Lecture 12

Discrete and Fast Fourier Transforms

12.1 Introduction

The goal of the chapter is to study the Discrete Fourier Transform (DFT) and the Fast Fourier Transform (FFT). In the course of the chapter we will see several similarities between Fourier series and wavelets, namely

- Orthonormal bases make it simple to calculate coefficients,
- Algebraic relations allow for fast transform, and
- Complete bases allow for arbitrarily precise approximations.

There is, however, a very important difference between Fourier series and wavelets, namely

Wavelets have compact support, Fourier series do not.

12.2 The Discrete Fourier Transform (DFT)

12.2.1 Definition and Inversion

Let $\vec{\mathbf{e}}_0, \ldots, \vec{\mathbf{e}}_{N-1}$ denote the usual standard basis for $\mathbb{C}^n$. A vector $\vec{\mathbf{f}} = (f_0, \ldots, f_{N-1}) \in \mathbb{C}^N$ may then be written as $\vec{\mathbf{f}} = f_0\vec{\mathbf{e}}_0 + \cdots + f_{N-1}\vec{\mathbf{e}}_{n-1}$.

Important example: Assume the array $\vec{\mathbf{f}}$ is a sample from a function $f : \mathbb{R} \to \mathbb{C}$, that is, we use the sample points $x_0 := 0, \ldots, x_\ell := \ell \cdot (2\pi/N), \ldots, x_{N_1} := (N-1)\cdot(2\pi/N)$ with values $\vec{\mathbf{f}} = (f(x_0), \ldots, f(x_\ell), \ldots, f(x_{N-1}))$.

The Discrete Fourier Transform expresses such an array $\vec{\mathbf{f}}$ with linear combinations of arrays of the type

$$\begin{aligned}\vec{\mathbf{w}}_k := \left(e^{ikx_\ell}\right)_{\ell=0}^{N-1} &= \left(1, e^{ik2\pi/N}, \ldots, e^{ik\ell\cdot 2\pi/N}, \ldots, e^{ik(N-1)\cdot 2\pi/N}\right) \\ &= (\vec{\mathbf{w}}_k)_\ell = \left(e^{i\cdot 2\pi/N}\right)^{k\ell} =: \omega_N^{k\ell}.\end{aligned}$$

Definition For each positive integer N, we define an inner product on $\mathbb{C}^N$ by

$$\langle \vec{z}, \vec{\mathbf{w}} \rangle_N = \frac{1}{N} \sum_{m=0}^{N-1} z_m \cdot \overline{w_m}.$$

Lemma For each positive integer N, the set

$$\{\vec{\mathbf{w}}_k \,|\, k \in \{0, \ldots, N-1\}\}$$

is orthonormal with respect to the inner product $\langle \cdot, \cdot \rangle_N$.

In fact $\{\vec{w}_0, \ldots, \vec{w}_{N-1}\}$ is an *orthonormal basis* for $\mathbb{C}^N$.

Sketch of Proof For all $k, \ell \in \mathbb{Z}$ so that $\ell = k + JN$ for some J, we have

$$\begin{aligned}\langle \vec{w}_k, \vec{w}_\ell \rangle_N = \frac{1}{N} \sum_{m=0}^{N-1} (\vec{w}_k)_m \overline{(\vec{w}_\ell)_m} &= \frac{1}{N} \sum_{m=0}^{N-1} e^{ikm\cdot 2\pi/N} e^{-i\ell m\cdot 2\pi/N} \\ = \frac{1}{N} \sum_{m=0}^{N-1} e^{ikm\cdot 2\pi/N} e^{-i(k+JN)m\cdot 2\pi/N} &= \frac{1}{N} \sum_{m=0}^{N-1} 1 = 1.\end{aligned}$$

For the remaining $k, \ell \in \mathbb{Z}$ we use the geometric series to see that

$$\begin{aligned}\langle \vec{\mathbf{w}}_k, \vec{\mathbf{w}}_\ell \rangle_N = \frac{1}{N} \sum_{m=0}^{N-1} (\vec{\mathbf{w}}_k)_m \overline{(\vec{\mathbf{w}}_\ell)_m} &= \frac{1}{N} \sum_{m=0}^{N-1} e^{i[k-\ell]m\cdot 2\pi/N} \\ = \frac{1}{N} \sum_{m=0}^{N-1} \left(e^{i[k-\ell]\cdot 2\pi/N}\right)^m &= \frac{1}{N} \frac{1 - (e^{i[k-\ell]m\cdot 2\pi/N})^N}{1 - e^{i[k-\ell]\cdot 2\pi/N}} \\ = \frac{1}{N} \frac{1 - (e^{2\pi i})^{[k-\ell]}}{1 - e^{i[k-\ell]\cdot 2\pi/N}} &= \frac{1}{N} \frac{1-1}{1 - e^{i[k-\ell]\cdot 2\pi/N}} = 0.\end{aligned}$$

12.2.2 The Fourier matrix

Definition For each positive integer N, define the *Fourier matrix* ${}_F^N\Omega$ by

$$ {}_F^N\Omega_{k,\ell} = (\vec{\mathbf{w}}_\ell)_k = e^{ik\ell 2\pi/N} = \omega_N^{k\ell}. $$

Example: If $N = 1$, then $\omega_N = \omega_1 = 1$, and ${}_F^N\Omega = 1$. If $N = 2$, then $\omega_N = \omega_2 = -1$, and

$$ {}_F^2\Omega = \begin{pmatrix} (\omega_2^0)^0 & (\omega_2^1)^0 \\ (\omega_2^0)^1 & (\omega_2^1)^1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad \text{(the Haar matrix)} $$

If $N = 4$, then $\omega_N = \omega_4 = e^{i2\pi/4} = i$, and

$$ {}_F^4\Omega = \begin{pmatrix} 1 & (i^1)^0 & (i^2)^0 & (i^3)^0 \\ 1 & (i^1)^1 & (i^2)^1 & (i^3)^1 \\ 1 & (i^1)^2 & (i^2)^2 & (i^3)^2 \\ 1 & (i^1)^3 & (i^2)^3 & (i^3)^3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}. $$

12.3 Discrete Fourier Transform

Definition For each positive integer N and each array $\vec{f} \in \mathbb{C}^N$, the *Discrete Fourier Transform* of $\vec{f}$ is the array $\hat{\mathbf{f}}$ defined by

$$ \hat{f}_k = \langle \vec{f}, \vec{\mathbf{w}}_k \rangle_N = \frac{1}{N} \sum_{m=0}^{N-1} f_m \cdot e^{-ikm\cdot 2\pi/N}. $$

It is very important to understand this definition. The left hand side is simply the *orthogonal projection of* $\vec{\mathbf{f}}$ onto the basis vector $\vec{\mathbf{w}}_k$.

Since the DFT consists of the coefficients of $\vec{\mathbf{f}}$ expressed with respect to the new basis $(\vec{\mathbf{w}}_k)_{k=0}^{N-1}$, the DFT simply "rotates" the coordinates in $\mathbb{C}^N$.

12.3.1 Two Results

Proposition The DFT corresponds to a multiplication by the transposed conjugate matrix $\frac{1}{N}\overline{{}_F^N\Omega^T}$.

Proof: Simply note that

$$\hat{f}_{N,k} = \langle \vec{\mathbf{f}}, \vec{\mathbf{w}}_k \rangle_N = \frac{1}{N} \sum_{m=0}^{N-1} f_\ell e^{-ik\ell \cdot 2\pi/N}$$
$$= \frac{1}{N} \sum_{m=0}^{N-1} \overline{{}_F^N\Omega^T} \cdot f_\ell = \frac{1}{N} \cdot \left(\overline{{}_F^N\Omega^T} \cdot \vec{\mathbf{f}} \right)_k.$$

Proposition 13 For each positive integer N and each array $\vec{\mathbf{f}} \in \mathbb{C}^N$, we have the following inversion formulas:

$$\vec{\mathbf{f}} = \sum_{m=0}^{N-1} \hat{f}_k \vec{\mathbf{w}}_k \quad , \quad \hat{f}_k = \langle \vec{\mathbf{f}}, \vec{\mathbf{w}}_k \rangle_N = \frac{1}{N} \sum_{m=0}^{N-1} f_m \cdot e^{-ikm \cdot 2\pi/N}.$$

One proof The easiest way to prove Proposition 13 is to note that $(\vec{\mathbf{w}}_k)_{k=0}^{N-1}$ forms an orthonormal basis for $\mathbb{C}^N$. Therefore, the formula

$$\vec{\mathbf{f}} = \sum_{k=0}^{N-1} \hat{f}_k \vec{\mathbf{w}}_k = \sum_{k=0}^{N-1} \langle \vec{\mathbf{f}}, \vec{\mathbf{w}}_k \rangle_N \, \vec{\mathbf{w}}_k$$

represents the orthogonal projection of $\vec{\mathbf{f}}$ on $\mathbb{C}^N$. But the projection of $\vec{\mathbf{f}}$ on $\mathbb{C}^N$ **is** $\vec{\mathbf{f}}$, since $\vec{\mathbf{f}}$ already lies in the range of the projection (the range being $\mathbb{C}^N$).

Example With $N = 4$, the fourth root of unity is $\omega_N = \omega_4 = i$. The N arrays $\vec{\mathbf{w}}_k$ thus become

$$\vec{\mathbf{w}}_0 = \left([i]^{0\cdot 0}, [i]^{0\cdot 1}, [i]^{0\cdot 2}, [i]^{0\cdot 3} \right) = (1, 1, 1, 1),$$
$$\vec{\mathbf{w}}_1 = \left([i]^{1\cdot 0}, [i]^{1\cdot 1}, [i]^{1\cdot 2}, [i]^{1\cdot 3} \right) = (1, i, -1, -i),$$
$$\vec{\mathbf{w}}_2 = \left([i]^{2\cdot 0}, [i]^{2\cdot 1}, [i]^{2\cdot 2}, [i]^{2\cdot 3} \right) = (1, -1, 1, -1),$$
$$\vec{\mathbf{w}}_3 = \left([i]^{3\cdot 0}, [i]^{3\cdot 1}, [i]^{3\cdot 2}, [i]^{3\cdot 3} \right) = (1, -i, -1, i).$$

Assume $\vec{\mathbf{f}} = (f_0, f_1, f_2, f_3) = (9, 7, 5, 7)$. Then (remember the complex conjugation!):

$$\hat{f}_0 = \langle \vec{\mathbf{f}}, \vec{\mathbf{w}}_0 \rangle_N = \langle (9, 7, 5, 7), (1, 1, 1, 1) \rangle_4 = 7,$$
$$\hat{f}_1 = \langle \vec{\mathbf{f}}, \vec{\mathbf{w}}_1 \rangle_N = \langle (9, 7, 5, 7), (1, i, -1, -i) \rangle_4 = 1,$$
$$\hat{f}_2 = \langle \vec{\mathbf{f}}, \vec{\mathbf{w}}_2 \rangle_N = \langle (9, 7, 5, 7), (1, -1, 1, -1) \rangle_4 = 0,$$
$$\hat{f}_3 = \langle \vec{\mathbf{f}}, \vec{\mathbf{w}}_3 \rangle_N = \langle (9, 7, 5, 7), (1, -i, -1, i) \rangle_4 = 1.$$

Therefore

$$\begin{aligned}\vec{\mathbf{f}} &= (9,7,5,7)\\ &= 7\cdot(1,1,1,1)+1\cdot(1,i,-1,-i)\\ &\quad +0\cdot(1,-1,1,-1)+1\cdot(1,-i,-1,i)\end{aligned}$$

is the Inverse Discrete Fourier Transform of the array $\vec{\mathbf{f}} = (9,7,5,7)$.

12.4 Unitary Operators

We have just seen that the Discrete Fourier Transform is a linear operator

$$DFT : \mathbb{C}^N \to \mathbb{C}^N \quad , \quad \vec{\mathbf{f}} \mapsto \widehat{\mathbf{f}} = \frac{1}{N}\overline{{}_F^N\Omega^T}\vec{\mathbf{f}}.$$

Its inverse is given by

$$DFT^{-1} : \mathbb{C}^N \to \mathbb{C}^N \quad , \quad \widehat{\mathbf{f}} \mapsto \vec{\mathbf{f}} = {}_F^N\Omega\vec{\mathbf{f}}.$$

Using the multiplicative constant $1/\sqrt{N}$ instead of $1/N$, the Discrete Fourier Transform thus is multiplication by the matrix ${}_F^NU := \frac{1}{\sqrt{N}}\overline{{}_F^N\Omega^T}$.

In the exercises you are asked to show that $({}_F^NU)^{-1} = \overline{({}_F^NU)}^T$. Such an operator has a special name:

Definition 17 For each linear space V with an inner product $\langle\cdot,\cdot\rangle$, a linear operator $L : V \to V$ is **unitary** if

$$\langle Lv, Lw\rangle = \langle v, w\rangle$$

for all $v, w \in V$.

12.5 The Fast Fourier Transform

12.5.1 Introduction

We have seen how to convert a sample $\vec{\mathbf{f}} = (f_0,\ldots,f_{N-1})$ to a frequency sample $\vec{\hat{\mathbf{f}}} = (\hat{f}_0,\ldots,\hat{f}_{N-1})$ and back again. But there is an aspect we haven't touched upon yet, namely

How long does it take to perform these calculations?

Let us try to estimate the number of calculations that are required. So let us start with N complex numbers $(f_0, \ldots, f_{N-1})$.

- Each number $f_j = \mathrm{Re} f_j + i\mathrm{Im} f_j$ is being multiplied by $e^{-2\pi ijn/N} = \cos\frac{2\pi ijn}{N} - i\sin\frac{2\pi ijn}{N}$; this gives $\boxed{4N}$ operations.
- This has to be done for each of the numbers f_j, totalling $4N^2$ real multiplications, or N^2 complex multiplications.
- The numbers should then be added, but that would merely amount to $2N$ operations. Our number of operations is thus $\boxed{4N^2}$.

So how big is this number?

An Example Assume we need one minute to compute the Fourier transform for a sequence with 4 samples. It would therefore take us approximately $\frac{60}{4\times 4^2} = \frac{60}{64} \approx 0.94$sec per operation.

- With $N = 8$ we would need $\frac{4\times 8^2\times 15}{16\times 60} = 4$ min,
- With $N = 16$ we would need $\frac{4\times 16^2\times 15}{16\times 60} = 16$ min,
- With $N = 32$ we would need $\frac{4\times 32^2\times 15}{16\times 60} = 64$ min, and
- With $N = 2^8 = 256$ we would need $\frac{4\times 256^2\times 15}{16\times 60} = 4096$ min, (almost three days).
- A standard TV need roughly 10000000 pixel values every second to preserve relevant information. This would take us around 118900256 years (!!!) to analyze all this information.

12.5.2 The Forward FFT

The point about the Fast Fourier Transform (FFT) is that it gives the same coefficients as the Discrete Fourier Transform (DFT) but requires less calculations. For each integer N of the form $N = 2^k$ for some k and for each array

$$\vec{\mathbf{f}} = (f_0, \ldots, f_{N-1}) = (f(0), \ldots, f(N-1)) \in \mathbb{C}^N,$$

we define two arrays

$$\begin{aligned} {}_{\text{even}}\vec{\mathbf{f}} &= (f(0), f(2), \ldots, f(2j), \ldots, f(N-2)) \text{ and} \\ {}_{\text{odd}}\vec{\mathbf{f}} &= (f(1), f(3), \ldots, f(2j+1), \ldots, f(N-1)). \end{aligned}$$

We want to know how the DFT of ${}_{\text{even}}\vec{\mathbf{f}}$ and the DFT of ${}_{\text{odd}}\vec{\mathbf{f}}$ are related to the DFT of $\vec{\mathbf{f}}$. The answer is given in the next result:

Central result

Lemma 19 For each $k \in \{0, 1, \ldots, \frac{N}{2}-1\}$ and all arrays $\vec{\mathbf{f}} = (f(0), \ldots, f(N-1)) \in \mathbb{C}^N$

$$\begin{aligned} \hat{f}_k &= \frac{1}{2} \cdot \left({}_{\text{even}}\hat{f}_k + [e^{-i\cdot 2\pi/N}]^k \cdot [{}_{\text{odd}}\hat{f}_k] \right), \\ \hat{f}_{k+\frac{N}{2}} &= \frac{1}{2} \cdot \left({}_{\text{even}}\hat{f}_k - [e^{-i\cdot 2\pi/N}]^k \cdot [{}_{\text{odd}}\hat{f}_k] \right). \end{aligned}$$

We will not give a proof but merely see what the Lemma says in the case $N = 2$. Then $N/2 = 1$ and $\vec{\mathbf{f}} = (f_0, f_1)$. Thus ${}_{\text{even}}\vec{\mathbf{f}} = (f_0)$ and ${}_{\text{odd}}\vec{\mathbf{f}} = (f_1)$. Using that $e^{-i\cdot 2\pi/2} = -1$, the Lemma simply says that

$$\hat{f}_0 = \frac{f_0 + f_1}{2} \quad \text{and} \quad \hat{f}_1 = \frac{f_0 - f_1}{2}.$$

Time required?

We will only count *complex* operations.

- We start with an array $\vec{\mathbf{f}} = (f_0, \ldots, f_{N-1})$ where $N = 2^k$.
- The DFT needed approximately N^2 complex multiplications.

- Using the FFT (where we decompose $\vec{\mathbf{f}}$ into two smaller arrays, divide each of these into two smaller arrays, and so on), we end up with k arrays each of length 2.

We thus need around kN complex multiplications instead of N^2.

Is this significant?

In particular, taking $N = 2^{10} \approx 1.024 \times 10^3$ and calculate the time, the DFT takes

$$\frac{4 \times 2^{20} \times 15}{16 \times 60 \times 60 \times 24 \times 365} \approx 1247 \text{ years}$$

but the FFT only takes

$$\frac{4 \times 10 \times 2^{10} \times 15}{16 \times 60 \times 60} \approx 10.67 \text{ min.}$$

Obviously this is a very distinct reduction in the time required to perform the calculations :-)

12.5.3 The Inverse FFT (IFFT)

For each integer N, the FFT admits an inverse map that we call the Inverse Fast Fourier Transform (IFFT). Starting with an array $\hat{\mathbf{f}} \in \mathbb{C}^N$ we thus contruct the array of data $\vec{\mathbf{f}} \in \mathbb{C}^N$ such that $\hat{\mathbf{f}}$ is the DFT of $\vec{\mathbf{f}}$.

Using that $\vec{\mathbf{f}} = \hat{f}_0\vec{\mathbf{w}}_0 + \cdots + \hat{f}_k\vec{\mathbf{w}}_k + \cdots + \hat{f}_{N-1}\vec{\mathbf{w}}_{N-1}$ we thus get for the coordinates with index ℓ that

$$f_\ell = \sum_{k=0}^{N-1} \hat{f}_k(\vec{\mathbf{w}}_k)_\ell = \sum_{k=0}^{N-1} \hat{f}_k e^{ik\ell\cdot 2\pi/N}.$$

We may interpret this formula as yet another FFT:

$$f_\ell = \sum_{k=0}^{N-1} \hat{f}_k e^{ik\ell\cdot 2\pi/N} = \overline{\sum_{k=0}^{N-1} \overline{\hat{f}_k}\overline{e^{ik\ell\cdot 2\pi/N}}}$$
$$= N \cdot \overline{\frac{1}{N}\sum_{k=0}^{N-1} \overline{\check{f}_k} e^{-ik\ell k\cdot 2\pi/N}} = N \cdot \overline{\hat{\check{f}}_\ell}$$

So in order to calculate the data $\vec{\mathbf{f}}$ from $\hat{\mathbf{f}}$, it suffices to form the complex conjugate of $\hat{\mathbf{f}}$, take its FFT multiplied by N, and then take the complex conjugate one more time.

12.5.4 Interpolation by the IFFT

Thus far we have used the sample points $x_\ell = \ell \cdot \frac{2\pi}{N}$, but the IFFT provides a way to interpolate the function f in *other* points. To see how it works, we recall that (by definition of the DFT),

$$f(x_\ell) = \sum_{k=0}^{N-1} \hat{f}_k e^{ikx_\ell}.$$

One way to interpolate f at x is therefore to approximate $f(x)$ by

$$f(x) \approx \sum_{k=0}^{N-1} \hat{f}_k e^{ikx}.$$

However, if we consider several values of x spaced by multiples of $\frac{2\pi}{N}$, the Fast Fourier Transform is a more convenient tool. So let $u = x - x_\ell$ for any ℓ. Thus

$$\begin{aligned} f(x) &\approx \sum_{k=0}^{N_1} \hat{f}_k e^{ikx} = \sum_{k=0}^{N-1} \hat{f}_k e^{ik[u+x_\ell]} \\ &= \sum_{k=0}^{N-1} \left(\hat{f}_k e^{iku}\right) e^{ikx_\ell} = \sum_{k=0}^{N-1} \left(\hat{f}_k e^{iku}\right) e^{ik\ell \cdot 2\pi/N}, \end{aligned}$$

where the sums thus obtained are just the IFFT applied to the coefficients $(\hat{f}_k e^{iku})_{k=0}^{N-1}$. Therefore, to interpolate f at the points $x = u + \ell \cdot \frac{2\pi}{N}$, $\ell \in \{0, \ldots, N-1\}$, it suffices to multiply each coefficient $\hat{f}_k$ by $e^{iku} = (e^{iu})^k$ and then calculate the IFFT of the array $(\hat{f}_k e^{iku})_{k=0}^{N-1}$.

12.5.5 Bit Reversal

One way to better facilitate a recursive calculation of the Fast Fourier Transform is to rearrange the initial date in such a way that the Fast Fourier Transform operates on adjacent pairs of arrays in each step of the recursion. More precisely:

Definition 23 *Bit reversal* transforms a finite sequence $(p_{k-1}, p_{k-2}, \ldots, p_1, p_0)$ of k (binary) integers $p_j \in \{0, 1\}$ into the *bit-reversed* sequence $(p_0, p_1, \ldots, p_{k-2}, p_{k-1})$.

So bit reversal transforms a binary integer

$$p = p_{k-1}2^{k-1} + p_{k-2}2^{k-2} + \cdots + p_1 2 + p_0$$

into the binary integer

$$q = B(p) := p_{k-1} + p_{k-2}2 + \cdots + p_1 2^{k-2} + p_0 2^{k-1}.$$

Proposition 24 For each index $N = 2^n$, the FFT amounts to the following chain of operations:

(0) For each binary index $p \in \{0, \ldots, N-1\}$, calculate the bit-reversed index $q = B(p)$ and arrange the data $\vec{\mathbf{z}} = (z_0, \ldots, z_{N-1})$ in the order $\vec{\mathbf{z}}_B := (z_{B(0)}, \ldots, z_{B(n-1)})$;

(1) For each $k \in \{0, \ldots, n-1\}$, perform one step of the FFT on each of the $2^{(n-k)-1}$ pairs of adjacent sequences of 2^k elements.

We can therefore perform the Fast Fourier Transform in the following way:

- For each $p \in \{0, \ldots, N-1\}$, calculate $q = B(p)$ and arrange the data $\vec{\mathbf{z}} = (z_0, \ldots, z_{N-1})$ as

$$\vec{\mathbf{z}}_B = (z_{B(0)}, \ldots, z_{B(N-1)})$$

- Compute the $N/2$ Fast Fourier Transform steps from one to two points of each of the $N/2$ pairs:

$$([\hat{z}_{B(0)}, \hat{z}_{B(1)}], \ldots, [\hat{z}_{B(N-2)}, \hat{z}_{B(N-1)}]).$$

- Compute the $N/4$ Fast Fourier Transform steps from two to four points of each of the $N/4$ sequences of four numbers:

$$([\hat{z}_{B(0)}, \hat{z}_{B(1)}, \hat{z}_{B(2)}, \hat{z}_{B(3)}], \ldots, [\hat{z}_{B(N-3)}, \hat{z}_{B(N-2)}, \hat{z}_{B(N-1)}, \hat{z}_{B(N-1)}]).$$

- Compute the Fast Fourier Transform step from $N/2$ to N points of the sequence of N numbers:

$$([\hat{z}_{B(0)}, \ldots, \hat{z}_{B(N/2)-1)}], [\hat{z}_{B(N/2)}, \ldots, \hat{z}_{B(N-1)}]).$$

12.5.6 Applications of the FFT

Noise Reduction Through FFT

The idea is that if we are given a periodic signal with a larger amplitude added to noise, the Fast Fourier Transform can be used to decompose the the superposition of noise and the signal itself into a linear combination if terms with different frequencies. This will identify and preserve the contribution of the signal due to the larger coefficients and rejects the smaller coefficients from the noise. The original signal thus reemerges.

See p.165-167 in the book.

12.5.7 Multidimensional DFT and FFT