TMA4125 Matematikk 4N

Fourier Transform

Ronny Bergmann

Department of Mathematical Sciences, NTNU.

March 7, 2022

Fourier for non-periodic functions?

Question. How can we analyse functions $f : \mathbb{R} \to \mathbb{C}$ that are not periodic?

Idea.

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

Definition: (Continuous) Fourier Transform

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

Then the (continuous) Fourier Transform is defined by

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

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

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

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

- $\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} f(x)\mathrm{e}^{-\mathrm{i}\omega x}\,\mathrm{d}x$ (ours, both $\mathcal{F}$ and $\mathcal{F}^{-1}$ with same factor)
- $\int_{-\infty}^{\infty} f(x)\mathrm{e}^{-\mathrm{i}\omega x}\,\mathrm{d}x$ (then the inverse has $\frac{1}{2\pi}$ as a factor)
- $\frac{1}{2\pi}\int_{-\infty}^{\infty} f(x)\mathrm{e}^{-\mathrm{i}\omega x}\,\mathrm{d}x$ (no factor for the inverse)
- $\int_{-\infty}^{\infty} f(x)\mathrm{e}^{-2\pi\mathrm{i}\omega x}\,\mathrm{d}x$ (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 wich definition was used.

Cosine & Sine Transform

Using Eulers formula we can do the same derivation as before using $\cos$ and $\sin$.

If f is an even function, we define the Fourier Cosine Transform (and its inverse) as

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

If f is an odd function, we define the Fourier Sine Transform (and its inverse) as

$$\hat{f}_{\mathrm{c}}(\omega) = \sqrt{\frac{2}{\pi}} \int_0^\infty f(x)\sin(\omega x)\,\mathrm{d}x \qquad \check{g}_{\mathrm{c}}(x) = \sqrt{\frac{2}{\pi}} \int_0^\infty g(\omega)\sin(\omega x)\,\mathrm{d}\omega$$

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

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

Proof. Ommitted.

Time vs Frequency Domain

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

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

and we call

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

Example I

Example. As a first xample let's look at an 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 $f(x) = \mathrm{e}^{-a|x|}$. Then

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

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

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

or in other words: The Fourier transform of a Gaussian is again a Gaussian.

Linearity & Derivatives and the Fourier transform

Theorem. Let $a, b \in C$ 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 $f(x) \to 0$ for $x \to \pm\infty$.
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

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

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.

The Discrete Fourier Transform

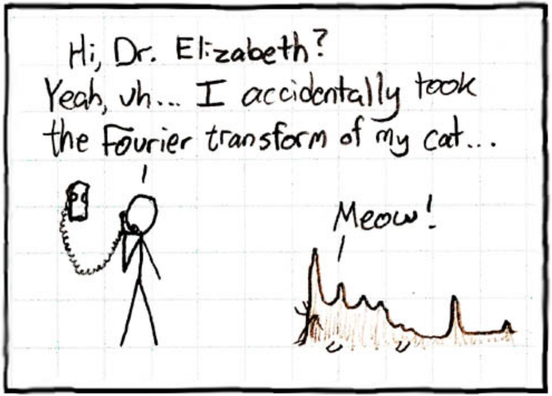
Overview and Motivation

Concerning Fourier Transforms we now have Transforms for

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

Question. Can we Fourier transform a Signal $(f_0, f_1, \ldots, f_{N-1})$?

(or: what else can we Fourier transform?)



source: https://xkcd.com/26/

Discrete Fourier Transform: Discretizing $c_k(f)$

Idea. Use a composite trapezoidal rule to approximate $c_k(f)$. For ease of notation we use $[0, 2\pi)$ and remember that f is 2π periodic.

So our sampling points are $x_j = \frac{2\pi j}{N}$, $j = 0, \ldots, N-1$.
Then we obtain for any $k \in \mathbb{Z}$.

$$c_k(f) = \frac{1}{2\pi} \int_0^{2\pi} f(x) \mathrm{e}^{\mathrm{i}kx} \,\mathrm{d}x \approx \frac{1}{N} \sum_{j=0}^{N-1} f\left(\frac{2\pi j}{N}\right) \mathrm{e}^{2\pi \mathrm{i} jk/N}$$

Instead of having the function f in mind, we could also (just) think of a given signal (or vector) of values

$$\left(f_0, f_1, \ldots, f_{N-1}\right)$$

with

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

The Discrete Fourier Transform

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

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

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

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

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

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

is the Nth Fourier matrix.

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), Euler used them to determine orbits of celestial bodies
- was also used by Gauss to determine the orbit of Ceres ($N = 12$, ≈ 1801)

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

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

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

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

$$\hat{f}_k = \hat{f}_{k+N}$$

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

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

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

Results of Aliasing

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

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

(or in other words only the terms $\ell = 0$ for $k = -N/2+1, \ldots, N/2-1$ are present)

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

Remark. This also means: Given $f_0, \ldots, f_{N-1}$ sampling values of this function f, we can uniqely reconstruct f.

This is also called periodic interpolation or trigonometric interpolation. If you just have the sampling values, this enables you to find the unique trigonometric polynomial of degree $N/2$ that interpolates your data.

Inverse Discrete Fourier Transform

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

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

Remember $w_N = \mathrm{e}^{-2\pi\mathrm{i}/N}$ and $\boldsymbol{f} = (f_0, \ldots, f_{N-1})^{\mathrm{T}} \in \mathbb{C}^N$.
Then we have since $\mathrm{e}^{2\pi\mathrm{i}jk/N} = \overline{w_N^{jk}}$ that

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

Proof. *Idea.* We have to show (plugging in $\hat{\boldsymbol{f}} = \mathcal{F}_N \boldsymbol{f}$) that $\boldsymbol{f} = \frac{1}{N} \overline{\mathcal{F}_N} \mathcal{F}_N \boldsymbol{f}$ 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 into even and odd indices j
2. One of them is directly a DFT($\frac{N}{2}$), the other one has a factor w_N upfront.

$\Rightarrow$ reduces $\mathcal{O}(N^2)$ to "2 times $\mathcal{O}(\frac{N^2}{4})$ plus $\mathcal{O}(\frac{N}{2})$ multiplications"

NTNU

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

Where the number of additions to combine the DFT(1)s again is nN.
$\Rightarrow$ 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.

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.
$\Rightarrow$ 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 %)