
# Die Fourier-Transformation

## Mathematisches Seminar - Numerik

Insitut für Mathematik

**Autor:** Jan Pfeiffer

07.01.2025

# Inhalt

# Triviales Beispiel

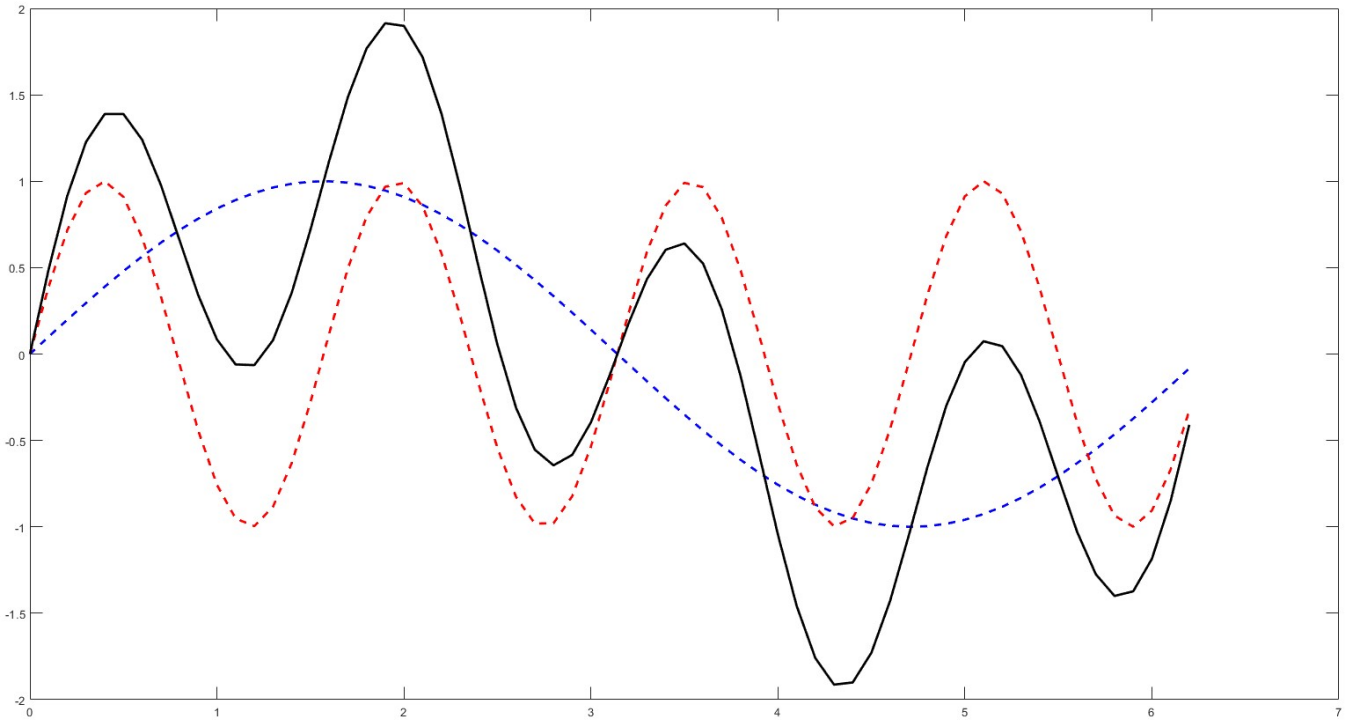


Abbildung: Überlagerung zweier Sinusfunktionen

# Nicht-triviales Beispiel

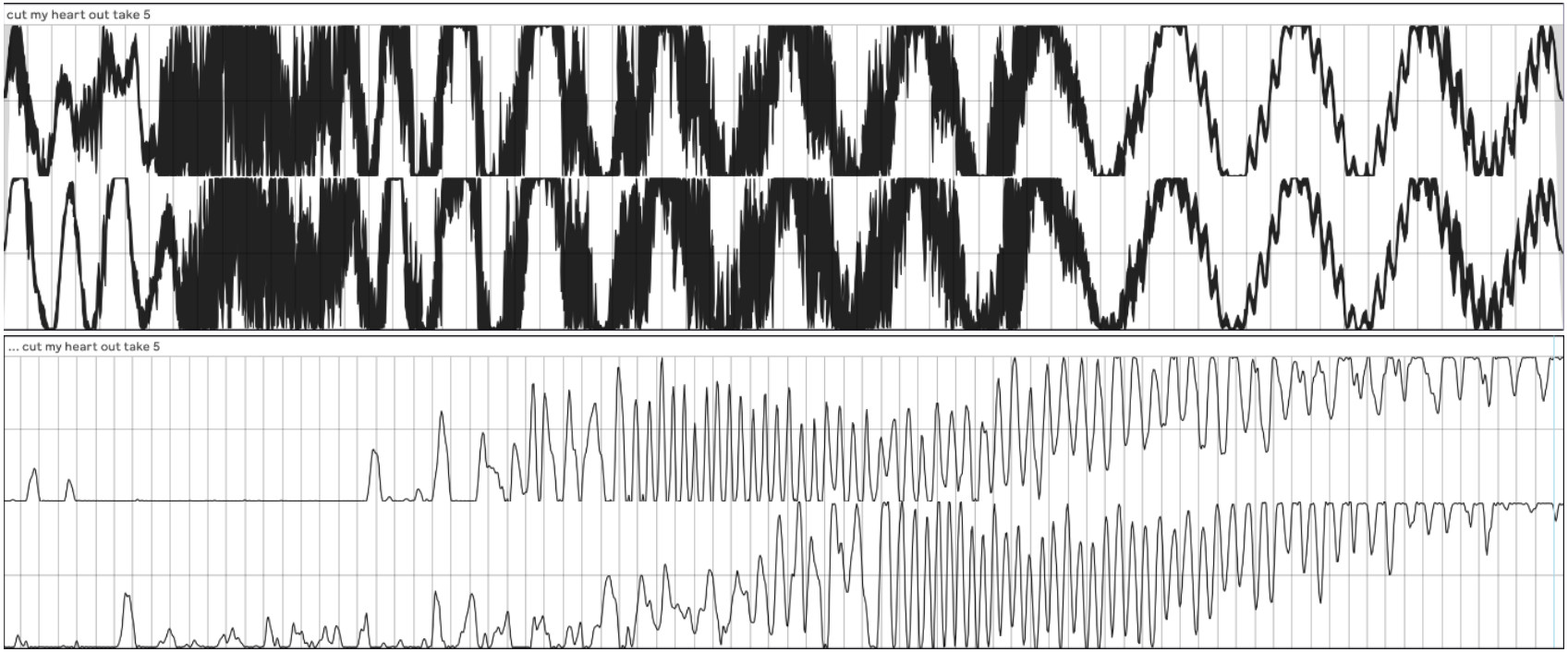


Abbildung: Ausschnitte eines Musikstücks

($\sim$ 0.5s oben, $\sim$ 0.005s unten)

# Analytische Definition

Analytische Definition für jegliche Funktionen:
kontin. Fourieranalyse:

$$f(x) = \frac{1}{\sqrt{2\pi}} \cdot \int_{-\infty}^{+\infty} y(t) \cdot (cos(xt) + i \cdot sin(xt))dt$$

kontin. Fourieranalyse (komplex):

$$f(x) = \frac{1}{\sqrt{2\pi}} \cdot \int_{-\infty}^{+\infty} y(t)e^{ixt}dt$$

Für periodische Funktionen:
Fourier-Reihe:

$$f(x) = \sum_{k=0}^{\infty} (c_k cos(kx) + d_k sin(kx)) \tag{1}$$

Ausgeschrieben bedeutet das

$$f(x) = (c_0 + c_1 cos(x) + d_1 sin(x)) + (c_2 cos(2x) + d_2 sin(2x)) + \dots$$

Fourier-Reihe (komplex):

$$f(x) = \sum_{k=0}^{\infty} \beta_k e^{ikx} \tag{2}$$

# Num. Berechnung - Diskrete Fouriertransformation

Voraussetzungen für die numerische Interpolationsaufgabe:

- Der Anschaulichkeit halber gilt $x \in [0, 2\pi]$
- Anzahl der Stützpunkte $N$
- Stützstellen $x_j = 2\pi \cdot \frac{j}{N}$
- Stützwerte $y_j \in \mathbb{R}$
- $j = 0, 1, \ldots, N-1$

Die Aufgabe besteht darin, geeignete $a_k, b_k$ zu finden, sodass mit

$$T(x) = \sum_{k=0}^{N-1} (a_k cos(kx) + b_k sin(kx)) \tag{3}$$

$T(x_j) = y_j$ gilt.

Summe geht auch über $k = 0, 1, \ldots, \lceil N/2 \rceil$. Geht auf das Nyquist-Shannon-Theorem zurück (siehe R1).

Eine Überführung in die komplexe Schreibweise bietet sich an, um das Problem zu vereinfachen.
Mithilfe der Euler'schen Formel $e^{i\phi} = cos(\phi) + i \cdot sin(\phi)$, folgt:

$$T(x) = \beta_0 + \beta_1 e^{ix} + \cdots + \beta_{N-1} e^{i(N-1)x} = \sum_{k=0}^{N-1} \beta_k e^{ikx} \qquad (4)$$

Gesucht sind hierbei die $\beta_k \in \mathbb{C}$, sodass $T(x_j) = y_j$ gilt.

Bei der Verallgemeinerung der gesuchten Parameter von $a_k$ und $b_k$ auf $\beta_k$ gehen keine Informationen verloren, weil

$$\begin{aligned} a_0 &= \beta_0 \\ a_k &= \beta_k + \beta_{N-k} \\ b_k &= i(\beta_k - \beta_{N-k}) \\ a_{N-1} &= \beta_{N-1} \end{aligned}$$

gilt. (Beweis siehe [Q2])

Die Interpolationsbedingung $T(x) = y$ in vektorieller Schreibweise:

$$y = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{bmatrix} = \sum_{k=0}^{N-1} \beta_k \begin{bmatrix} e^{ikx_0} \\ e^{ikx_1} \\ \vdots \\ e^{ikx_{N-1}} \end{bmatrix} = \sum_{k=0}^{N-1} \beta_k \omega^k \qquad (5)$$

Es gilt demnach

$$\omega^k = \begin{bmatrix} e^{ikx_0} \\ e^{ikx_1} \\ \vdots \\ e^{ikx_{N-1}} \end{bmatrix}$$

Mit $x_j = 2\pi \cdot j/N$ (siehe oben) und $j \in (0, 1, \ldots, N-1)$, folgt

$$\omega_N^k = \left[ \begin{array}{c} 1 \\ e^{ik \cdot 2\pi/N} \\ e^{ik \cdot 2\pi \cdot 2/N} \\ \vdots \\ e^{ik \cdot 2\pi \cdot (N-1)/N} \end{array} \right] \Bigg\} j = 0, 1, \ldots, N-1$$

Das Auflösen der Summe über k ergibt Folgendes:

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{bmatrix} = \beta_0 \cdot \begin{bmatrix} e^{i\cdot 0 x_0} \\ e^{i\cdot 0 x_1} \\ \vdots \\ e^{i\cdot 0 x_{N-1}} \end{bmatrix} + \cdots + \beta_{N-1} \cdot \begin{bmatrix} e^{i\cdot (N-1) x_0} \\ e^{i\cdot (N-1) x_1} \\ \vdots \\ e^{i\cdot (N-1) x_{N-1}} \end{bmatrix}$$

Mit $\omega_N^k = [e^{ikx_0}, e^{ikx_1}, \ldots, e^{ikx_{N-1}}]^T$ folgt

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{bmatrix} = \beta_0 \cdot \omega_N^0 + \beta_1 \cdot \omega_N^1 + \cdots + \beta_{N-1} \cdot \omega_N^{N-1}$$

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{bmatrix} = \begin{bmatrix} \beta_0 e^{i\cdot 0 x_0} + \beta_1 e^{i\cdot 1 x_0} + \cdots + \beta_{N-1} e^{i\cdot (N-1) x_0} \\ \beta_0 e^{i\cdot 0 x_1} + \beta_1 e^{i\cdot 1 x_1} + \cdots + \beta_{N-1} \cdot e^{i\cdot (N-1) x_1} \\ \vdots \\ \beta_0 e^{i\cdot 0 x_{N-1}} + \beta_1 e^{i\cdot 1 x_{N-1}} + \cdots + \beta_{N-1} \cdot e^{i\cdot (N-1) x_{N-1}} \end{bmatrix} \tag{6}$$

Gleichung (6) lässt sich auch durch folgende Matrix-Vektor-Multiplikation berechnen:

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2\cdot (N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2\cdot (N-1)} & \cdots & \omega^{(N-1)\cdot (N-1)} \end{bmatrix} \cdot \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{N-1} \end{bmatrix} \tag{7}$$

Zusammenfassend gilt

$$y = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{bmatrix} = \sum_{k=0}^{N-1} \beta_k \begin{bmatrix} e^{ikx_0} \\ e^{ikx_1} \\ \vdots \\ e^{ikx_{N-1}} \end{bmatrix} = \sum_{k=0}^{N-1} \beta_k \omega^k = F_N \cdot \beta \quad (8)$$

$$F_N \cdot \beta = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2\cdot(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2\cdot(N-1)} & \dots & \omega^{(N-1)\cdot(N-1)} \end{bmatrix} \cdot \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{N-1} \end{bmatrix} \quad (9)$$

Wobei $F_N$ die Fouriermatrix ist.

Da wir die Koeffzienten $\beta_k$ suchen, stellen wir um:

$$\beta = F^{-1}y = \frac{1}{N}\bar{F}_N^T y \tag{10}$$

(Beweis für $F_N^{-1} = \frac{1}{N} \cdot \bar{F}_N^T$ siehe Q2)

# Num. Berechnung - Beispiel $N = 4$

Veranschaulichung am Beispiel $N = 4$: $\omega_4^x = e^{ix \cdot 2\pi/4} = e^{ix \cdot \pi/2}$ repräsentieren die 4-ten Einheitswurzeln (x hier nur als Laufvariable):

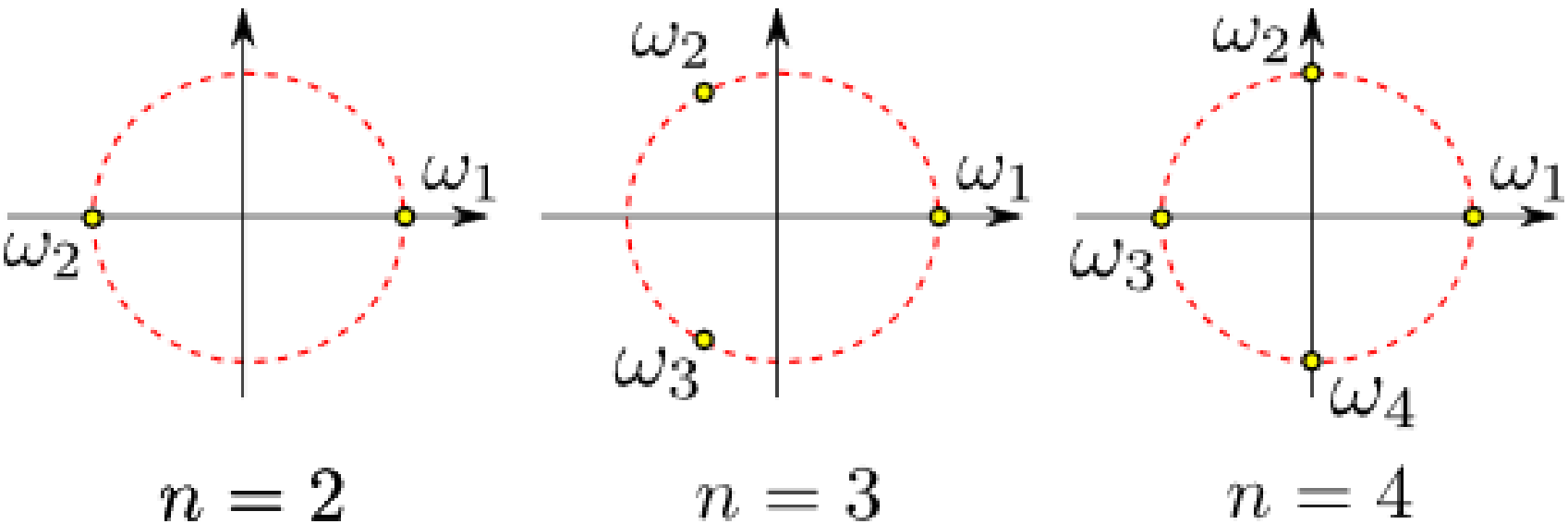


Abbildung: Veranschaulichung Einheitswurzeln [B1]

Es ist der Zusammenhang $\omega_4^x = \omega_4^{x \bmod 4}$ zu erkennen

Somit vereinfacht sich unsere Fourier-Matrix $F_4$ von

$$\begin{bmatrix} \omega_4^0 & \omega_4^0 & \omega_4^0 & \omega_4^0 \\ \omega_4^0 & \omega_4^1 & \omega_4^2 & \omega_4^3 \\ \omega_4^0 & \omega_4^2 & \omega_4^4 & \omega_4^6 \\ \omega_4^0 & \omega_4^3 & \omega_4^6 & \omega_4^9 \end{bmatrix} \text{ zu } \begin{bmatrix} \omega_4^0 & \omega_4^0 & \omega_4^0 & \omega_4^0 \\ \omega_4^0 & \omega_4^1 & \omega_4^2 & \omega_4^3 \\ \omega_4^0 & \omega_4^2 & \omega_4^0 & \omega_4^2 \\ \omega_4^0 & \omega_4^3 & \omega_4^2 & \omega_4^1 \end{bmatrix}$$

Einsetzen von $\omega_4 = e^{i \cdot 2\pi/4} = e^{i \cdot \pi/2}$ liefert

$$\begin{bmatrix} e^{0 \cdot i \cdot \pi/2} & e^{0 \cdot i \cdot \pi/2} & e^{0 \cdot i \cdot \pi/2} & e^{0 \cdot i \cdot \pi/2} \\ e^{0 \cdot i \cdot \pi/2} & e^{1 \cdot i \cdot \pi/2} & e^{2 \cdot i \cdot \pi/2} & e^{3 \cdot i \cdot \pi/2} \\ e^{0 \cdot i \cdot \pi/2} & e^{2 \cdot i \cdot \pi/2} & e^{0 \cdot i \cdot \pi/2} & e^{2 \cdot i \cdot \pi/2} \\ e^{0 \cdot i \cdot \pi/2} & e^{3 \cdot i \cdot \pi/2} & e^{2 \cdot i \cdot \pi/2} & e^{1 \cdot i \cdot \pi/2} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}$$

Zu lösende Ausgangsgleichung: $\beta = \frac{1}{N}\bar{F}_4^T y$

$$F_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix} \Longrightarrow \bar{F}_4^T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}$$

$$\Longrightarrow \beta = \frac{1}{N} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix} \cdot \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

Der Aufwand dieser Berechnung ist proportional zu $\mathcal{O}(N \cdot N) = \mathcal{O}(N^2)$

# Num. Berechnung - Schnelle Fouriertransformation

- FFT (Fast-Fourier-Transformation) verringert Aufwand zu $\mathcal{O}(N \cdot \log_2(N))$ - [Beweis siehe Q2]
- Einfachster Algorithmus Cooley-Tukey Algorithmus (nur für den Fall $N = 2^m$)
- Inputvektor $y$ wird zerlegt
- Über Symmetriebetrachtung wird eine Lösung berechnet, der Rest daraus hergeleitet (siehe Q4, R2)

# Anwendung: Fouriertransformation Musikstück

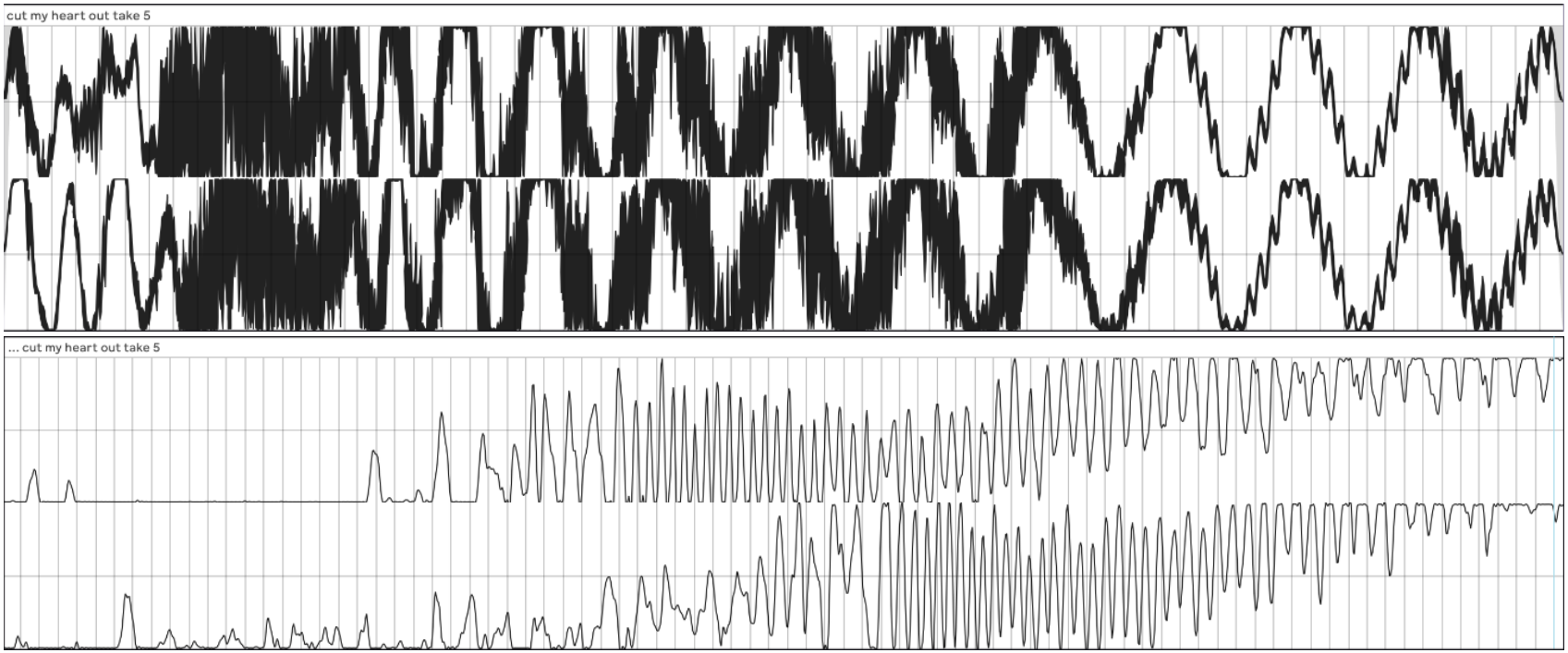


Abbildung: Ausschnitte eines Musikstücks - im Folgenden "Signal"

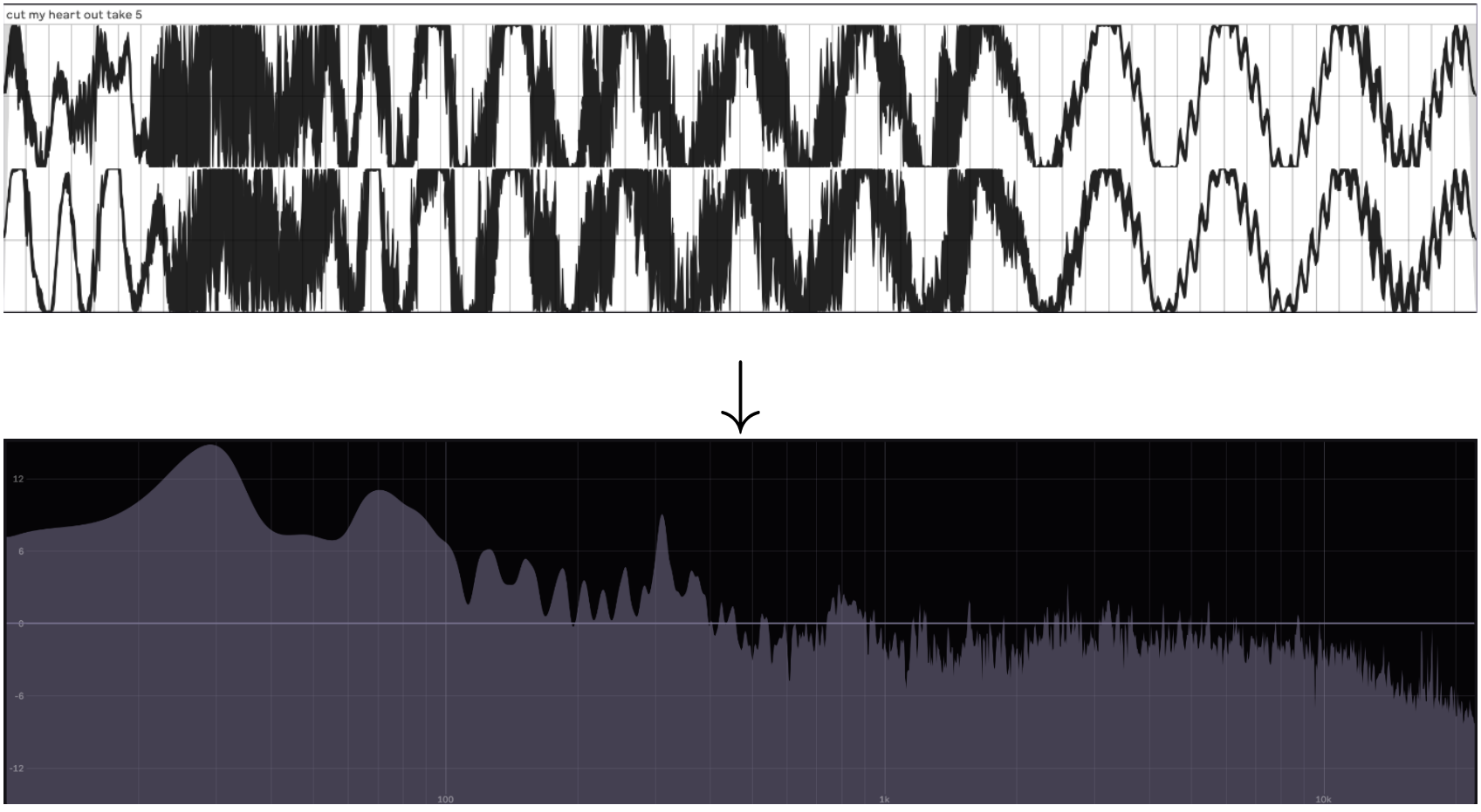


Abbildung: Ursprüngliches Signal (oben), FFT (unten)

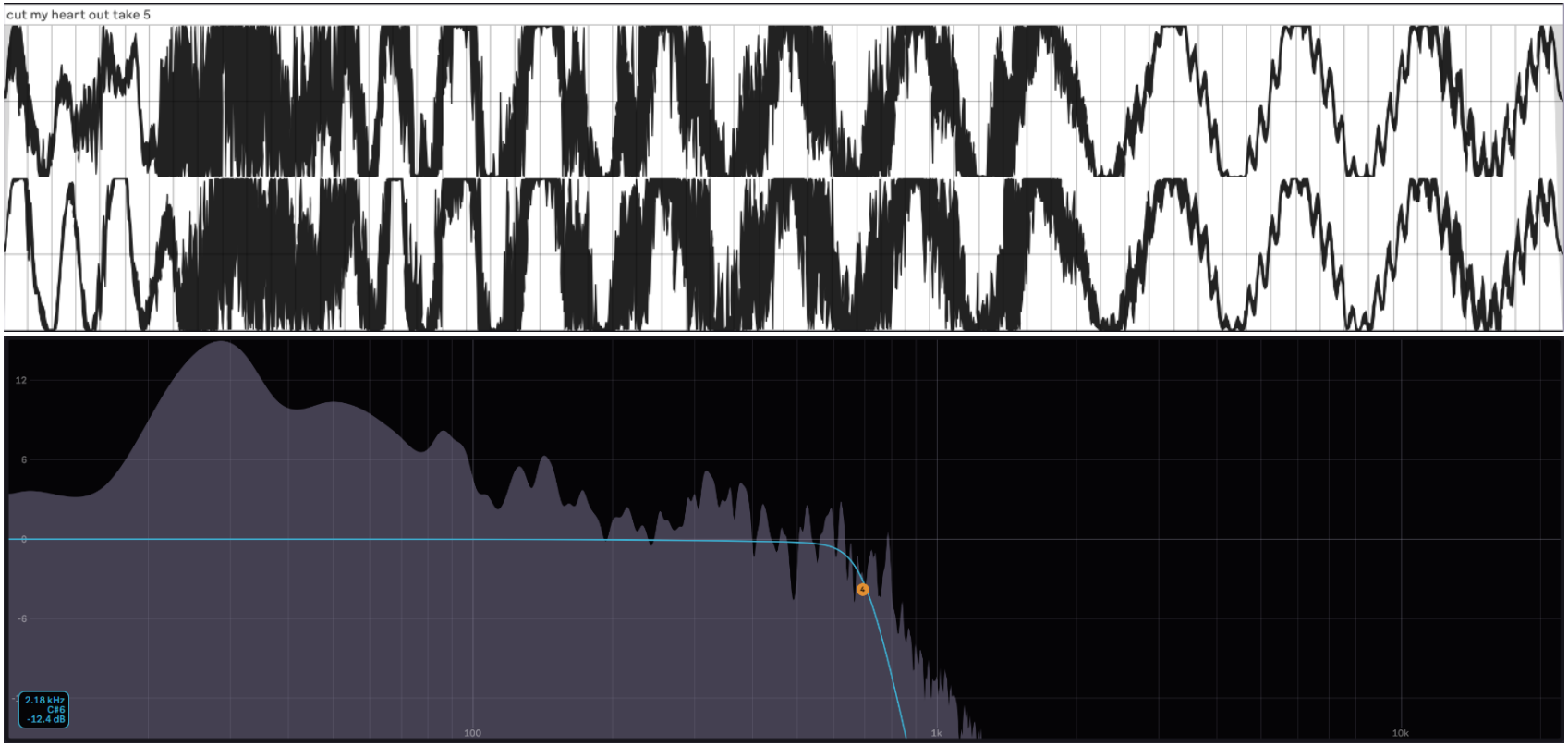


Abbildung: Tiefpass auf die FFT angewandt

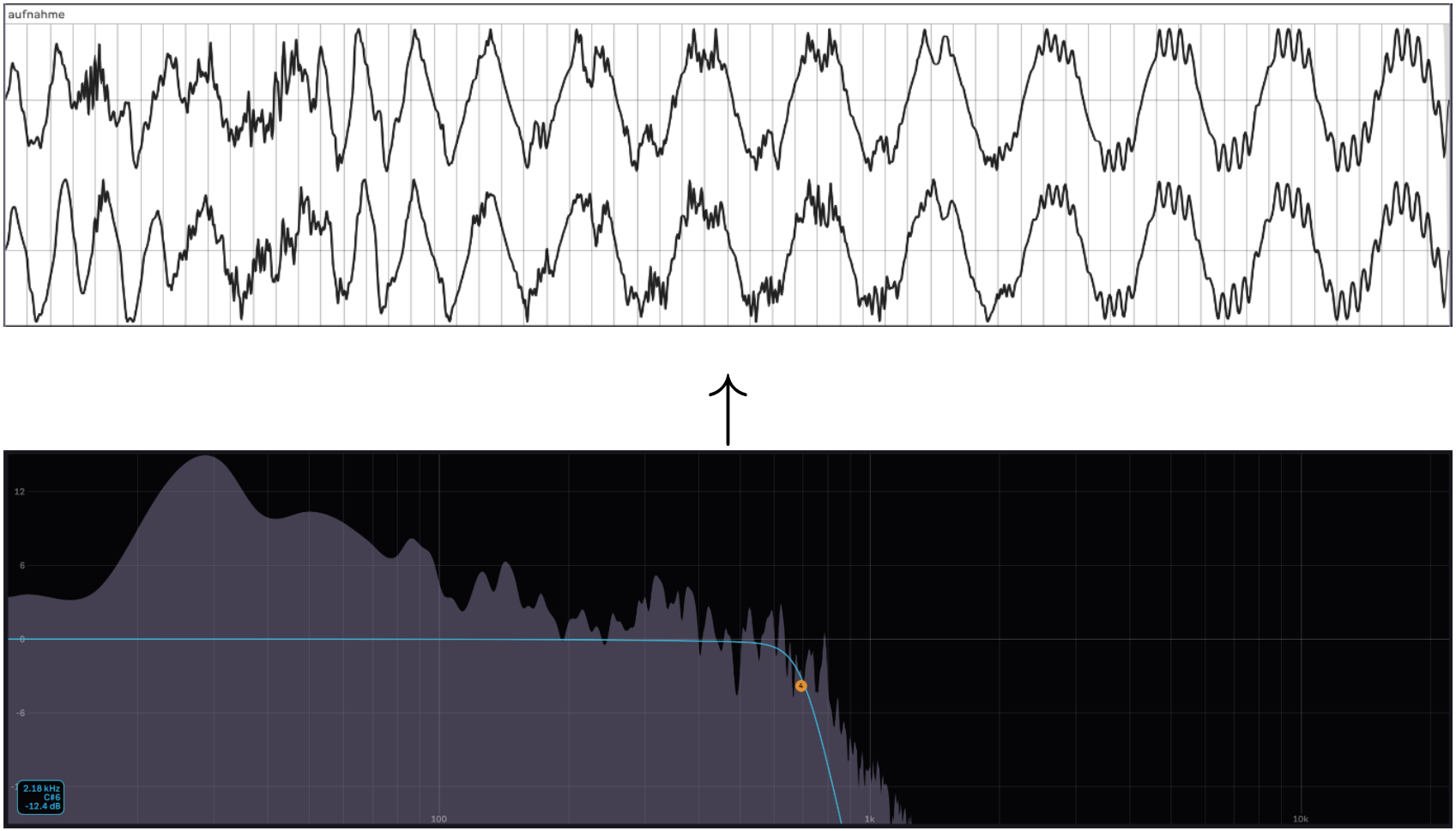


Abbildung: Rücktransformation des Frequenzspektrums - Signal mit Tiefpass

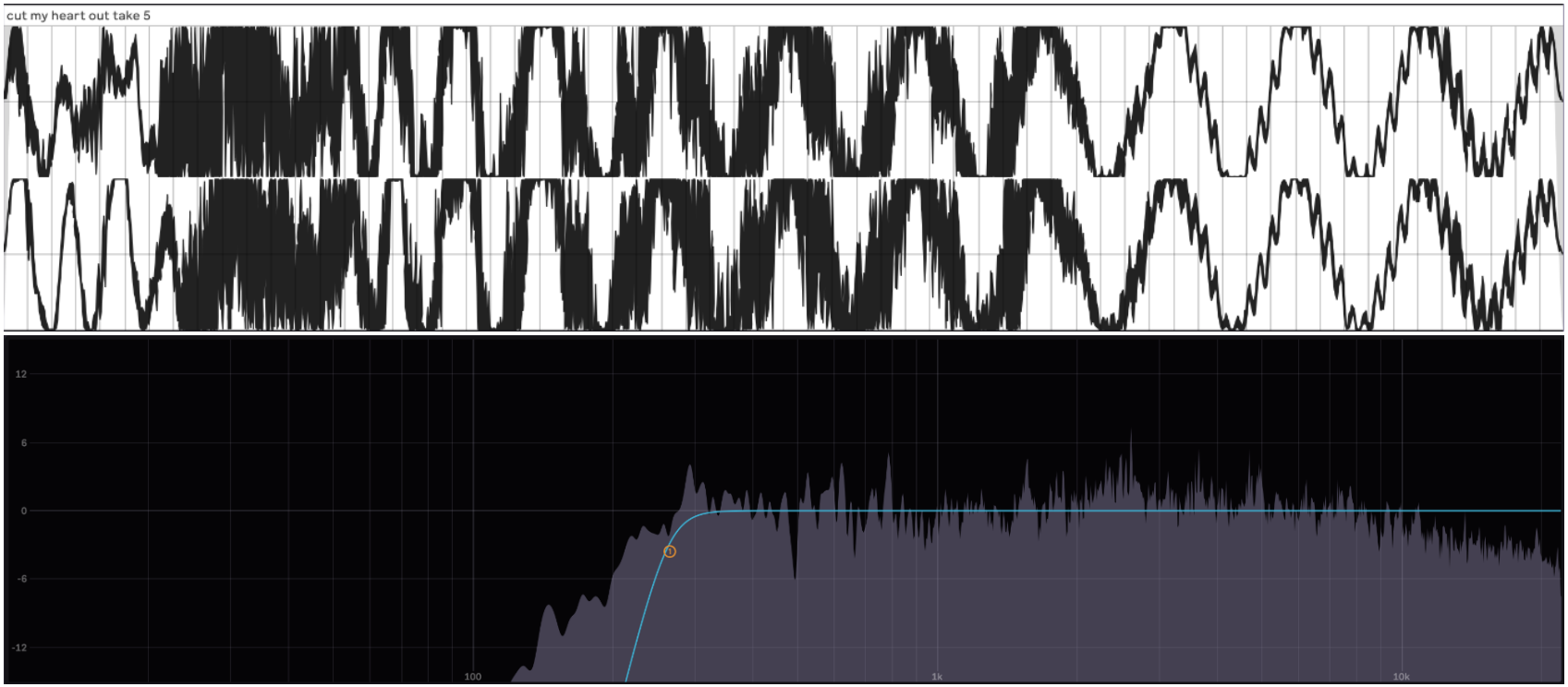


Abbildung: Hochpass auf das Frequenzspektrum angewandt

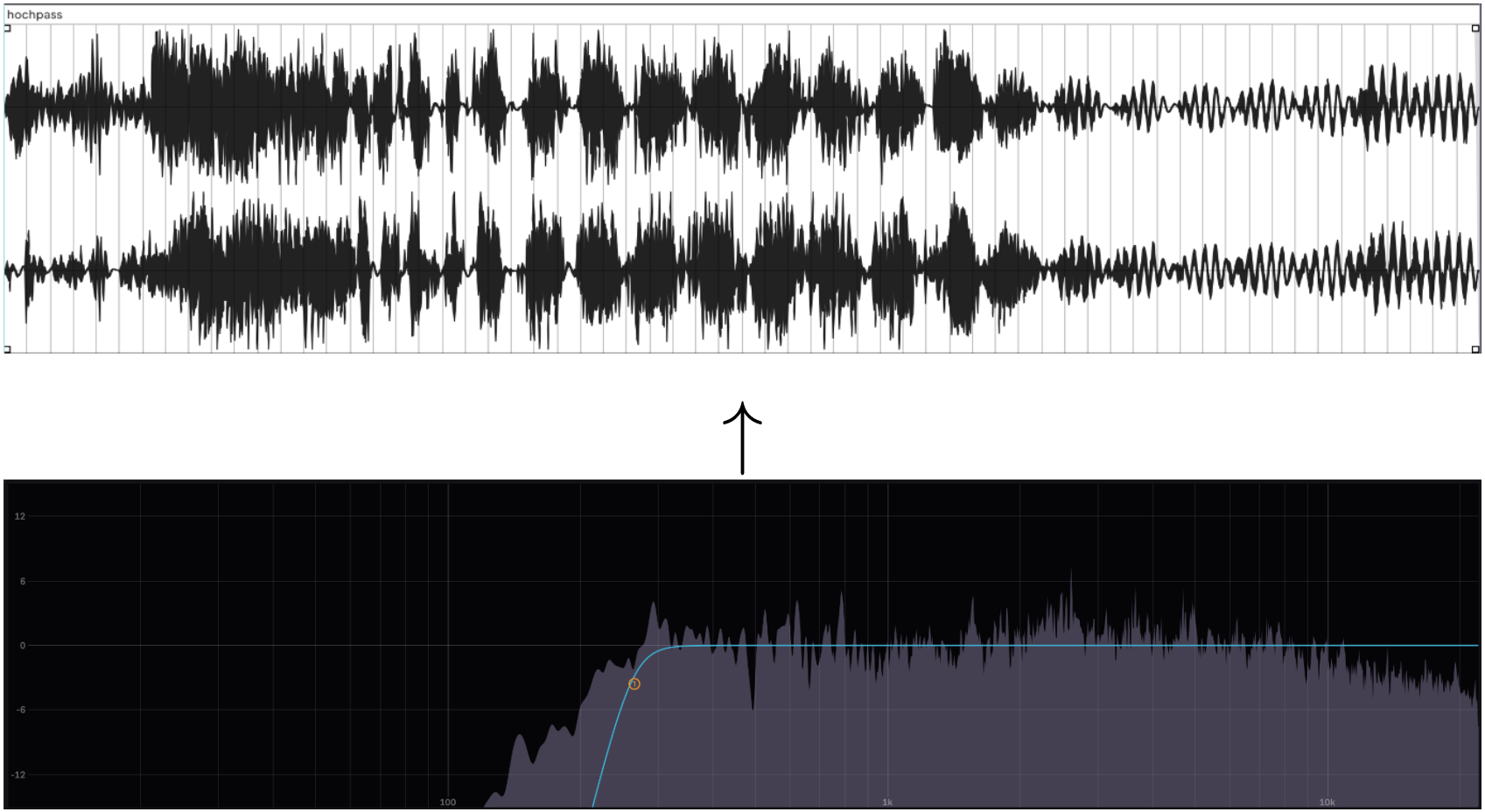


Abbildung: Rücktransformation des Frequenzspektrums - Signal mit Hochpass

# Anwendung in MATLAB

Unser Ziel für die MATLAB-Implementierung ist die Approximation einer Rechteckfunktion f(x) und ihrer Fourieranalyse mittels folgender Definition:

$$f(x) = \frac{4}{\pi} \sum_{k \text{ ungerade}}^{N} \frac{1}{k} sin(kx) \tag{11}$$

```
% Initialisierung der Parameter
Fs = 1000; % Abtastrate (Hz), Feinheit des Plots später
T = 1; % Signalperiode (s), in der Präsentation wurde 2\pi gewählt
N = Fs * T; % Anzahl der Samples, äquivalent zu N in der Präsentation
t = T*(0:N-1)/N; %auch = (0:N-1)/Fs ; Zeitvektor, äquivalent zu x_j in der Präsentation
anzahl_wellen = 1000;% Anzahl verwendeten Sinuswellen
```

Abbildung: Initialisierung der Parameter

```
% Endliche Fourier-Reihendarstellung der Square-Waveform
rechteck = zeros(size(t)); % Initialisierung des Signals
for k = 1:2:anzahl_wellen    % Nur ungerade Sinuswellen aufaddiert
    rechteck = rechteck + (1/k) * sin(k * 2 * pi * t / T);
end
```

Abbildung: Approximation der Rechteckfunktion über endliche Fourierreihe

```
% FFT-Analyse
fft_ausgabe = fft(rechteck); % Berechnung der FFT
fft_betrag = abs(fft_ausgabe/N); % Betrag der FFT normiert
fft_plot = fft_betrag(1:N/2+1); % Einseitiges Spektrum
f = Fs * (0:(N/2)) / N; % Frequenzvektor
```

Abbildung: Anpassung der Fouriertransformation

Anschließend plot(t, rechteck) und stem(f, fft_plot)

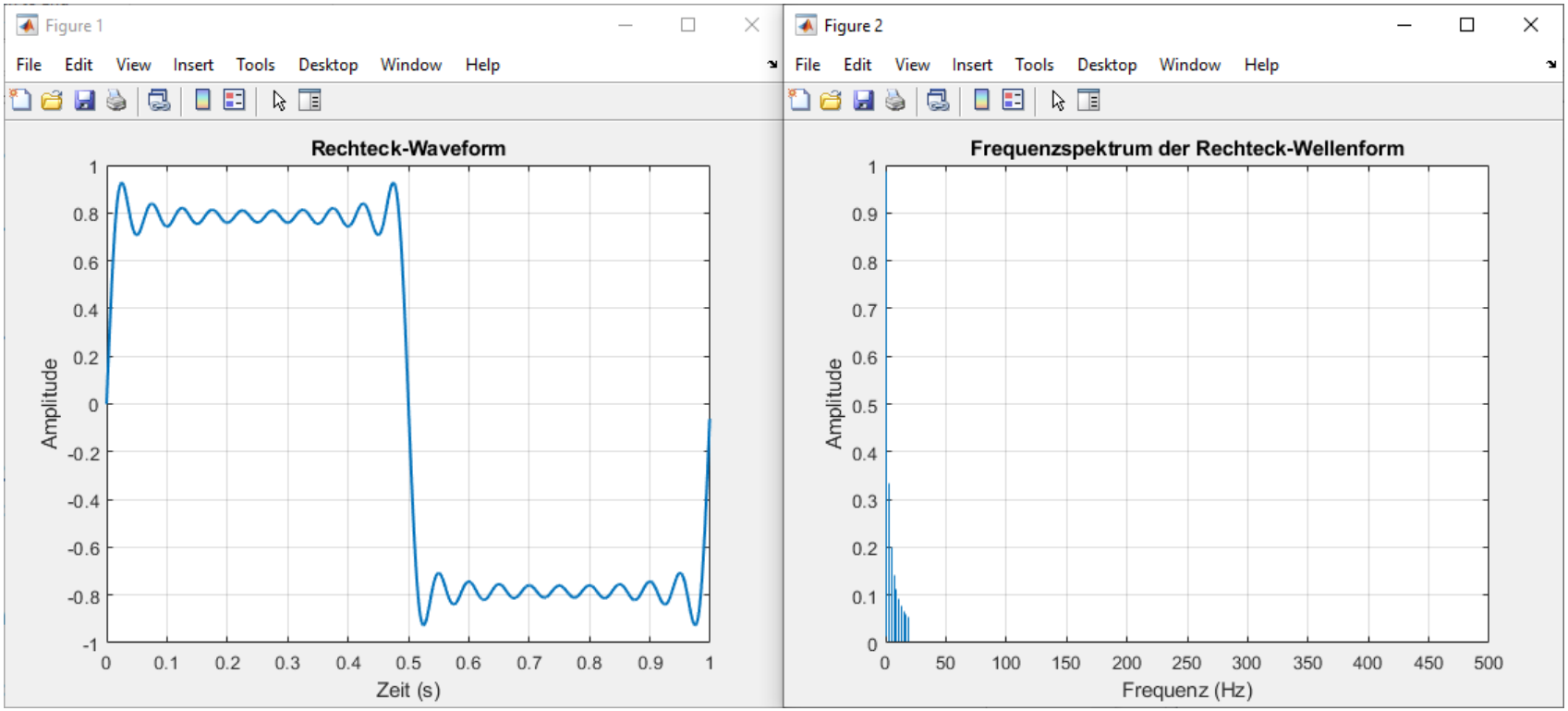


Abbildung: Ausgabe für N=10

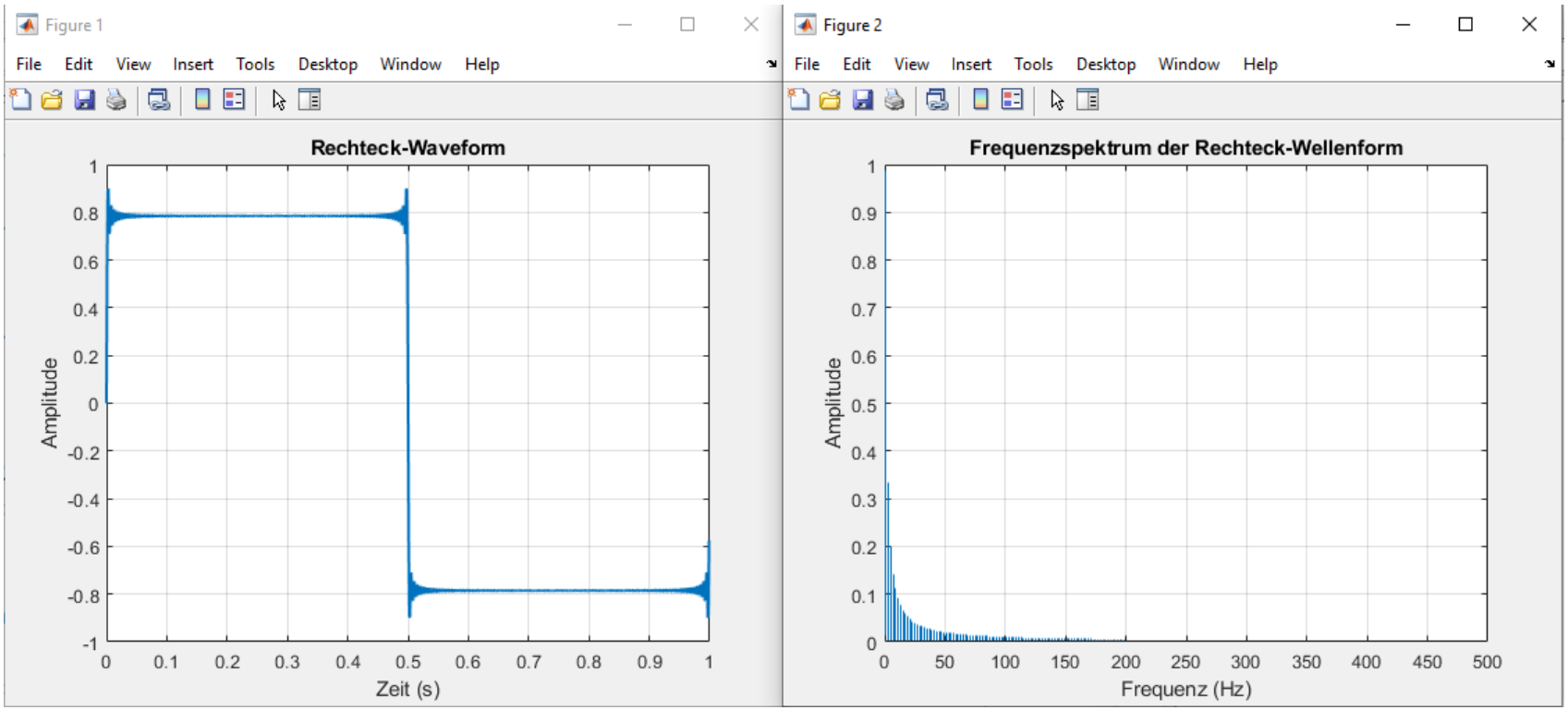


Abbildung: Ausgabe für N=100

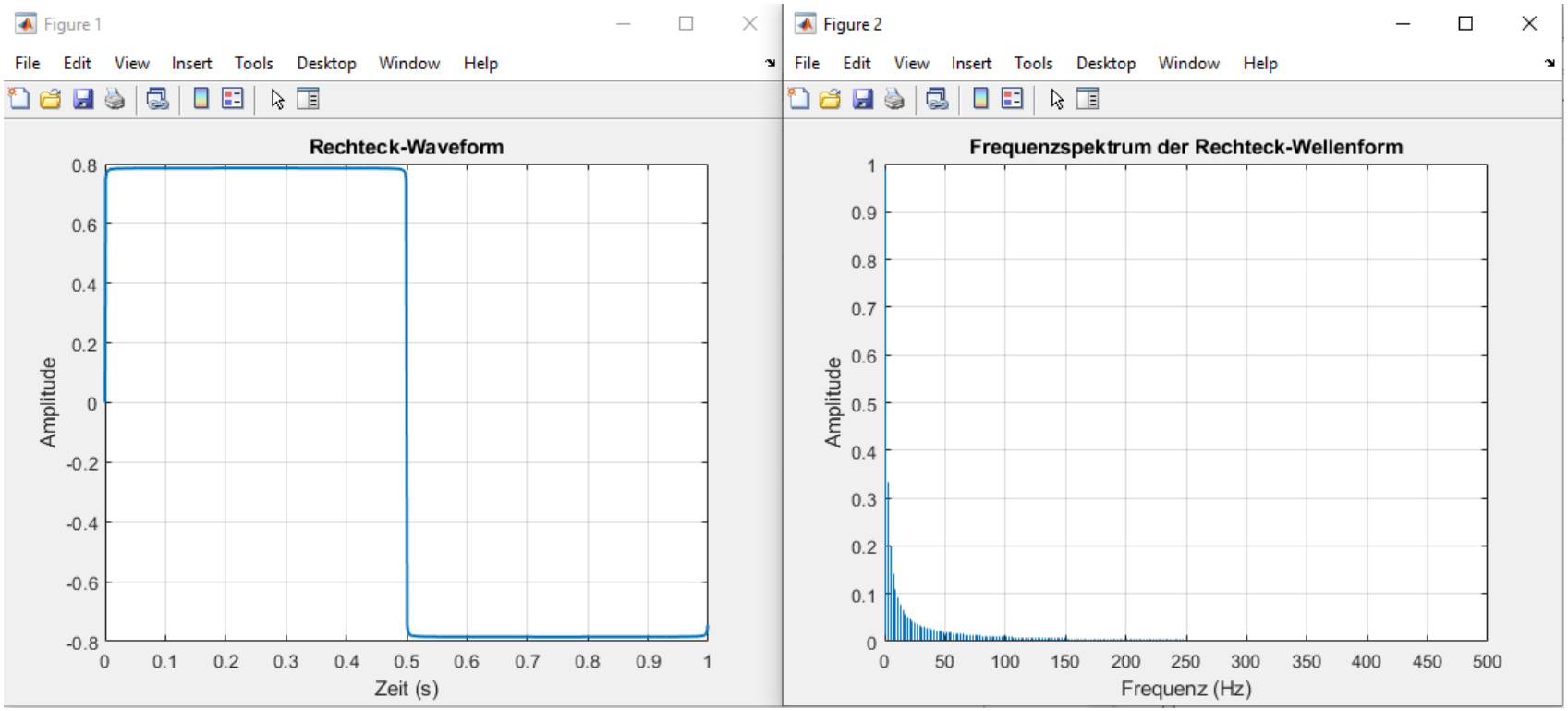


Abbildung: Ausgabe für N=1000

# Fazit

- Die Fouriertransformation interpoliert Funktionen und bietet dabei die Möglichkeit der Spektralanalyse (Umwandlung Zeitskala zu Frequenzskala)
- Diskrete Fouriertransformation: $y \mapsto \beta = \frac{1}{N} \cdot \bar{F}_N^T y$
- Diskrete Fouriersynthese: $\beta \mapsto y = \bar{F}_N^T \beta$
- Eine Berechnung realistischer Beispiele sollte niemals per Hand erfolgen, da zu umfangreich
- Anwendung findet die Fouriertransformation in der Komprimierung von Dateien wie .mp3, .jpeg, indem redundante Frequenzen entfernt werden

# Quellenverzeichnis


- **Q1: Moler, C.B. (2004).** *Numerical Computing with MATLAB.* Society for Industrial and Applied Mathematics (SIAM).
- **Q2: Bartels, S. (2023).** *Numerik 3x9: Drei Themengebiete in jeweils neun kurzen Kapiteln.* Seite 109 - 115. Springer Spektrum.
- **Q3: Forster, O. (2016).** *Analysis 1: Differential- und Integrlrechnung einer Veränderlichen.* 12. Auflage, Springer Spektrum.
- **Q4: Arenz, P. (2005)** *Diskrete und Schnelle Fourier Transformation.* Link: `https://www.math.uni-trier.de/~schulz/Prosem-0405/Arenz.pdf` (zul. abgerufen am 05.01.2024)

# Quellenverzeichnis


- **Q5: Timmann, S. (2007).** *Repetitorium der Funktionentheorie.* Binomi Verlag.
- **Q6: Fourier Transforms.** Link: `https://www.mathworks.com/help/matlab/math/fourier-transforms.html` (zul. abgerufen am 07.01.2024)

# Bildverzeichnis

- **B1:** Link: `https://www.mathe-online.at/mathint/komplex/i.html` (zul. abgerufen am 07.01.2024)

# Weiterführende Ressourcen


- **R1: Nyquist-Shannon-Theorem**
  https://de.wikipedia.org/wiki/Nyquist-Shannon-Abtasttheorem (zul. abgerufen am 07.01.2024)
- **R2: The Discrete Fourier Transform: Most Important Algorithm Ever?**
  https://www.youtube.com/watch?v=yYEMxqreA10 (zul. abgerufen am 07.01.2024)

# Weiterführende Ressourcen


- **R3: Aber was ist die Fourier-Transformation? Eine visuelle Einführung.**
  https://www.youtube.com/watch?v=spUNpyF58BY
  (zul. abgerufen am 07.01.2024)
- **R4: The Most Important Algorithm Of All Time**
  https://www.youtube.com/watch?v=nmgFG7PUHfo
  (zul. abgerufen am 07.01.2024)