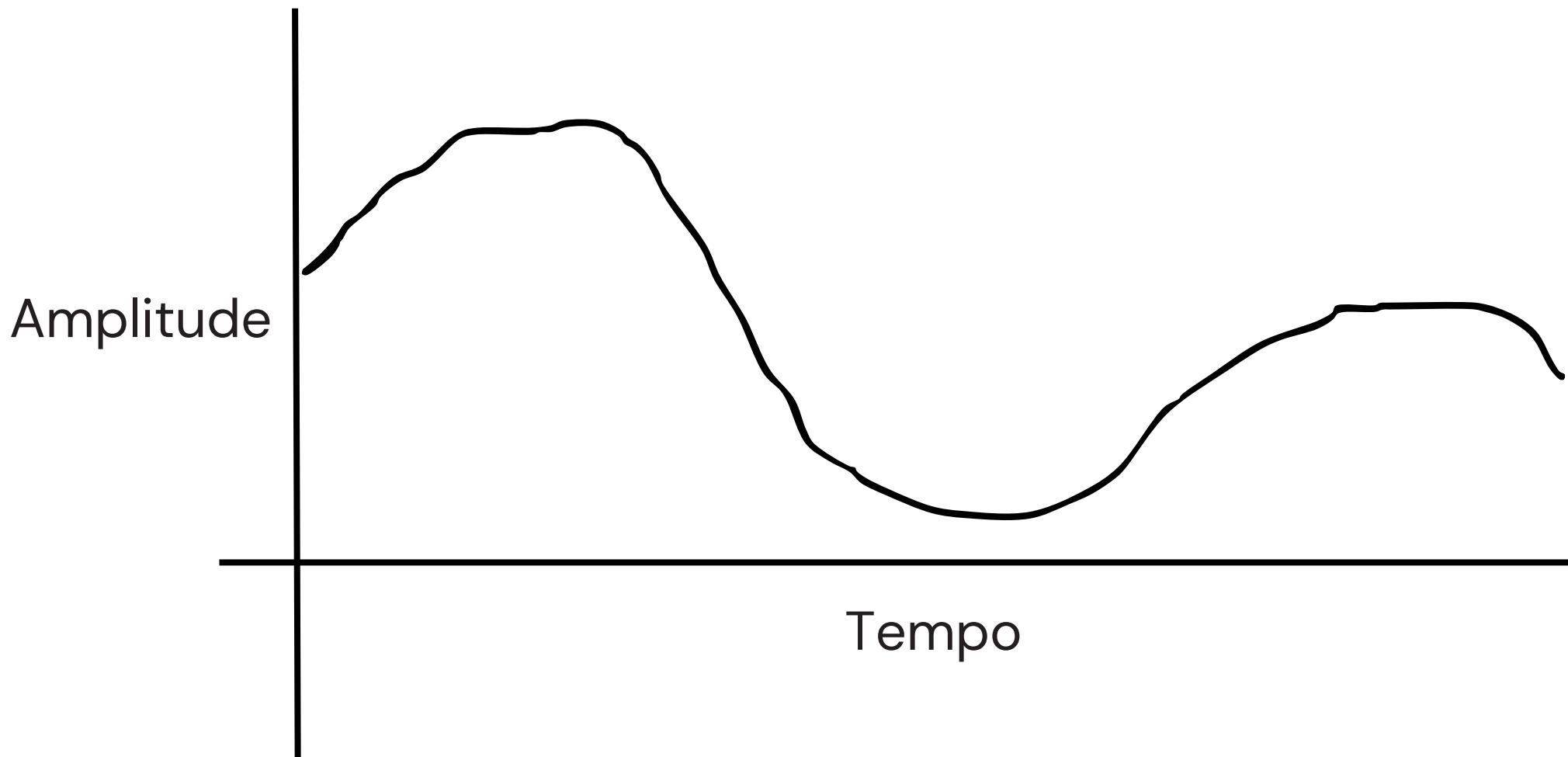
**TRANSFORMADA
DE
HADAMARD**

MANOEL SILVA

INTRODUÇÃO

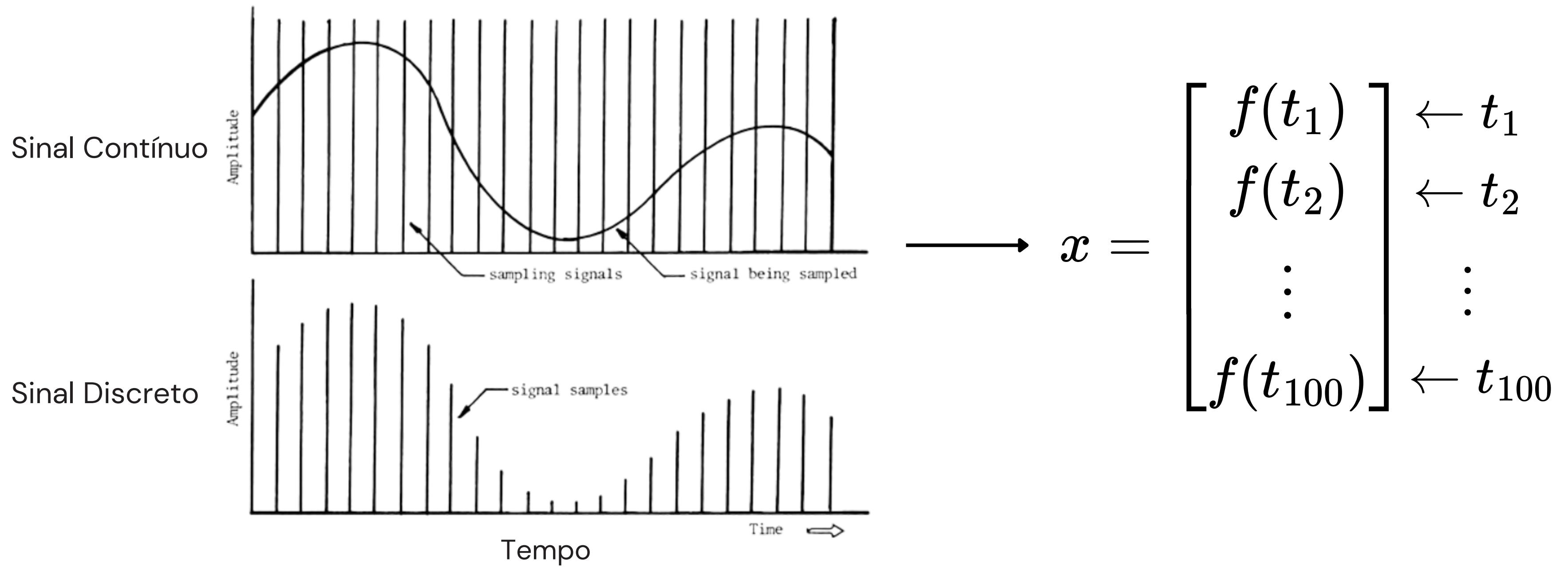
Sinais são um conjunto de dados observados que representam algum fenômeno, geralmente ao longo do tempo:

- Áudios
- Imagens
- Vídeos
- Redes sem fio
- Fenômenos físicos



INTRODUÇÃO

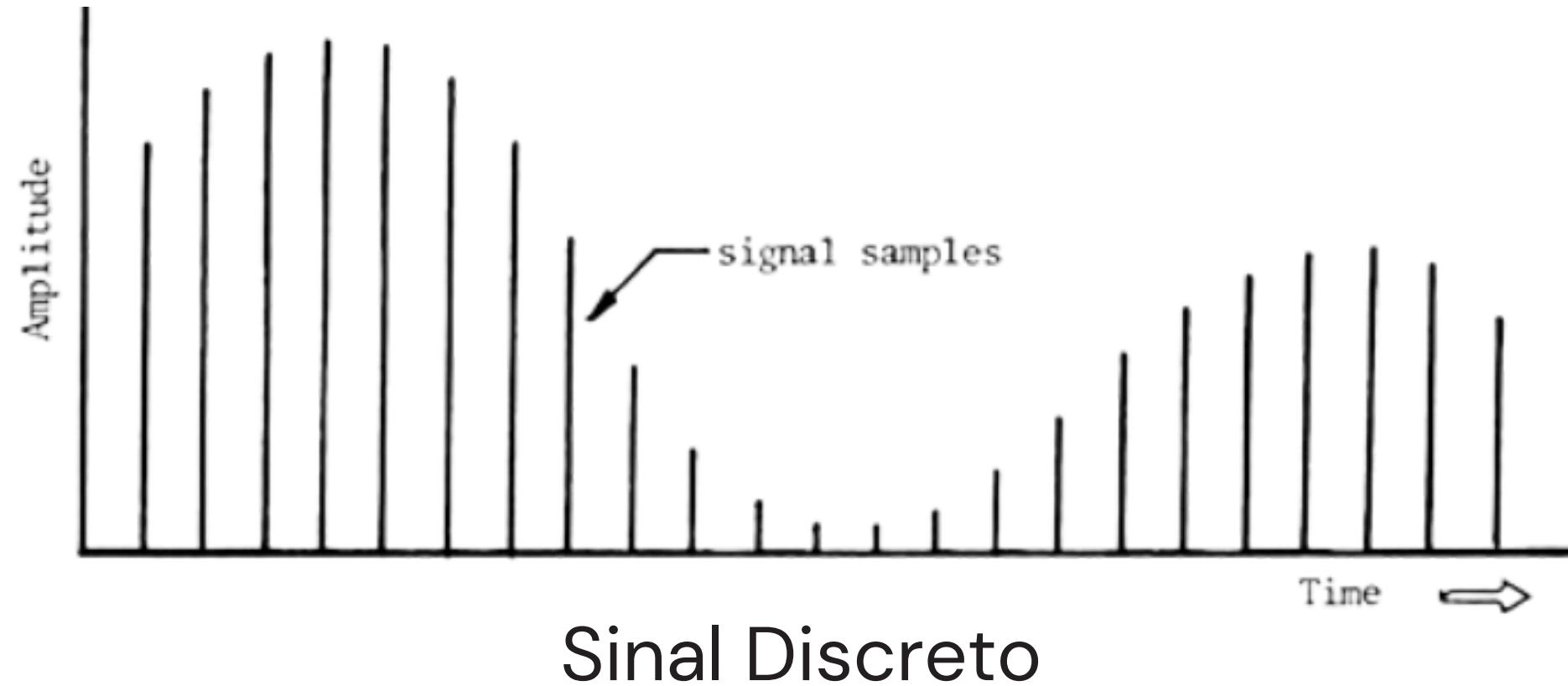
Apesar dos sinais serem contínuos na natureza, apenas conseguimos vê-los de forma discreta



INTRODUÇÃO

Imagine o seguinte problema:

- Descobrir a combinação de sinais que o compõe
- Comprimir Imagens
- Notas Musicais
- Prever Maré
- Remover Ruídos

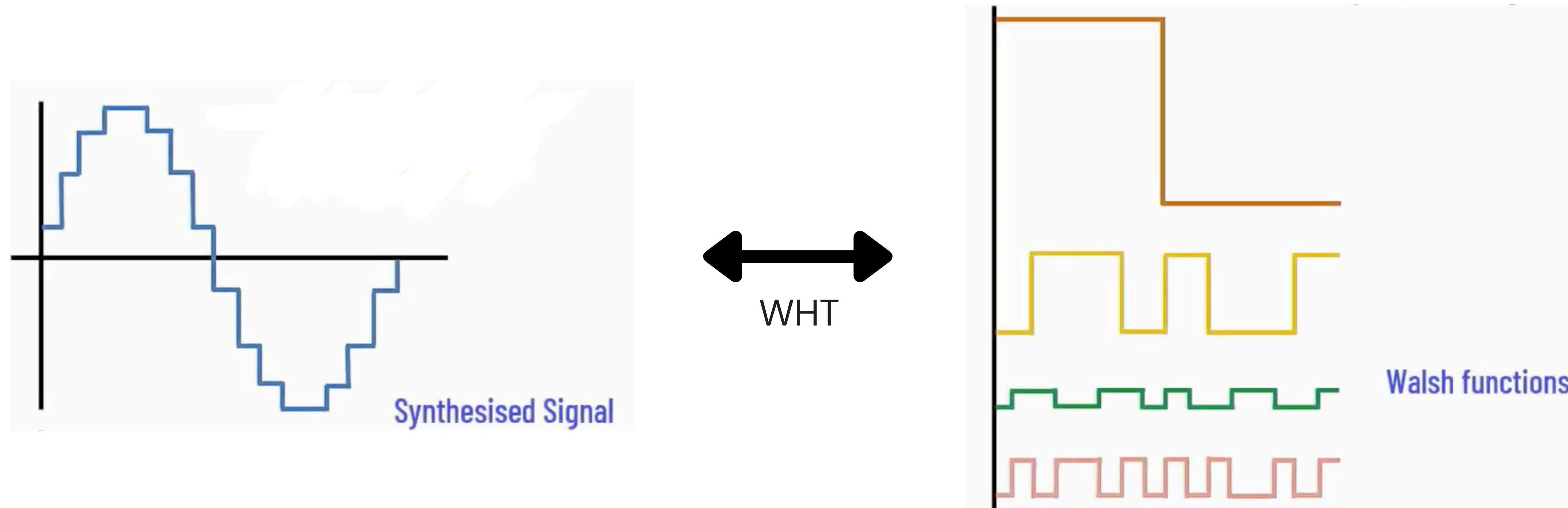


→ Como decompor esse sinal?

TRANSFORMADA DE HADAMARD

Também conhecida como Walsh–Hadamard transform (WHT) é uma classe de transformada de Fourier. Sendo esta uma transformação ortogonal não-sinoidal.

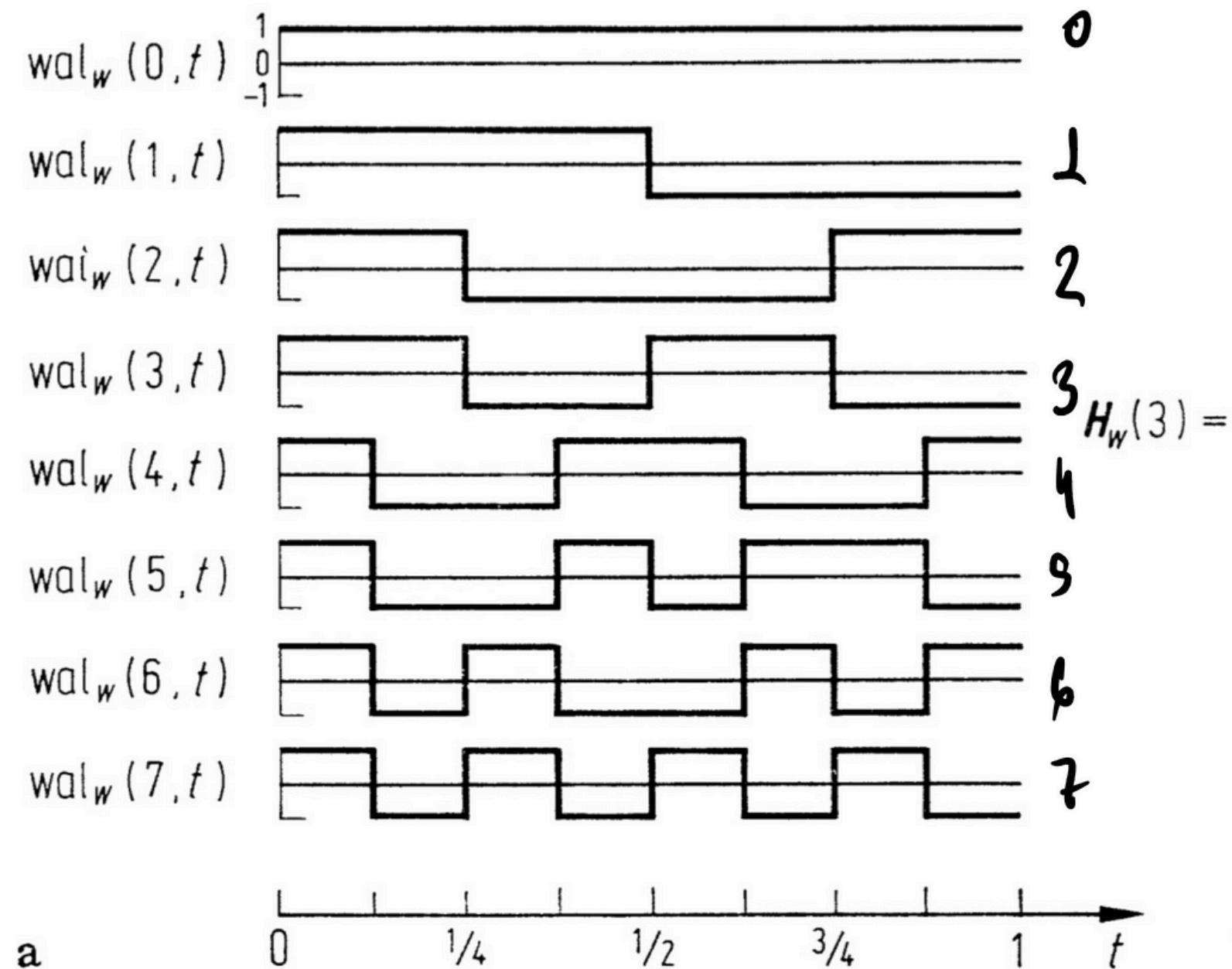
Usa como base de transformação sinais quadrados, que variam entre 1 e -1.



FUNÇÕES DE WALSH

São os sinais em que usaremos como base para a transformação que formam um conjunto ortonormal.

Cada onda quadrada é distinguida por um parâmetro chamado **Sequência**

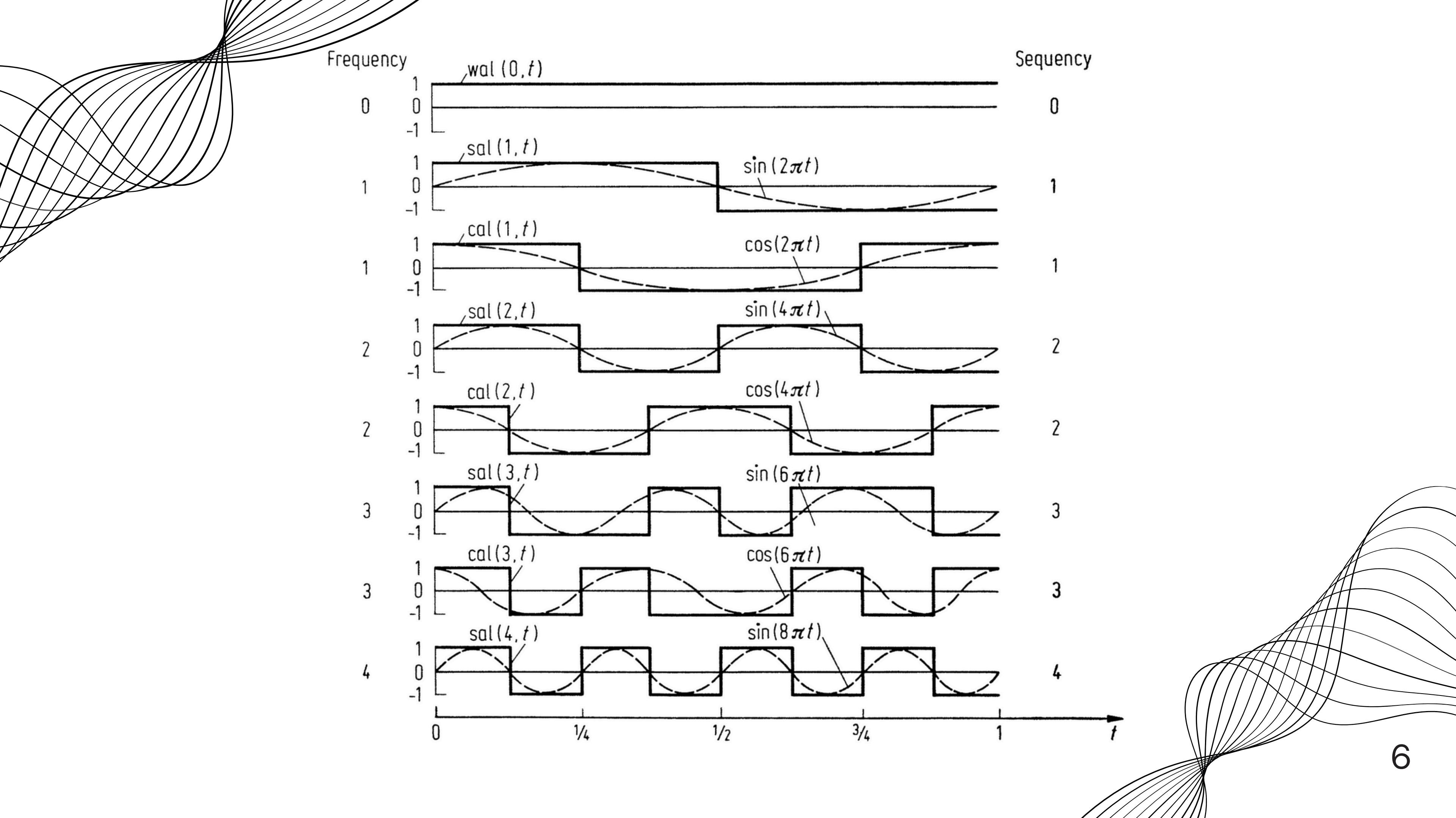


$$H_w(3) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}$$

b

A sequência é o número de vezes que o sinal cruza o zero.

A sequência está diretamente atrelada à frequência da onda



MATRIZ DE HADAMARD

Uma forma extremamente fácil de se gerar essas funções, de forma discreta, é pela matriz de Hadamard.

O método de Sylvester gera de forma recursiva uma matriz, em que cada linha/coluna é um desses sinais, que é base de nossa transformação.

$$n = \log_2(k),$$

$$H(0) = [1]$$

$$\begin{aligned} H(n) &= \begin{bmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{bmatrix} \\ &= \begin{bmatrix} H_{n-1} & 0 \\ 0 & H_{n-1} \end{bmatrix} * \begin{bmatrix} I & I \\ I & -I \end{bmatrix} \end{aligned}$$

$$H(2) = \begin{bmatrix} H_1 & H_1 \\ H_1 & -H_1 \end{bmatrix}$$

$$H(2) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

MATRIZ DE HADAMARD

Essa matriz possui 3 propriedades fundamentais para o WHT:

- Simétrica
- Ortogonal
- $\frac{1}{2^n} H_n = H_n^{-1}$ ($\frac{1}{\sqrt{2^n}} H_n = \frac{1}{\sqrt{2^n}} H_n^{-1}$)

Simétrica ($H_n = H_n^T$)

Base:

$$H(0) = [1]$$

$$H(0) = H^T(0) = [1]$$

OK

Hipótese de Indução

$$H_{n-1} = H_{n-1}^T$$

$$H(n) = \begin{bmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{bmatrix}$$

$$H^T(n) = \begin{bmatrix} H_{n-1}^T & H_{n-1}^T \\ H_{n-1}^T & -H_{n-1}^T \end{bmatrix}$$

↓ HI

$$H_n = H_n^T$$



MATRIZ DE HADAMARD

Ortogonal ($AA^T = I$)

Base:

$$H(0) = [1]$$

$$H(0)H^T(0) = [1] = I$$

OK

Hipótese de Indução

$$H_{n-1}H_{n-1}^T = 2^{n-1} * I$$

Indução

$$H(n) = \begin{bmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{bmatrix}$$

$$H_nH_n^T = \begin{bmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{bmatrix} \begin{bmatrix} H_{n-1}^T & H_{n-1}^T \\ H_{n-1}^T & -H_{n-1}^T \end{bmatrix}$$

$$H_nH_n^T = \begin{bmatrix} 2H_{n-1}H_{n-1}^T & 0 \\ 0 & 2H_{n-1}H_{n-1}^T \end{bmatrix}$$

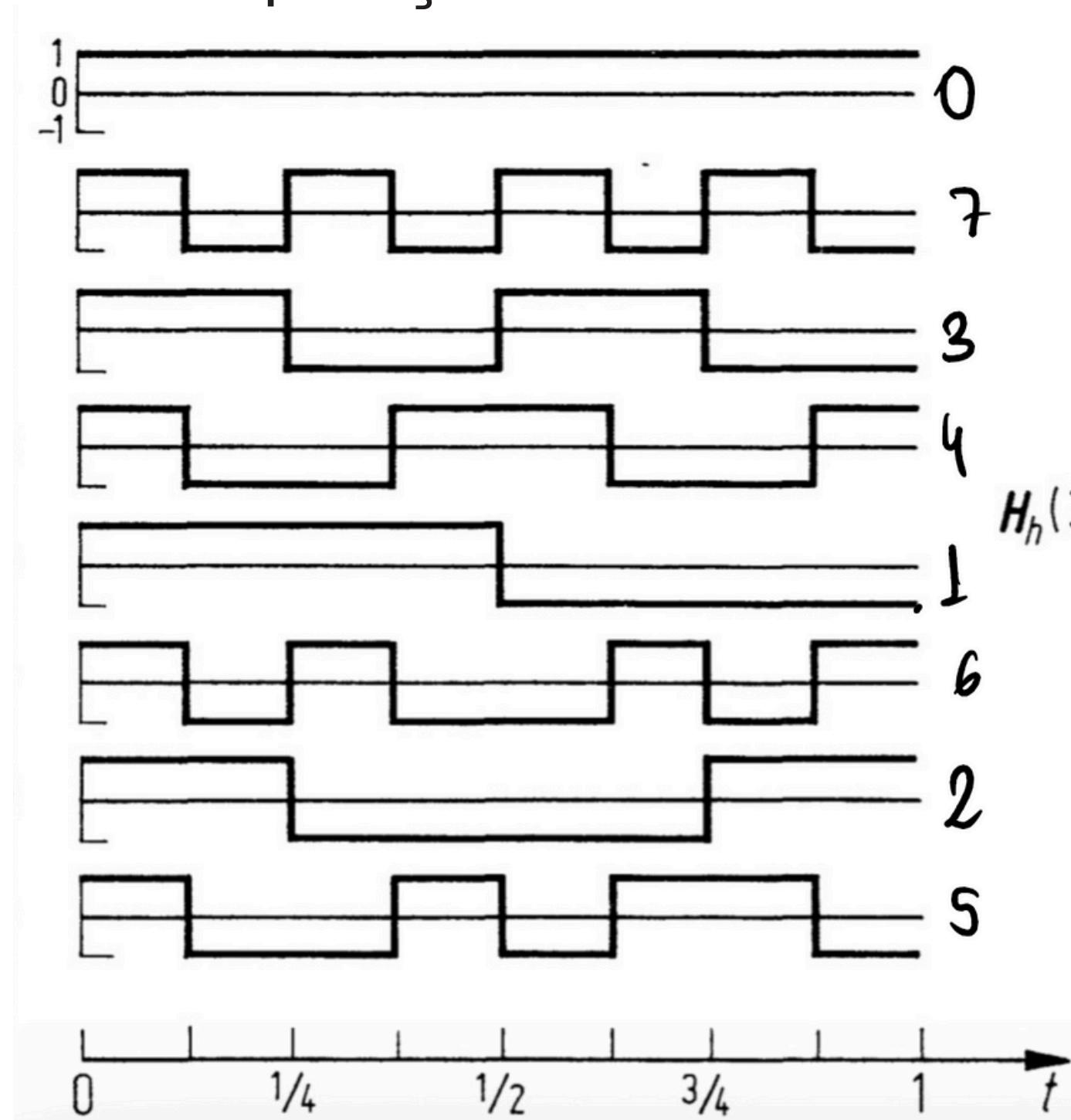
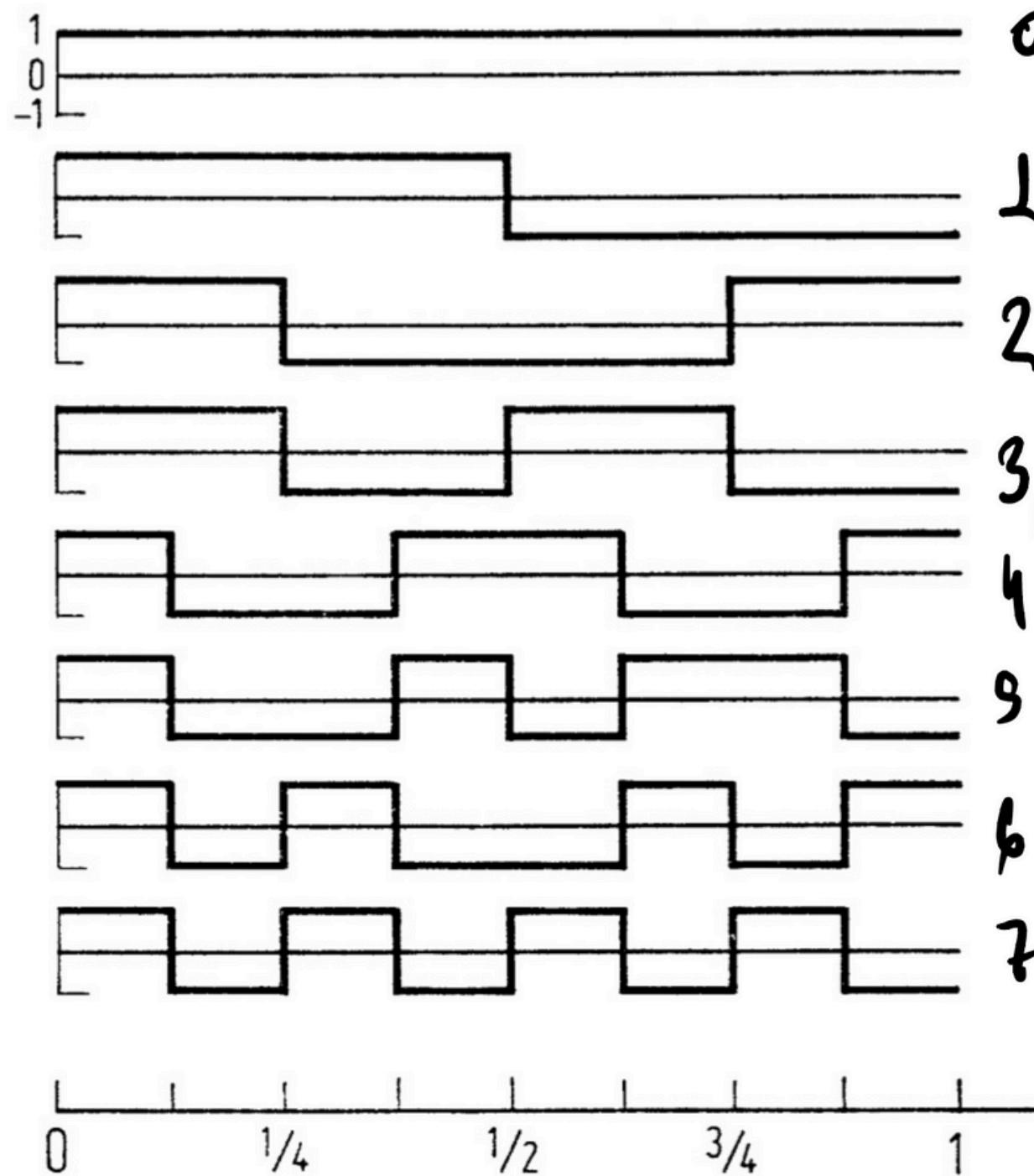
$$H_nH_n^T = \begin{bmatrix} 2^n I & 0 \\ 0 & 2^n I \end{bmatrix}$$

$$H_nH_n^T = 2^n I$$



MATRIZ DE HADAMARD

Contudo, a ordem de sequência está na ordem errada! Em várias aplicações, a ordem é extremamente necessária para a recomposição de sinais.



$$H_h(3) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

10

TRANSFORMADA DE HADAMARD

O WHT de um vetor x de tamanho n ($k = \log_2(n)$) é definido por uma multiplicação matriz-vetor:

$$b = (1/\sqrt{n})H(k)x$$

b = vetor de coeficientes

E por conta das propriedades da Matriz de Hadamard, a volta se dá por:

$$x = (1/\sqrt{n})H(k)b$$

Outra forma de calcular:

$$b = (1/n)H(k)x$$

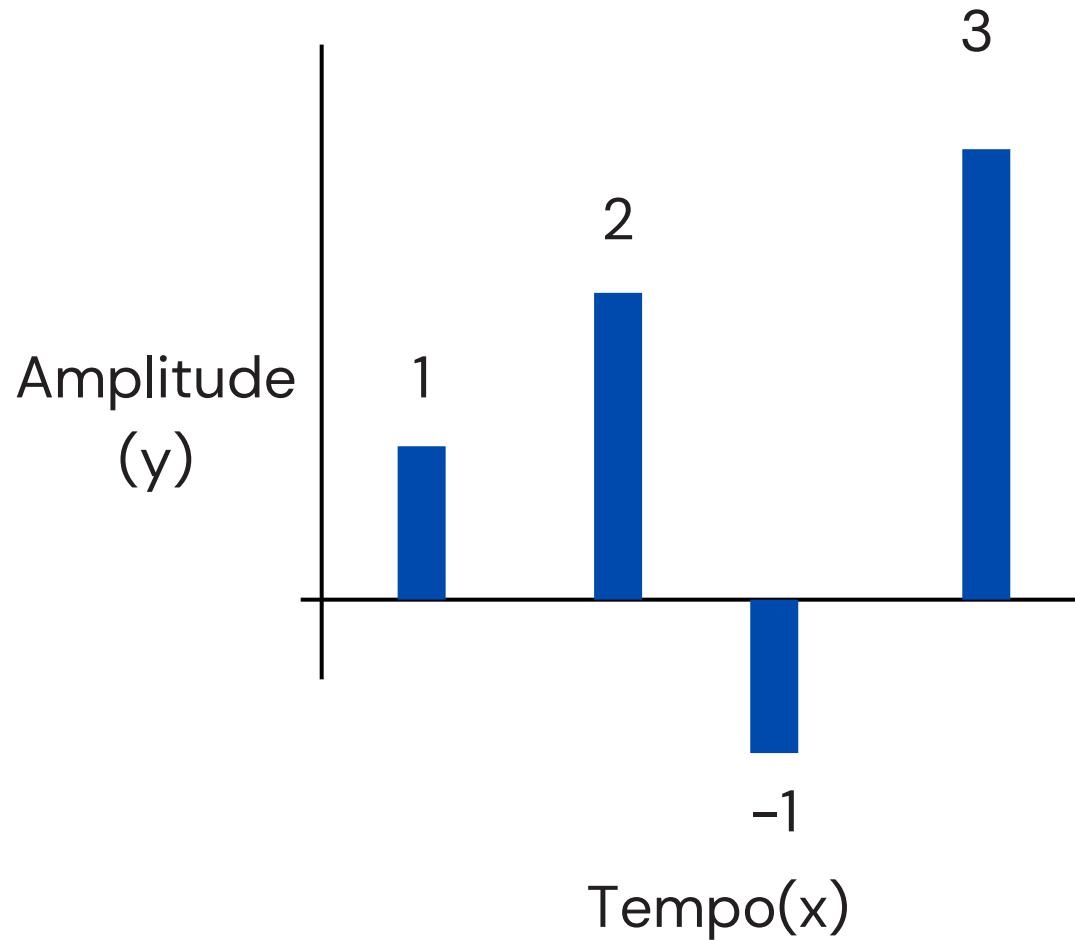
$$x = H(k)b$$

Por conta disso, a Transformada de Walsh-Hadamard possui algumas características muito importantes:

- Complexidade $O(n^2)$
- Usa Apenas Somas e Subtrações
- ? (Veremos adiante)

TRANSFORMADA DE HADAMARD

Exemplo:



$$x = \begin{bmatrix} 1 \\ 2 \\ -1 \\ 3 \end{bmatrix}$$

$$b = (1/n)H(k)x$$

$$b = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ -1 \\ 3 \end{bmatrix}$$

$$\begin{aligned} 1 + 2 - 1 + 3 &= 4b_0 \\ 1 - 2 - 1 - 3 &= 4b_1 \\ 1 + 2 + 1 - 3 &= 4b_2 \\ 1 - 2 + 1 + 3 &= 4b_3 \end{aligned}$$

$$b = \begin{bmatrix} 5/4 \\ -5/4 \\ 1/4 \\ 3/4 \end{bmatrix}$$



$$x = H(k)b$$

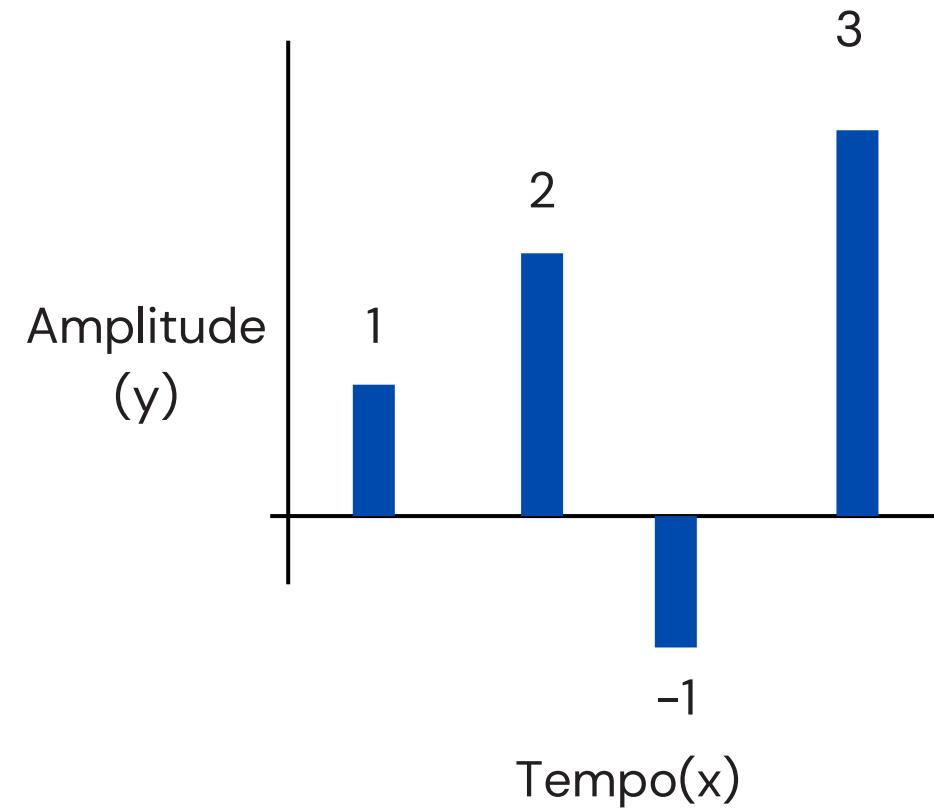
$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 5/4 \\ -5/4 \\ 1/4 \\ 3/4 \end{bmatrix}$$

$$\begin{aligned} 5/4 - 5/4 + 1/4 + 3/4 &= x_0 \\ 5/4 + 5/4 + 1/4 - 3/4 &= x_1 \\ 5/4 - 5/4 - 1/4 - 3/4 &= x_2 \\ 5/4 + 5/4 - 1/4 + 3/4 &= x_3 \end{aligned}$$

$$x = \begin{bmatrix} 1 \\ 2 \\ -1 \\ 3 \end{bmatrix}$$

TRANSFORMADA DE HADAMARD

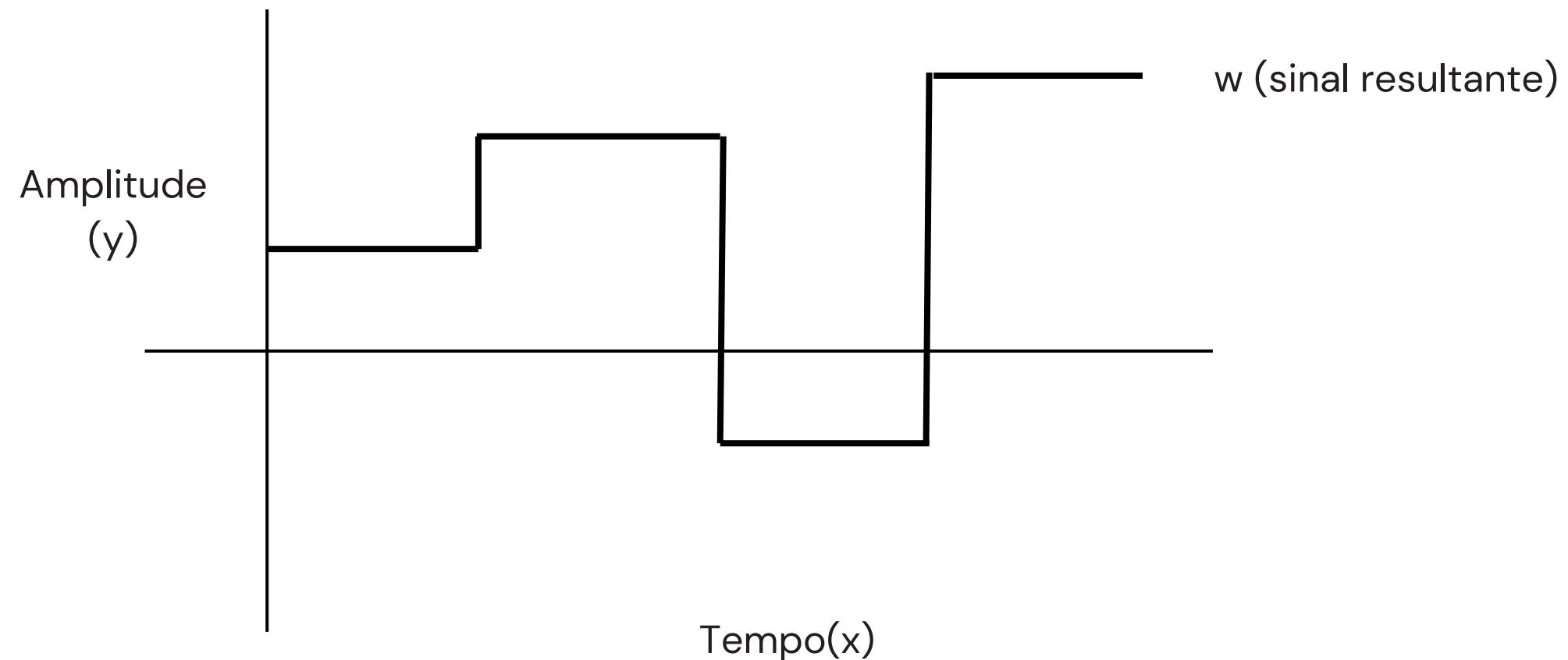
Exemplo:



$$x = \begin{bmatrix} 1 \\ 2 \\ -1 \\ 3 \end{bmatrix} \quad b = \begin{bmatrix} 5/4 \\ -5/4 \\ 1/4 \\ 3/4 \end{bmatrix}$$

$$\begin{aligned} w_0 &= [1 \ 1 \ 1 \ 1] (f = 0) \\ w_1 &= [1 \ -1 \ 1 \ -1] (f = 3) \\ w_2 &= [1 \ 1 \ -1 \ -1] (f = 1) \\ w_3 &= [1 \ -1 \ -1 \ 1] (f = 2) \end{aligned}$$

$$w = (5/4)w_0 + (-5/4)w_1 + (1/4)w_2 + (3/4)w_3$$



TRANSFORMADA DE HADAMARD

Outro Exemplo:

$$x = \begin{bmatrix} 19 \\ -1 \\ 11 \\ -9 \\ 7 \\ 13 \\ -15 \\ 5 \end{bmatrix} \quad b = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \frac{1}{8} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 19 \\ -1 \\ 11 \\ -9 \\ 7 \\ 13 \\ -15 \\ 5 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 4 \\ 0 \\ 3 \\ 10 \\ 0 \\ 0 \end{bmatrix}$$

Somente 4 sinais são não-nulos, logo:

$$w = (2)w_0 + (4)w_2 + (3)w_4 + (10)w_5$$

TRANSFORMADA RÁPIDA

Assim como a Transformada de Fourier, esta também possui sua versão rápida!

Como vimos anteriormente, a complexidade desta transformada até esse momento é $O(n^2)$

Da mesma forma que o FFT, a simetria (e a ortogonalidade) nos mostram que podemos diminuir o número de contas. Estamos repetindo operações desnecessariamente!

Para isso, iremos usar:

- Particionamento de Matrizes

TRANSFORMADA RÁPIDA

Particionamento de Matrizes

Considere um vetor x de tamanho n ($k=\log_2(n)$)

$$H_k x = b \iff \begin{bmatrix} H_{k-1} & H_{k-1} \\ H_{k-1} & -H_{k-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

$$\Updownarrow$$

$$H_{k-1}x_1 + H_{k-1}x_2 = b_1$$

$$H_{k-1}x_1 - H_{k-1}x_2 = b_2$$

$$\Updownarrow$$

$$H_{k-1}(x_1 + x_2) = b_1 \quad (x_1 + x_2) = x'_1$$

$$H_{k-1}(x_1 - x_2) = b_2 \quad (x_1 - x_2) = x'_2$$

TRANSFORMADA RÁPIDA

Pseudocódigo

Recursive function FWHT(x):

$$\begin{bmatrix} x_{1(k/2)} \\ x_{2(k/2)} \end{bmatrix} \leftarrow x_{(k)}$$

$$b'_1 \leftarrow FWHT(x_1)_{(k/2)} \quad O(FWHT(2^{k-1}))$$

$$b'_2 \leftarrow FWHT(x_2)_{(k/2)} \quad O(FWHT(2^{k-1}))$$

$$b_1 \leftarrow \frac{b'_1 + b'_2}{2} \quad O(k/2)$$

$$b_2 \leftarrow \frac{b'_1 - b'_2}{2} \quad O(k/2)$$

$$return \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

TRANSFORMADA RÁPIDA

Complexidade

$$\begin{aligned} O(FWHT(2^n)) &= O(FWHT(2^{n-1})) + O(FWHT(2^{n-1}) + O(n/2) + O(n/2) \\ &= 2O(FWHT(2^{n-1})) + O(n) \end{aligned}$$

Qual é a complexidade desse algoritmo?

$$O(2^n) = 2O(2^{n-1}) + 2^n \quad 2^n = k$$

$$O(k) = 2O(k/2) + k$$

$$O(k) = ?$$

TRANSFORMADA RÁPIDA

Complexidade

Provando que: $O(2^n) = 2O(2^{n-1}) + 2^n = 2^n * n$

Base:

$$n = 1$$

$$O(2^1) = 2^1 * 1 = 2$$

OK

Hipótese de Indução

$$O(2^n) = 2^n * n$$

Indução

$$O(2^{n+1}) = 2O(2^n) + 2^{n+1}$$

$$= 2(2^n * n) + 2^{n+1}$$

$$= 2^{n+1} * n + 2^{n+1}$$

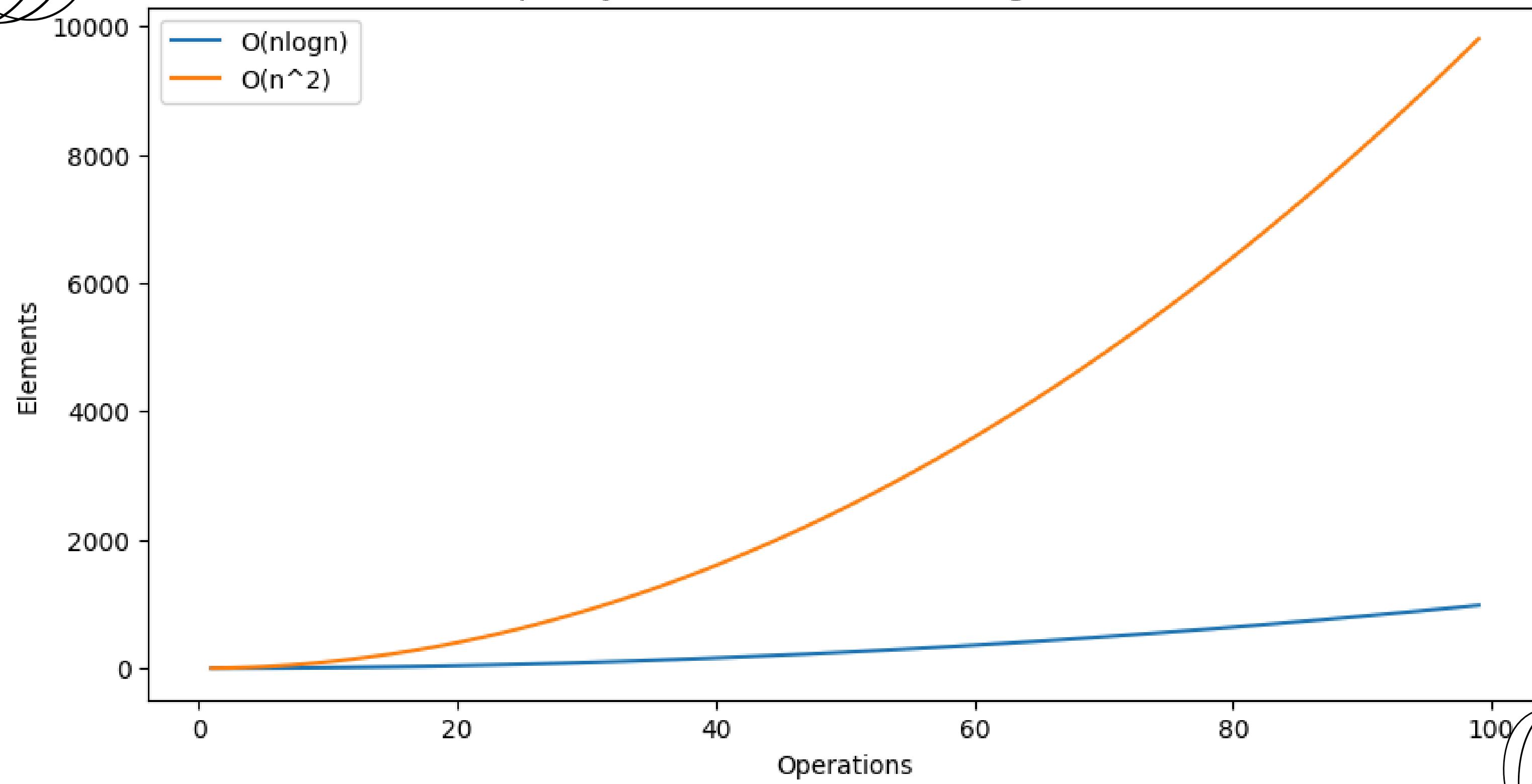
$$= 2^{n+1}(n + 1) \blacksquare$$

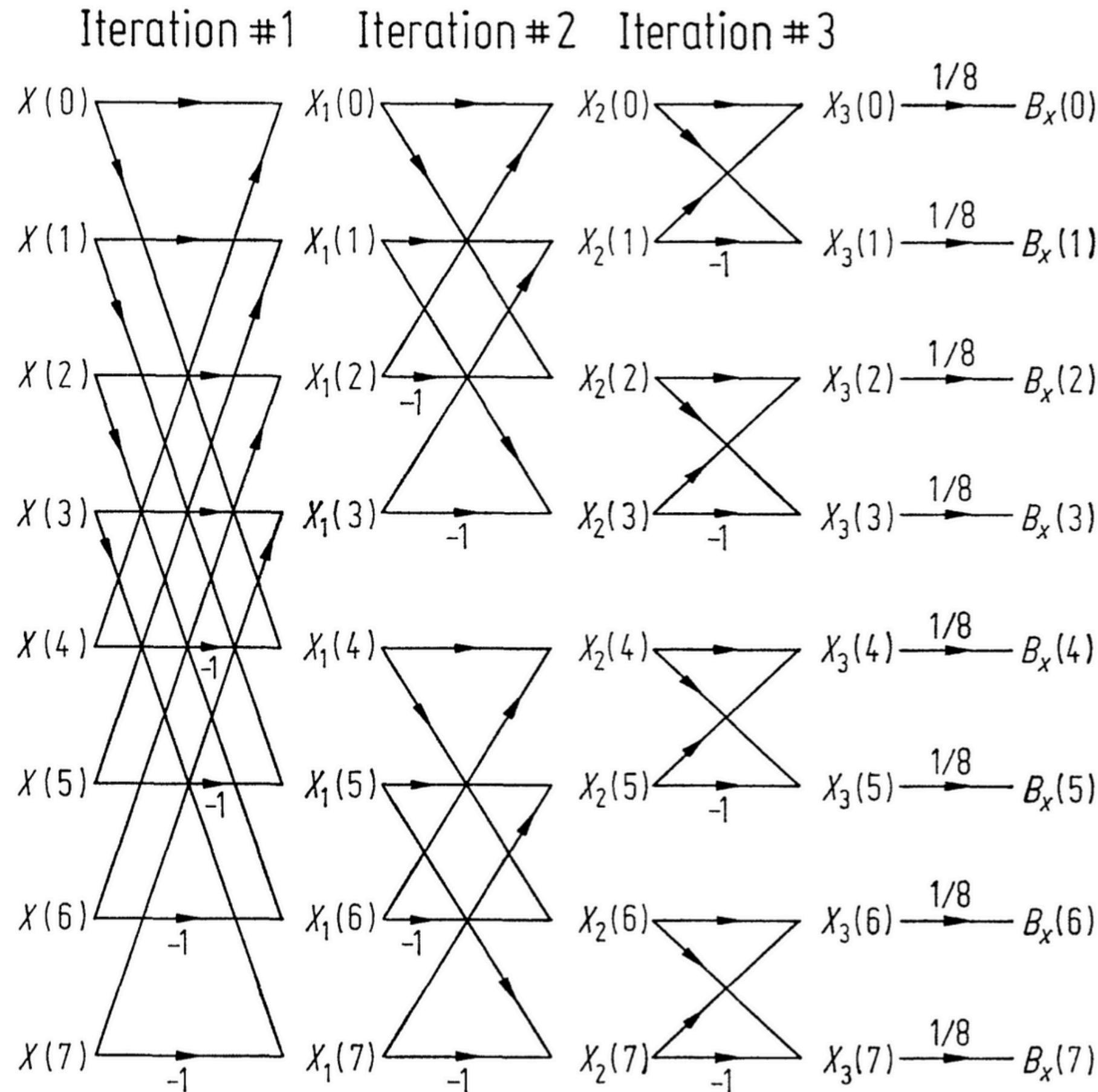
Logo, como: $O(2^n) = 2O(2^{n-1}) + 2^n = 2^n * n$

$$\Updownarrow 2^n = k$$

$$O(k) = 2O(k/2) + k = k * \log_2 k$$

Complexity difference between $O(n\log n)$ and $O(n^2)$

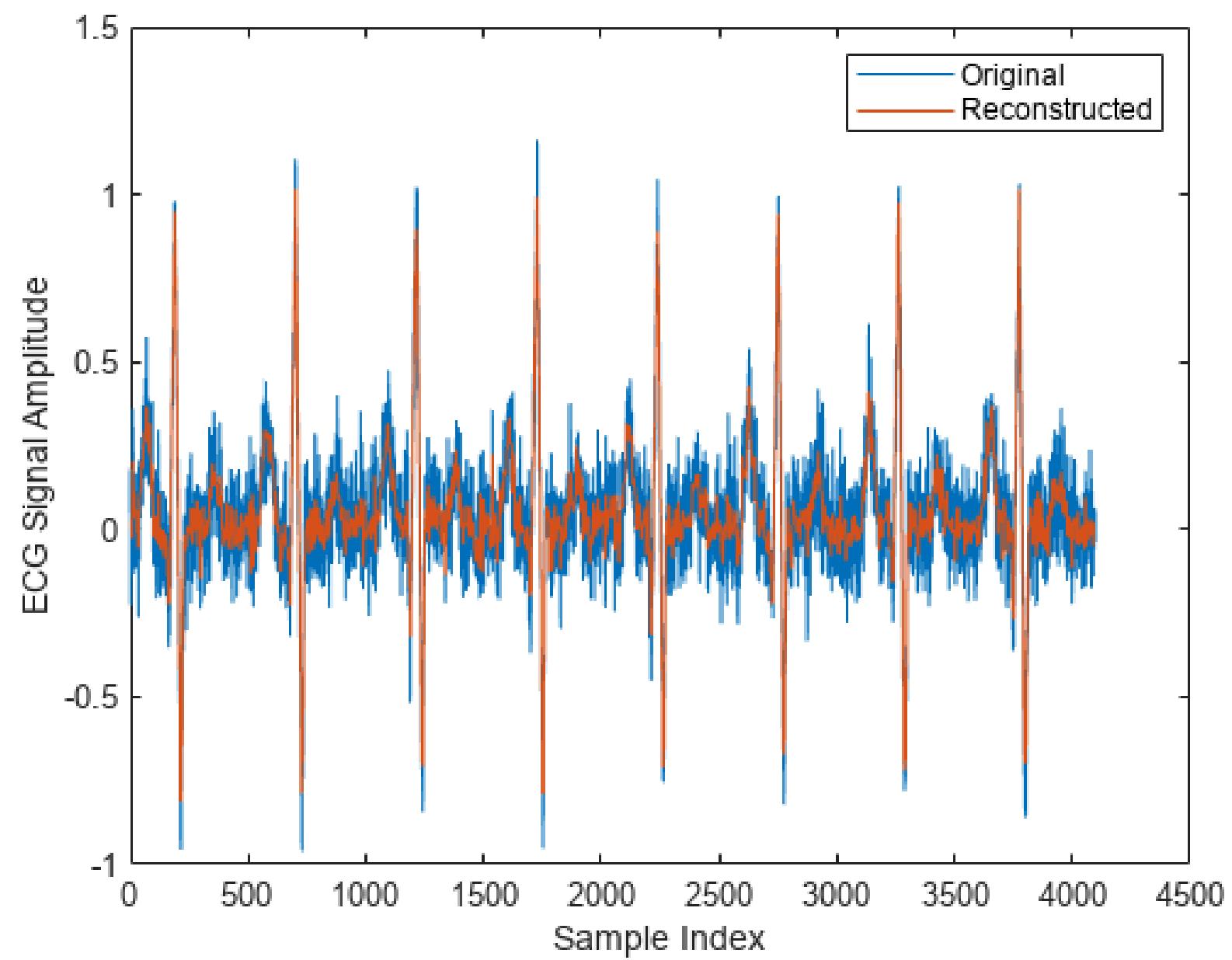
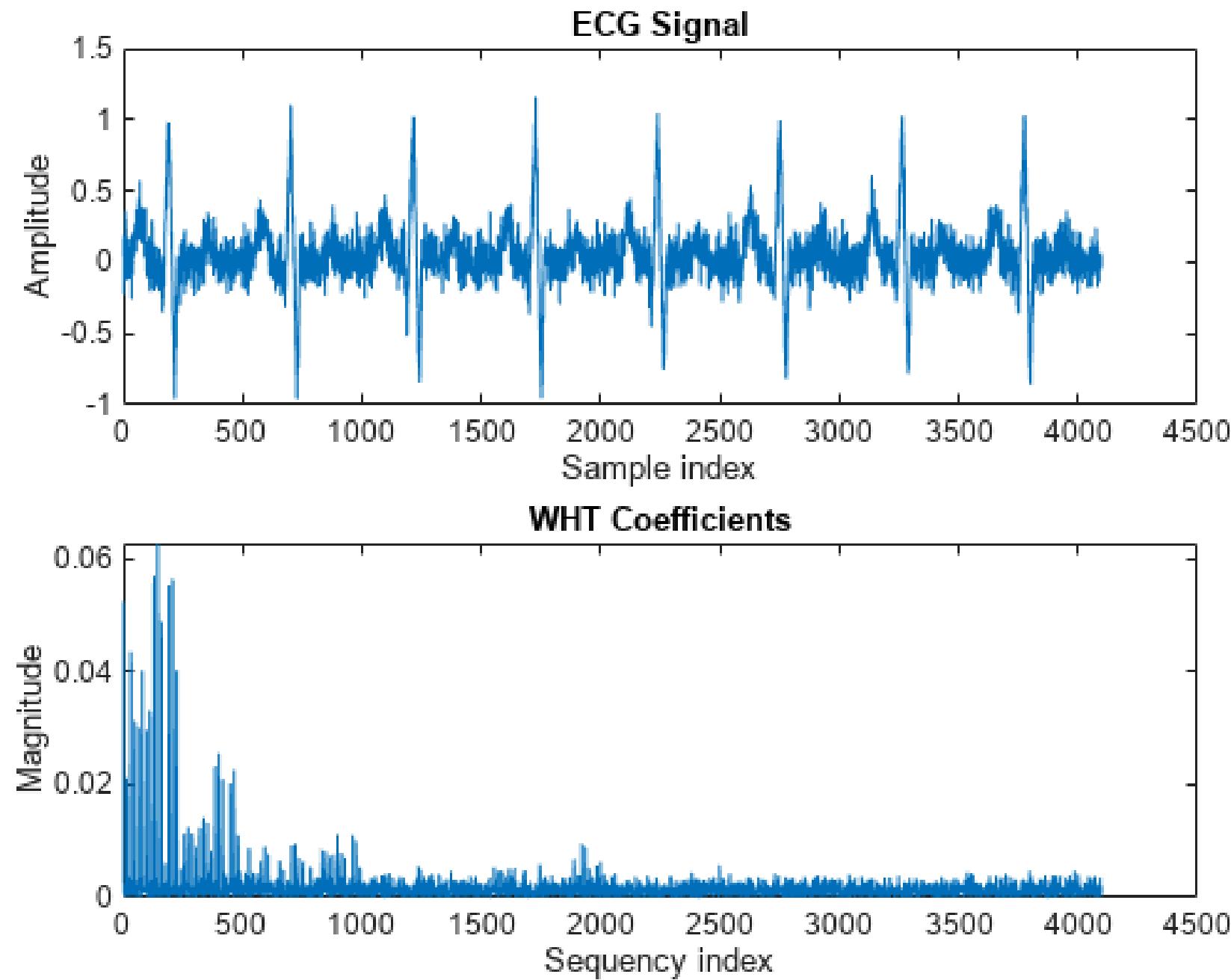




Grafo de um FWHT de $n=8$

APLICAÇÕES

Sinais Ruidosos



REFERÊNCIAS

AHMED, Nasir; RAO, Kamisetty Ramamohan. Orthogonal transforms for digital signal processing. Springer Science & Business Media, 2012.

PRATT, William K.; KANE, Julius; ANDREWS, Harry C. Hadamard transform image coding. Proceedings of the IEEE, v. 57, n. 1, p. 58-68, 1969.

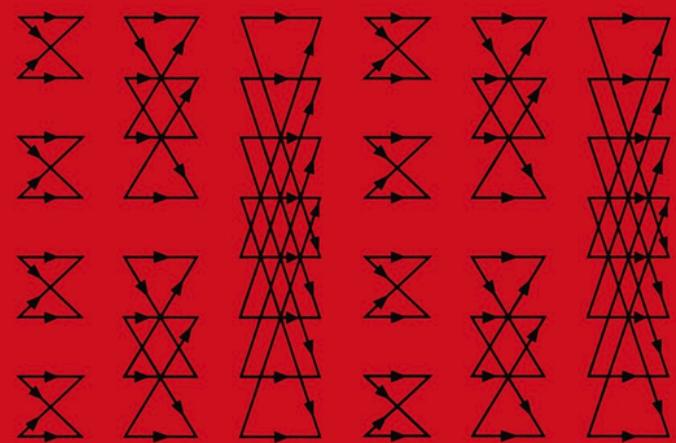
Lee, M., & Kaveh, M. (1986). Fast Hadamard transform based on a simple matrix factorization. IEEE transactions on acoustics, speech, and signal processing, 34(6), 1666-1667.

Whelchel, E., & Quinn, F. (1968). The Fast Fourier-Hadamard Transform and Its Use in Signal Representation and Classification. Defense Technical Information Center

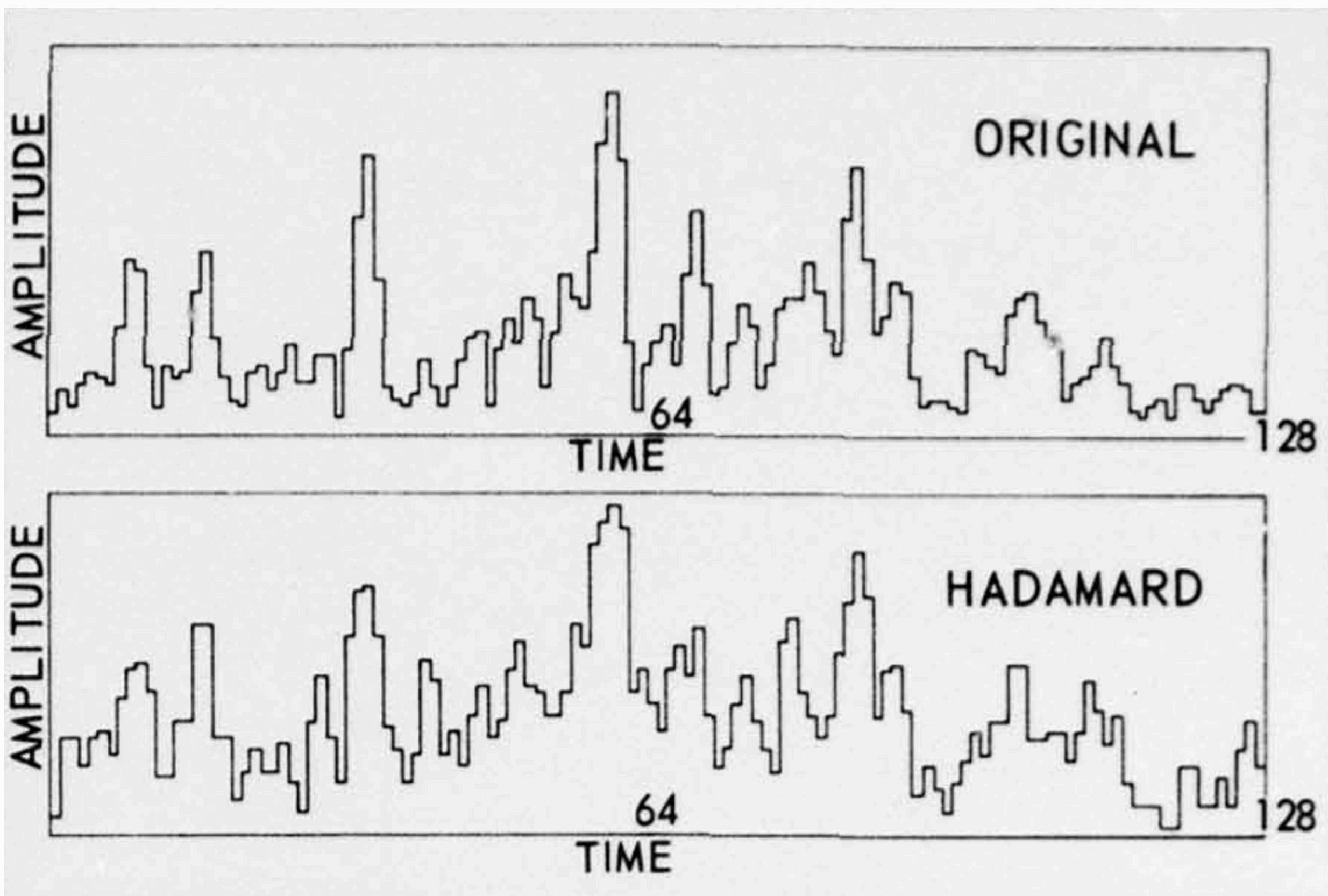
Taboga, Marco (2021). "Properties of the Kronecker product", Lectures on matrix algebra.

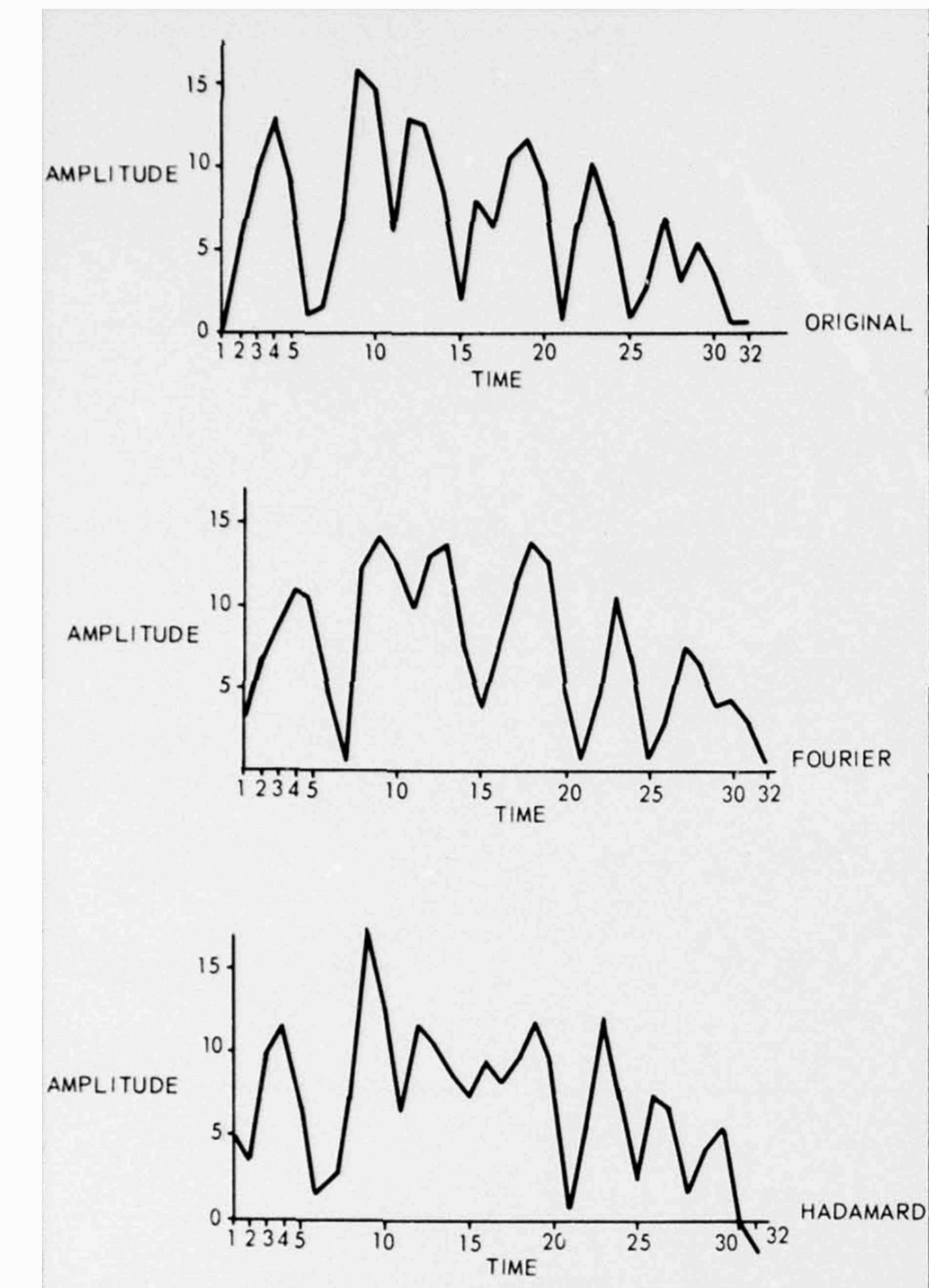
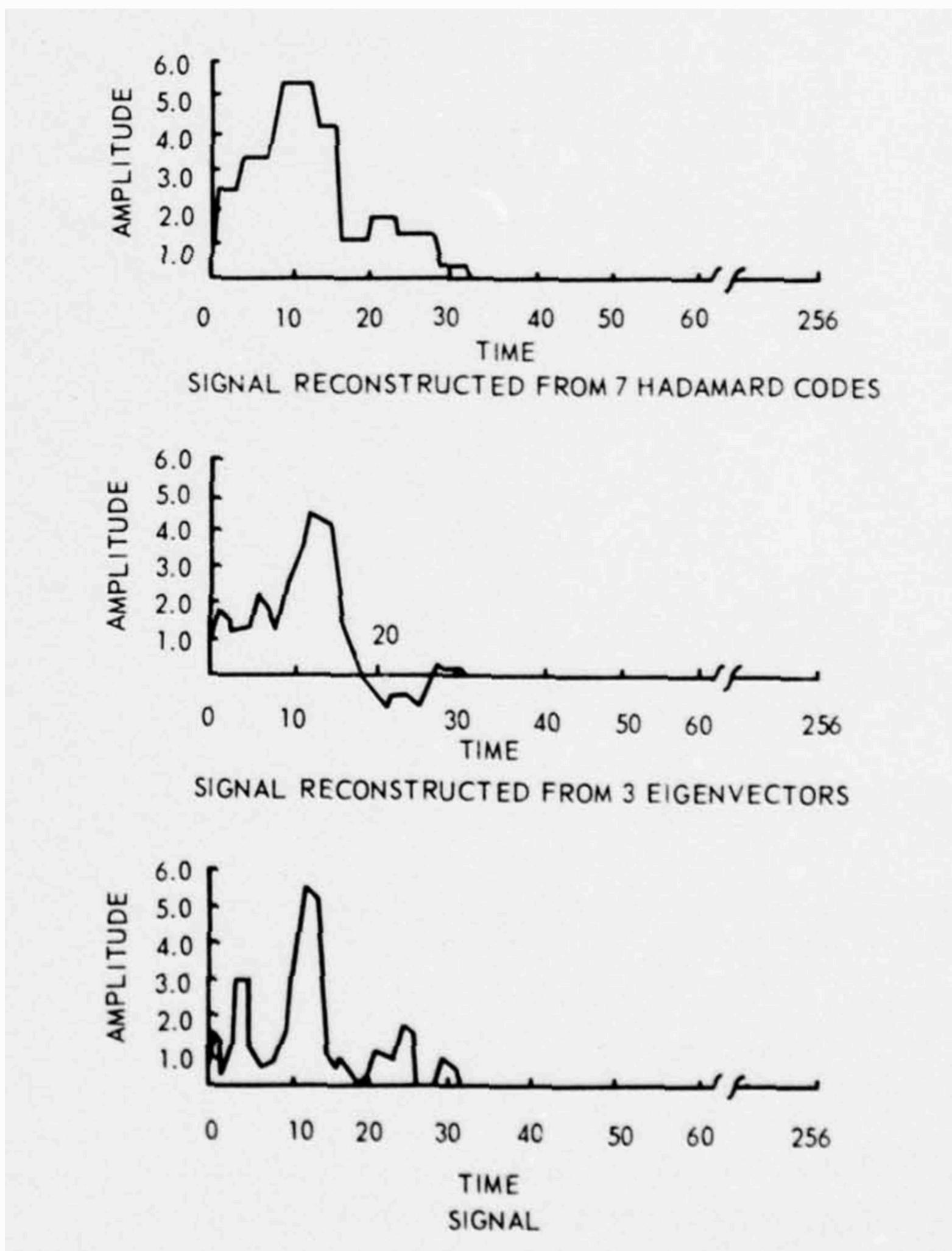
N. Ahmed · K. R. Rao

Orthogonal Transforms
for Digital
Signal Processing



Springer-Verlag
Berlin · Heidelberg · New York





ÁLGEBRA LINEAR APLICADA
OBRIGADO!

**TRANSFORMADA
DE
HADAMARD**

MANOEL SILVA