



NTNU

Norwegian University of Science and Technology

TMA4125 Matematikk 4N

Fourier Transforms: Continuous and Discrete.

Ronny Bergmann

Institute of Mathematical Sciences, NTNU.

February 16, 2023

Fourier for non-periodic functions?

Question.

How can we analyse functions $f: \mathbb{R} \rightarrow \mathbb{C}$ that are **not periodic**?

Idea.

- ▶ For some L “cut out” the interval $[-L, L]$
- ▶ assume that it **is periodic** and use Fourier series
- ▶ Let $L \rightarrow \infty$

Definition: (Continuous) Fourier Transform

Let $f \in L_1(\mathbb{R})$ (absolute integrable, i. e. $\int_{-\infty}^{\infty} |f(x)| dx < \infty$).

Then the (continuous) Fourier Transform is defined by

$$\hat{f}(\omega) := \mathcal{F}(f)(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-i\omega x} dx, \quad \omega \in \mathbb{R}.$$

If $g(\omega)$ is absolutely integrable, then the inverse Fourier transform is defined by

$$\check{g}(x) := \mathcal{F}^{-1}(g) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} g(\omega) e^{i\omega x} d\omega, \quad x \in \mathbb{R}.$$

Be careful with the definition of the Fourier Transform

Warning. It is not uniquely given, where to place the factor $\frac{1}{2\pi}$
The following definitions are usually used

- ▶ $\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x)e^{-i\omega x} dx$ (ours, both \mathcal{F} and \mathcal{F}^{-1} with same factor)
- ▶ $\int_{-\infty}^{\infty} f(x)e^{-i\omega x} dx$ (then the inverse has $\frac{1}{2\pi}$ as a factor)
- ▶ $\frac{1}{2\pi} \int_{-\infty}^{\infty} f(x)e^{-i\omega x} dx$ (no factor for the inverse)
- ▶ $\int_{-\infty}^{\infty} f(x)e^{-2\pi i\omega x} dx$ (frequency in Hz, no factor for the inverse)

Sometimes even the minus in the exponent might be with the inverse transform.

Be careful. When you see a Fourier transform and first check [which definition](#) was used.

Cosine & Sine Transform

Using Euler's formula we can do the same derivation as before using \cos and \sin .

Let f, g be **even** functions and $f, g \in L_1(\mathbb{R})$.

We define the **Fourier Cosine Transform** (and its inverse) as

$$\hat{f}_c(\omega) = \sqrt{\frac{2}{\pi}} \int_0^{\infty} f(x) \cos(\omega x) \, dx \quad \check{g}_c(x) = \sqrt{\frac{2}{\pi}} \int_0^{\infty} g(\omega) \cos(\omega x) \, d\omega$$

Let f, g be **odd** functions and $f, g \in L_1(\mathbb{R})$.

$$\hat{f}_s(\omega) = \sqrt{\frac{2}{\pi}} \int_0^{\infty} f(x) \sin(\omega x) \, dx \quad \check{g}_s(x) = \sqrt{\frac{2}{\pi}} \int_0^{\infty} g(\omega) \sin(\omega x) \, d\omega$$

Advantage. If f (and g) are real-valued, so are their transforms.

Warning. Again – be careful with the scaling.

Inversion of the Fourier Transform

Theorem. Let f and \hat{f} be absolutely integrable.

Then

$$f(x) = \mathcal{F}^{-1}(\hat{f}) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \hat{f}(\omega) e^{i\omega x} d\omega$$

Proof. Omitted.

Time vs Frequency Domain

When we consider f and its Fourier Transform \hat{f} we have

$$f(x) \underset{\mathcal{F}^{-1}}{\overset{\mathcal{F}}{\rightleftharpoons}} \hat{f}(\omega)$$

and we call

- ▶ f to be in the **time domain**
- ▶ \hat{f} to be in the **frequency domain**

Example I

Example. Let $a, b \in \mathbb{R}$, $a < b$. We consider the indicator function

$$f(x) = X_{[a,b]} = \begin{cases} 1 & \text{if } x \in [a, b] \\ 0 & \text{else.} \end{cases}$$

Further examples

Example. Let $a > 0$ be given and set $f(x) = e^{-a|x|}$. Then

$$\mathcal{F}(e^{-a|x|}) = \sqrt{\frac{2}{\pi}} \frac{a}{a^2 + \omega^2}$$

Example. Let $a > 0$ be given and set $f(x) = e^{-ax^2}$. Then

$$\mathcal{F}(e^{-ax^2}) = \frac{1}{\sqrt{2a}} e^{-\frac{\omega^2}{2a}}$$

or in other words:

The Fourier transform of a Gaussian is again a Gaussian.

Linearity & Derivatives and the Fourier transform

Theorem. Let $a, b \in \mathbb{C}$ and $f, g \in L^1(\mathbb{R})$. Then

$$\mathcal{F}(af + bg) = a\mathcal{F}(f) + b\mathcal{F}(g)$$

Proof. Left as an exercise, but follows directly from linearity of integration

Theorem. Assume that both f and f' are in $L^1(\mathbb{R})$ and that $f(x) \rightarrow 0$ for $x \rightarrow \pm\infty$.

Then it holds that

$$\mathcal{F}(f') = i\omega\mathcal{F}(f).$$

Proof.

Fourier Transform and Convolution

Definition. (Convolution III, Fourier version)

Let $f, g \in L^1(\mathbb{R})$. Then the convolution $f * g$ is defined by

$$(f * g)(x) = \int_{-\infty}^{\infty} f(y)g(x - y) \, dy.$$

This is actually well defined and we even have $f * g \in L^1(\mathbb{R})$.

Theorem. Similar to the 2π periodic case we have $f * g = g * f$.

Theorem. Let $f, g \in L^1(\mathbb{R})$. Then

$$\mathcal{F}(f * g) = \sqrt{2\pi} \cdot \mathcal{F}(f) \cdot \mathcal{F}(g).$$

So again. A **convolution** in time turns into a **multiplication** in frequency domain.



NTNU

Norwegian University of Science and Technology

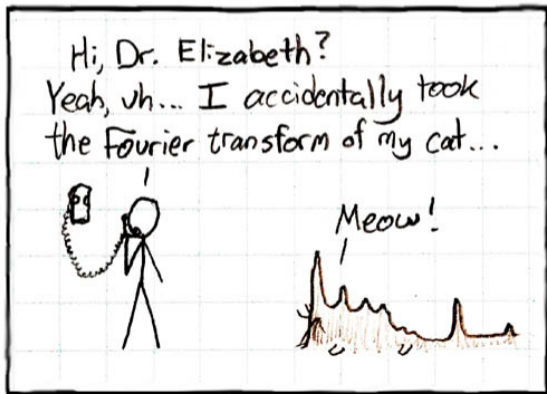
The Discrete Fourier Transform

Overview and Motivation

Concerning Fourier Transforms we now have Transforms for

1. 2π -periodic functions $f \Rightarrow$ **Fourier Series** $c_k(f), k \in \mathbb{Z}$
2. functions $f \in L^1(\mathbb{R})$ we obtain the **Fourier Transform** $\hat{f}(\omega)$

Question. Can we Fourier transform a **Signal** $(f_0, f_1, \dots, f_{N-1})$?
(or: what else can we Fourier transform?)



Towards Signals: Discretizing Fourier coefficients

Idea. Use a composite trapezoidal rule to approximate $c_k(f)$.

We use $[0, 2\pi)$ and a 2π -periodic function f .

We choose N equidistant sampling points $x_j = \frac{2\pi j}{N}$, $j = 0, \dots, N - 1$.

Then we obtain for any $k \in \mathbb{Z}$.

$$c_k(f) = \frac{1}{2\pi} \int_0^{2\pi} f(x) e^{-ikx} dx \approx \frac{1}{N} \sum_{j=0}^{N-1} f\left(\frac{2\pi j}{N}\right) e^{-2\pi i j k / N}$$

Shift view: Instead of the function f
 \Rightarrow think of a given **signal** (or vector) of values

$$(f_0, f_1, \dots, f_{N-1}) \in \mathbb{C}^N$$

here:

$$f_j := f\left(\frac{2\pi j}{N}\right), \quad j = 0, \dots, N - 1.$$

The Discrete Fourier Transform

For a given signal $\mathbf{f} = (f_0, f_1, \dots, f_{N-1})^T \in \mathbb{C}^N$
the **Discrete Fourier Transform** (DFT) is defined as

$$\hat{f}_k = \sum_{j=0}^{N-1} f_j e^{-2\pi i j k / N}, \quad k = 0, \dots, N-1.$$

We introduce $w_N = e^{-2\pi i / N}$ and $\hat{\mathbf{f}} = (\hat{f}_0, \dots, \hat{f}_{N-1})^T \in \mathbb{C}^N$.
Then we write the DFT also in **matrix-vector form**

$$\hat{\mathbf{f}} = \mathcal{F}_N \mathbf{f}$$

where the matrix $\mathcal{F}_N \in \mathbb{C}^{N \times N}$ is given by

$$\mathcal{F}_N = \left(e^{-2\pi i j k / N} \right)_{j,k=0}^{N-1} = \left(w_N^{j k} \right)_{j,k=0}^{N-1}$$

is the N th **Fourier matrix**.

Note. Following Kreyszig we get $\hat{f}_k \approx N c_k(f)$

DFT – Historical Remarks

The discrete Fourier transform is even older than the theory developed by Fourier (1807)

- ▶ mentioned by Lagrange 1758 for sine functions (DST)
- ▶ Lagrange, Clairaut (1754), and Euler used them to determine orbits of celestial bodies
- ▶ was also used by Gauss to determine the orbit of Ceres
($N = 12$, \approx in 1801)

Properties of the discrete Fourier coefficients \hat{f}_k

N -periodicity The discrete Fourier coefficients are N -periodic, i. e.

$$\hat{f}_k = \hat{f}_{k+N}, \quad k \in \mathbb{Z}.$$

(Exercise, note what happens to w_N^N).

Symmetry for real-valued signals if the f_j (or the function f) is real-valued then

$$\hat{f}_k = \overline{\hat{f}_{-k}}$$

Aliasing formula. Let $f \in C([-\pi, \pi))$ and let $\sum_k |c_k(f)| < \infty$ then the **Aliasing formula** holds:

$$\hat{f}_k = \sum_{\ell \in \mathbb{Z}} c_{k+\ell N}(f), \quad k \in \mathbb{Z}.$$

Results of Aliasing

For even N and if f is a trigonometric polynomial of degree $\frac{N}{2}$, i. e. of the form

$$f(x) = \sum_{k=-N/2+1}^{N/2-1} c_k(f) e^{2\pi i k x}$$

in other words: in the sums only the terms $\ell = 0$ are present

Then $\hat{f}_k = c_k(f)$, $k = -N/2 + 1, \dots, N/2 - 1$.

Remark.

This also means: Given f_0, \dots, f_{N-1} sampling values of such an f , we can **uniquely reconstruct** the function.

Having “exactly enough samples”, $2m + 1$ for highest frequency $m = N/2$ is called the **Nyquist rate**.

This is also called **periodic interpolation** or **trigonometric interpolation**.

Inverse Discrete Fourier Transform

Theorem. For a given signal $\hat{\mathbf{f}} = (\hat{f}_0, \hat{f}_1, \dots, \hat{f}_{N-1})^T \in \mathbb{C}^N$ the Inverse Discrete Fourier Transform (IDFT) is given by

$$f_j = \frac{1}{N} \sum_{k=0}^{N-1} \hat{f}_k e^{2\pi i j k / N}, \quad j = 0, \dots, N-1.$$

Remember $w_N = e^{-2\pi i / N}$ and $\mathbf{f} = (f_0, \dots, f_{N-1})^T \in \mathbb{C}^N$.

Then we have since $e^{2\pi i j k / N} = \overline{w_N^{jk}}$ that

$$\mathbf{f} = \frac{1}{N} \overline{\mathcal{F}_N} \hat{\mathbf{f}}$$

Proof. *Idea.* We have to show (plugging in $\hat{\mathbf{f}} = \mathcal{F}_N \mathbf{f}$) that $\mathbf{f} = \frac{1}{N} \overline{\mathcal{F}_N} \mathcal{F}_N \mathbf{f}$ or in other words that $\frac{1}{N} \overline{\mathcal{F}_N} \mathcal{F}_N$ is the identity matrix.

The Fast Fourier Transform – Motivation

Disadvantage of the DFT: Computing it takes $\mathcal{O}(N^2)$ operations.

Example.

In processing audio: Filter (e. g. remove noise) is a convolution , but 44.1kHz Sampling in Audio $\Rightarrow N = 44100$ for 1 sec. of audio data.

Audio length	Operations	(sec.) on modern CPU (i9)
1 sec.	$1.94 \cdot 10^9$	$0.388 \cdot 10^{-4}$
1 min.	$7 \cdot 10^{12}$	0.14
1 hr	$2.52 \cdot 10^{16}$	504 (8 Min.)

That is not feasible, e. g. when we need this “in real-time” (below $\frac{1}{50}$ th of a second).

Can we maybe use the structure of \mathcal{F} to make it faster?

The Fast Fourier Transform – Approach

Cooley & Tuckey (1965) presented an algorithm for the DFT (and hence IDFT) that we illustrate for the case for $N = 2^n$

The Radix-2-FFT.

We will just present the idea.

1. For each k computing \hat{f}_k can be split the sum the first and second half
 2. Then looking at even and odd k , each of these sets is a DFT($\frac{N}{2}$) (after N additions/subtractions and $N/2$ multiplications)
- ⇒ reduces $\mathcal{O}(N^2)$ to “2 times $\mathcal{O}(\frac{N^2}{4})$ plus $\mathcal{O}(\frac{N}{2})$ multiplications”

FFT – Number of operations

If we iterate this idea, we can do the DFT(N) (remember $N = 2^n$)

- ▶ 2 DFT($\frac{N}{2}$) and $N/2$ multiplications
- ▶ 4 DFT($\frac{N}{4}$) and $2\frac{N}{2}$ multiplications
- ▶ ...
- ▶ 2^k DFT($\frac{N}{2^k}$) and $k\frac{N}{2}$ multiplications
- ▶ ...
- ▶ $N = 2^n$ DFT(1) and $n\frac{N}{2}$ multiplications

The number of additions to combine the DFT(1)s again is nN .

⇒ combining these yields that the Fast Fourier transform requires $\mathcal{O}(nN) = \mathcal{O}(\log(N)N)$ operations.

Example.

The approx. 8 min. in the audio example reduce to $5 \cdot 10^{-4}$ sec.

speedup of about $8.9 \cdot 10^6$

FFT – Historical Remarks

You can use arbitrary factorisations $N = N_1 N_2$ but then the formula is more complicated

- ▶ 1965 Cooley & Tukey introduced the FFT, which made it fast and usable on computers
- ▶ 1905 the main idea of the FFT was described already by C. Runge
- ▶ 1801 Gauss (to determine the orbit of Ceres) split his DFT $N = 12$ into $N = 4$ and $N = 3$

Today there is a large variety of algorithms/splitting techniques collected within the FFTW.

Python

Discrete Sine and Cosine Transform (DST & DCT)

For the discrete case you can use the same ideas we already saw for **even** (cosine) and **odd** (sine) transform approaches.

Advantage. A real-valued signal stays real-valued

Variants. For the discrete signal there are **4 ways** of continuing a signal even/odd.

⇒ There exist

- ▶ 4 discrete sine transforms (DST-I to DST-IV)
- ▶ 4 discrete cosine transforms (DCT-I to DCT-VI)

Fast implementations based on the FFT idea are available here as well.
Most used: DCT-II.

Application: Image Processing and Compression: Barbara



172 kB

Application: Image Processing and Compression: Barbara



29.9 kB (17.3 %)

Application: Image Processing and Compression: Barbara



18.9 kB (11 %)

Application: Image Processing and Compression: Barbara



10 kB (5.8 %)

Application: Image Processing and Compression: Barbara



5.59 kB (3.25 %)

Application: Image Processing and Compression: Barbara



3 kB (1.74 %)