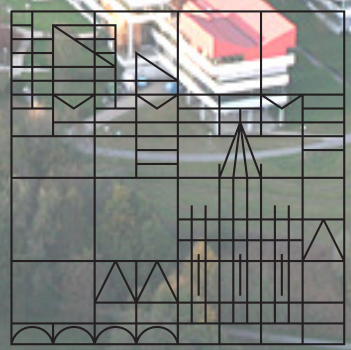
Computerphysik II

Teil 2 - Signalverarbeitung

S. Gerlach

WiSe 2020/21

Universität
Konstanz



Signalverarbeitung

INHALT

1. Entwicklung von Funktionen
2. DFT und Anwendungen
3. Signalverarbeitung und -analyse
4. Bildbearbeitung
5. Bildanalyse

Entwicklung nach Basisfunktionen

Basis: Sinus/Cosinus bzw. kompl. Exponentialfkt.
$\rightarrow$ **Fourierreihe**

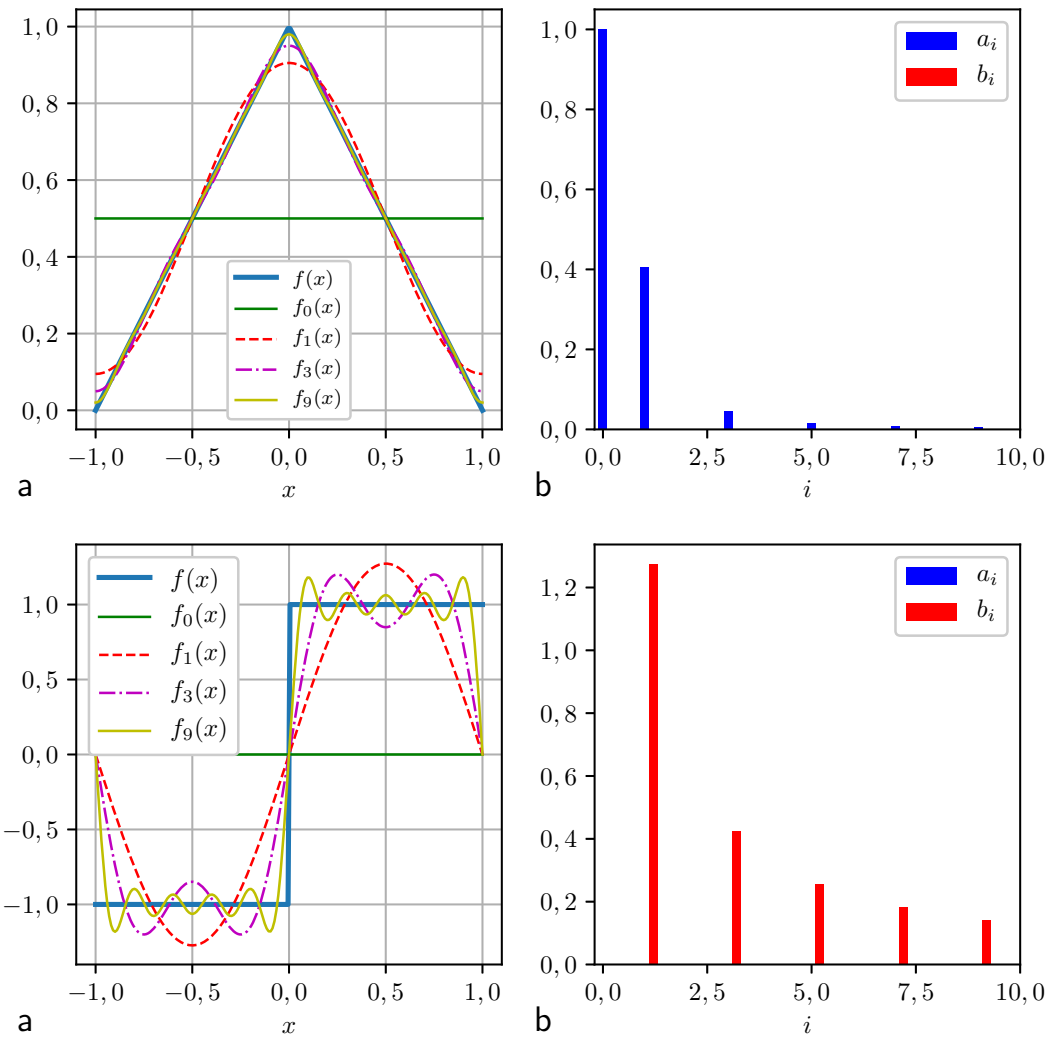
$$f_n(x) = \sum_{j=-n}^{n} c_j e^{ik_j x}, c_j = \frac{1}{L} \int_a^b f(x) e^{-ik_j x} \, dx.$$

* Kovergenz der Fourierkoeff. c_j quadratisch (wenn stetig), sonst linear.
* **Gibbs-Phänomen**: Überschwingen bei Sprungstellen
* Übung: Entwicklung nach Legendre-Polynomen

(s. Buch, Kap. 9.4)

Fourierreihe

Zerlegung in Fourierkoeff. eindeutig. → Transformation Ort in Impuls (k) bzw. Zeit in Frequenz.

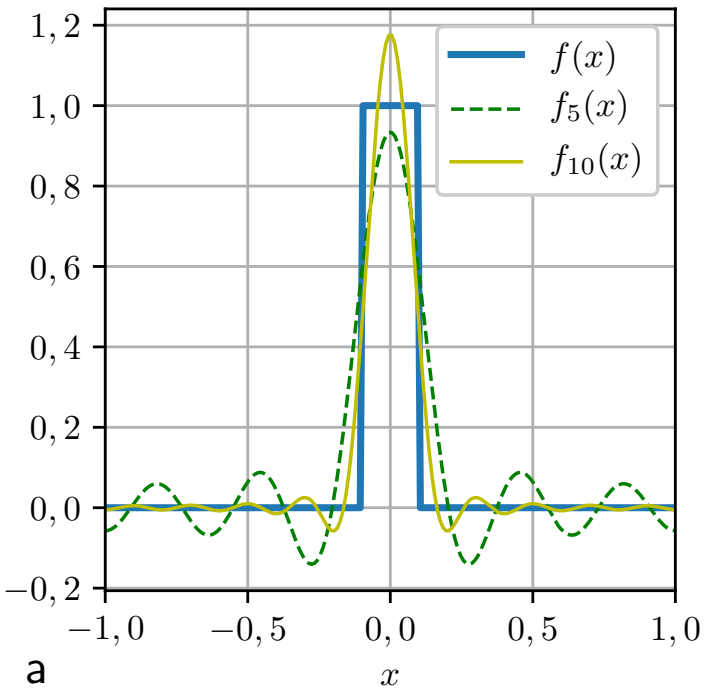


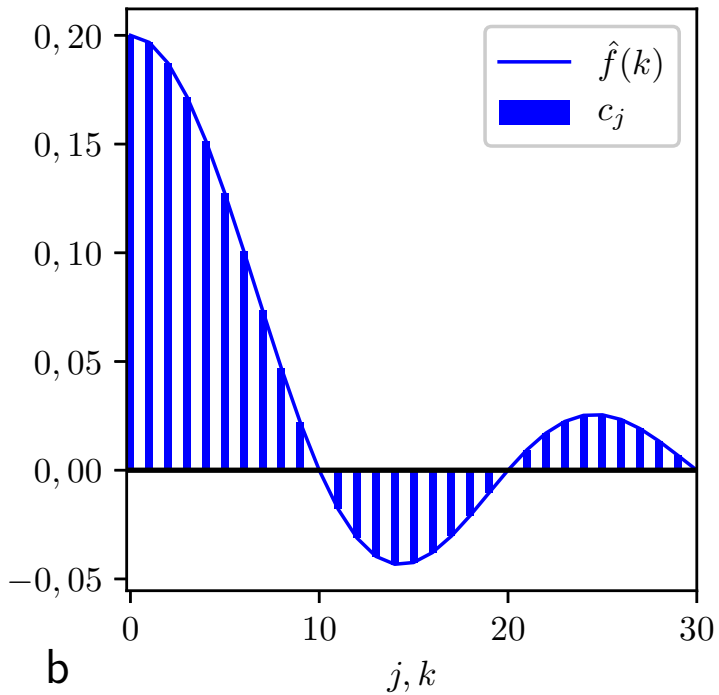
Fouriertrafo

Verallgemeinerung nicht-period. Funktionen: $L/T \to \infty$
Fourierkoeff.

$$c_j = \frac{1}{L}\int_{-L/2}^{L/2} f(x)\mathrm{e}^{-\mathrm{i}k_j x}\,\mathrm{d}x \overset{L\to\infty}{\to} \int_{-\infty}^{\infty} f(x)\mathrm{e}^{-\mathrm{i}kx}\,\mathrm{d}x = \hat{f}(k)$$

(kont.) **Fouriertransformation**





Diskrete Fouriertrafo

Im Computer nur diskrete Werte möglich. D. h. Diskretisierung nötig:
$x = x_0 + k\Delta x \rightarrow$ Diskrete Fouriertrafo

$$\hat{f}_j = \sum_{k=0}^{N-1} f_k e^{-i2\pi\frac{jk}{N}} \quad (j = 0, \cdots, N-1)$$

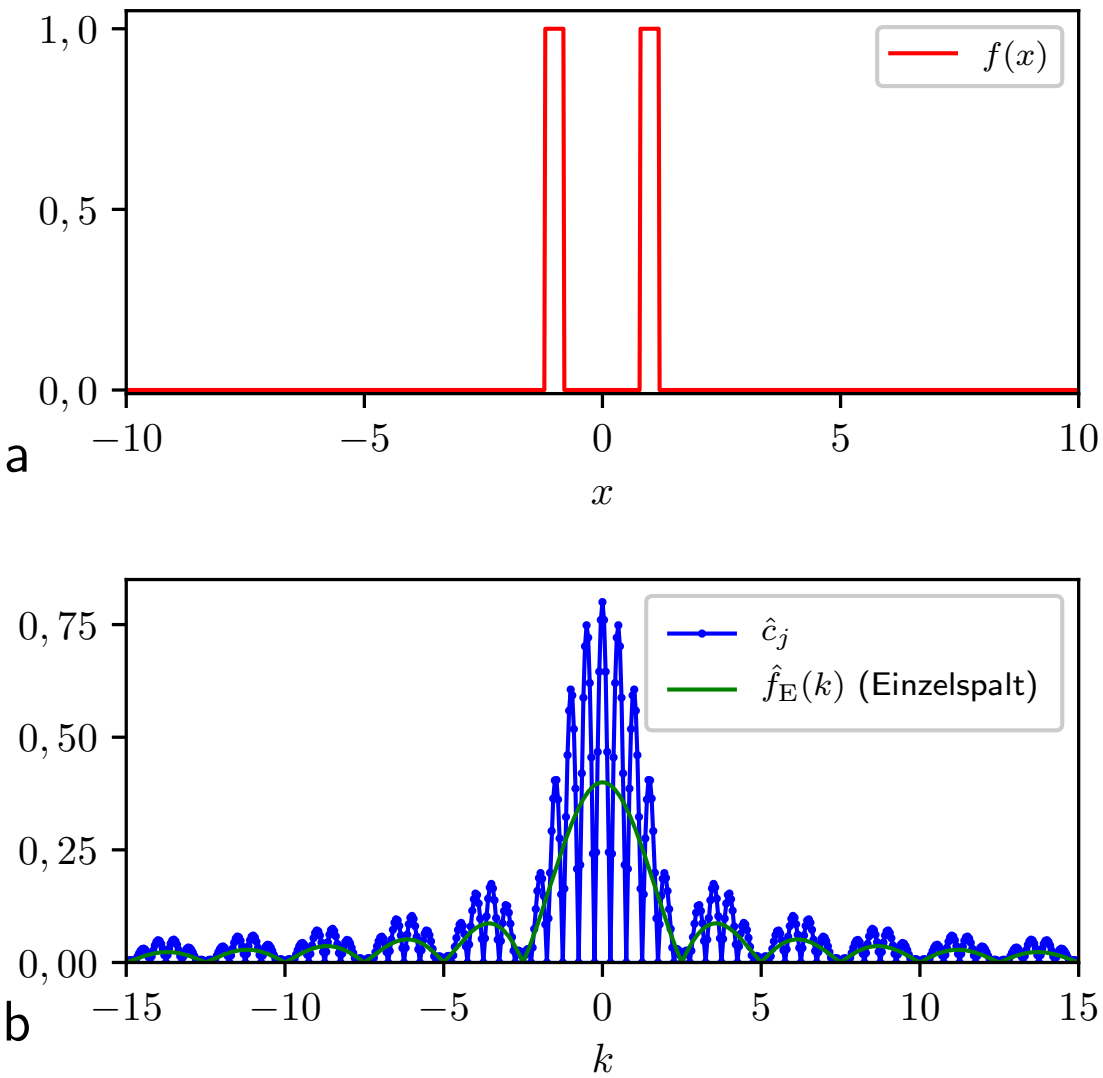
$$f_k = \sum_{j=0}^{N-1} \hat{f}_j e^{i2\pi\frac{jk}{N}} \quad (k = 0, \cdots, N-1)$$

DFT

Berechnung mit Hilfe von FFT-Algorithmen. (Fast-Fourier-Trafo)
`numpy.fft.fft()`, `numpy.fft.ifft()`,
`scipy.fftpack.fft()`, `scipy.fftpack.ifft()`

Anwendung Diskrete Fouriertrafo

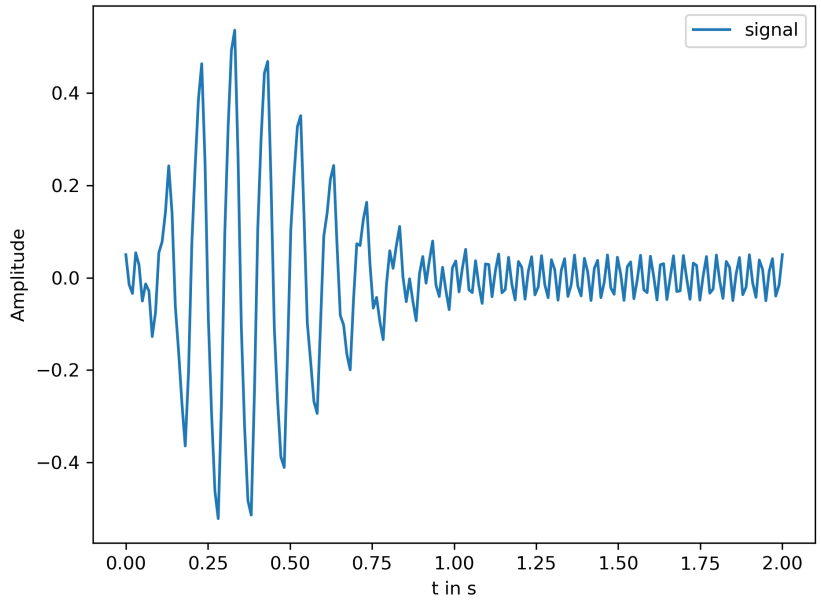
Beugungsmuster eines Doppelspaltes:

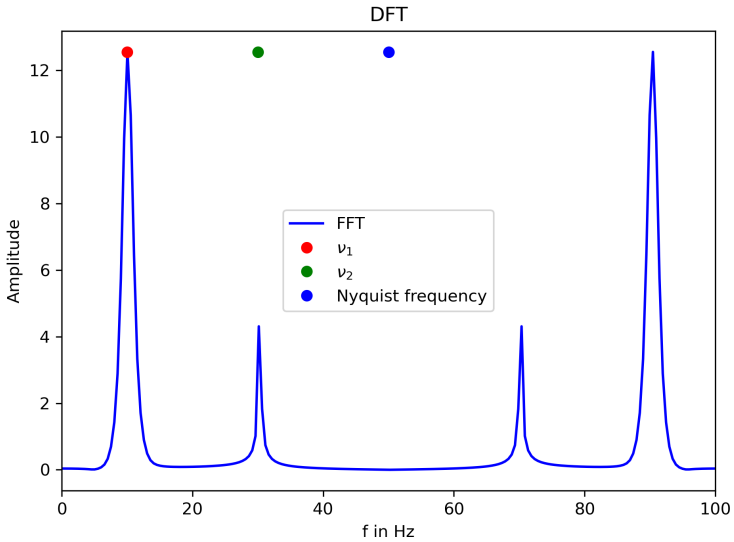


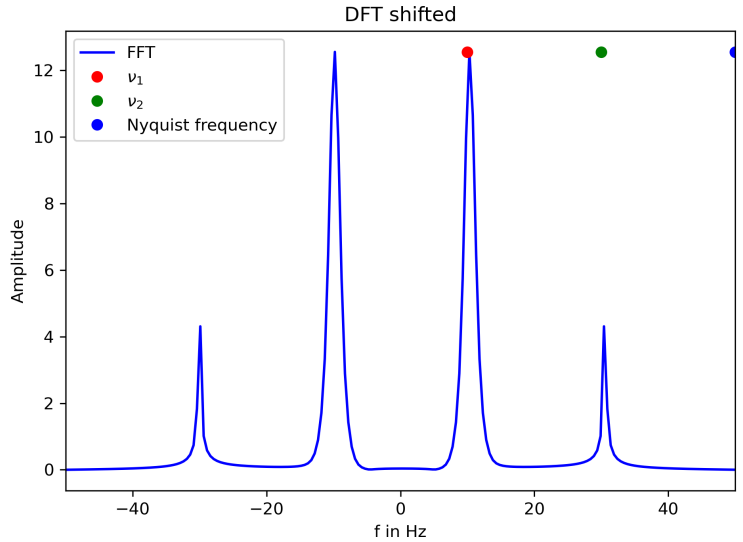
DFT verstehen

Beispiel: **Frequenzanalyse**
ΔT - Abstand der Messungen (Samples)
$\rightarrow$ Samplefrequenz: $\nu = 1/\Delta T$
Maximal messbare Frequenz (3 Messpunkte): $\nu_{\mathrm{Ny}} = \frac{\nu}{2} = \frac{1}{2\Delta T}$
(Nyquist-Frequenz)
Frequenzabstand: $\Delta\nu = \frac{\nu}{N} = \frac{1}{N\Delta T} = \frac{1}{T} \rightarrow$ Je länger die Messzeit, desto genauer die **Frequenzauflösung**.

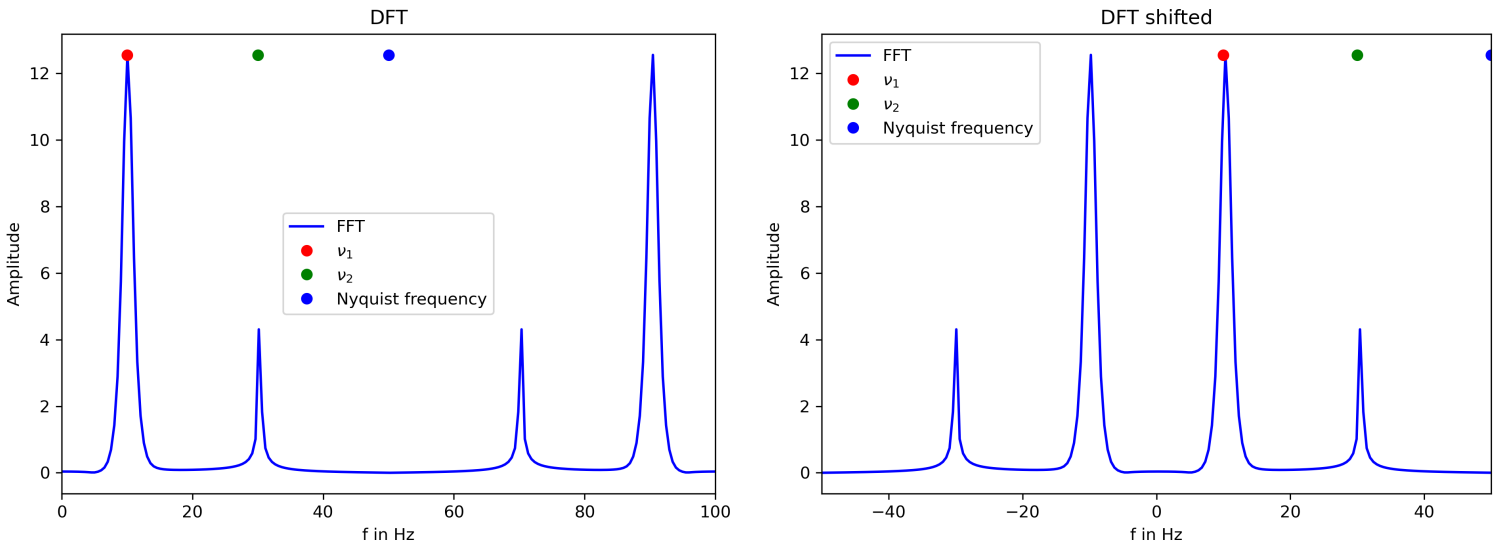
DFT verstehen

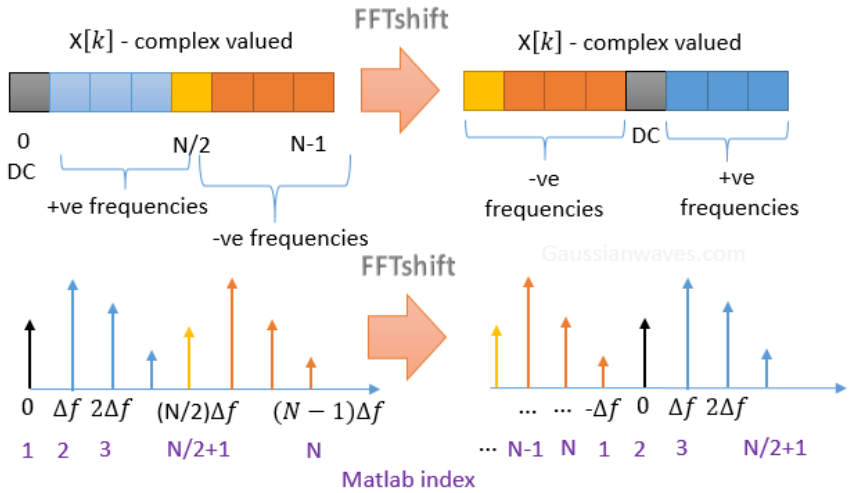






DFT verstehen

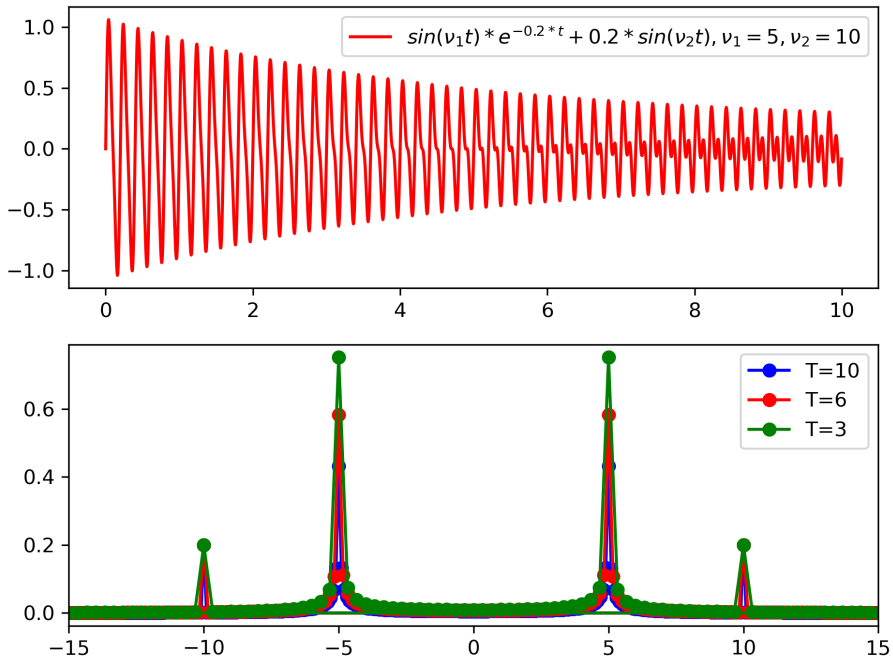


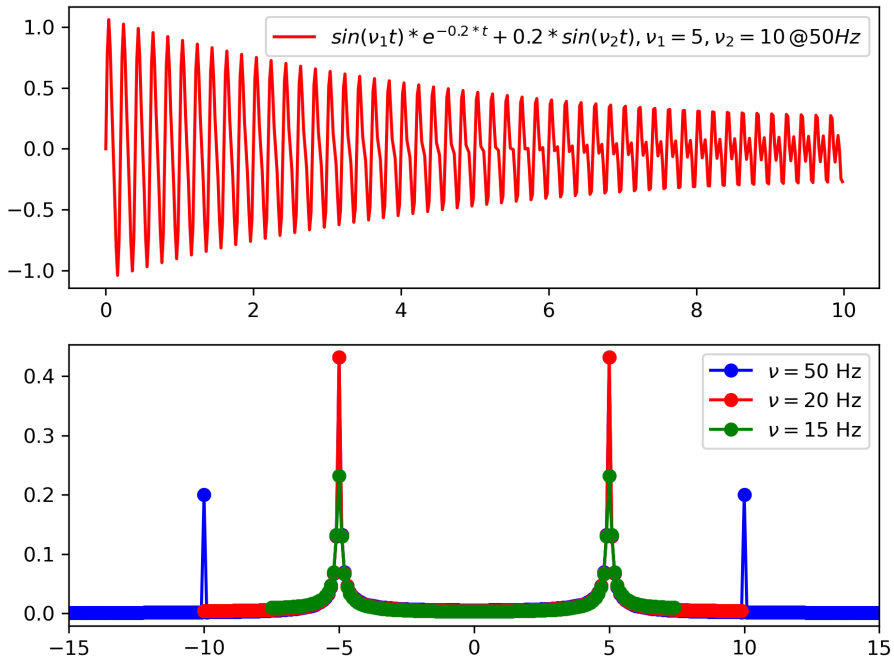


`numpy.fft.fftshift()`, `numpy.fft.fftfreq()`

DFT verstehen

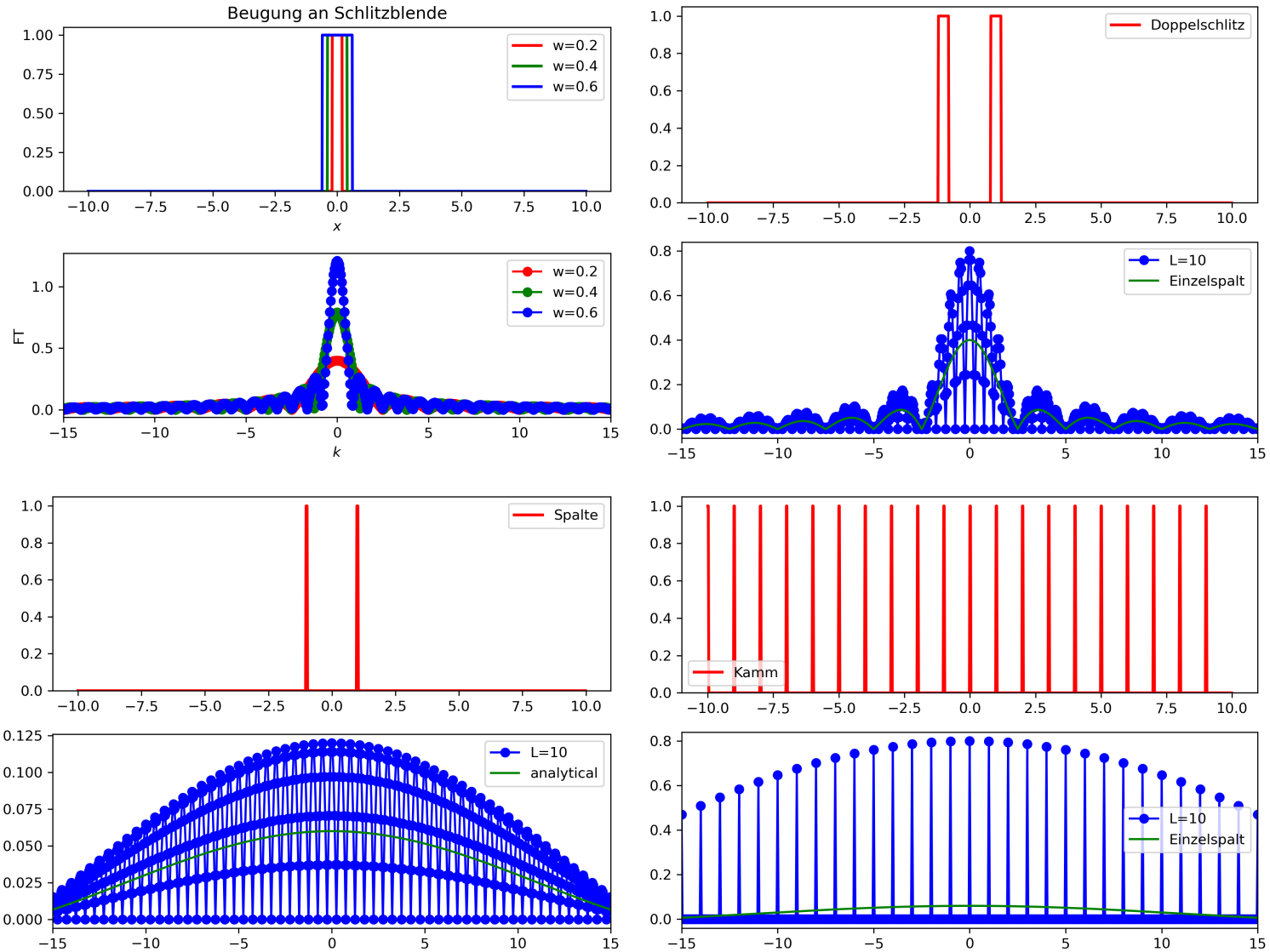
Variation: Messzeit, Samplefrequenz






```
1 dt = T/N
2 x = fftfreq(N, dt)
3 Y = fft(y) * 2.0/N
4 pl.plot(x,abs(Y))
```

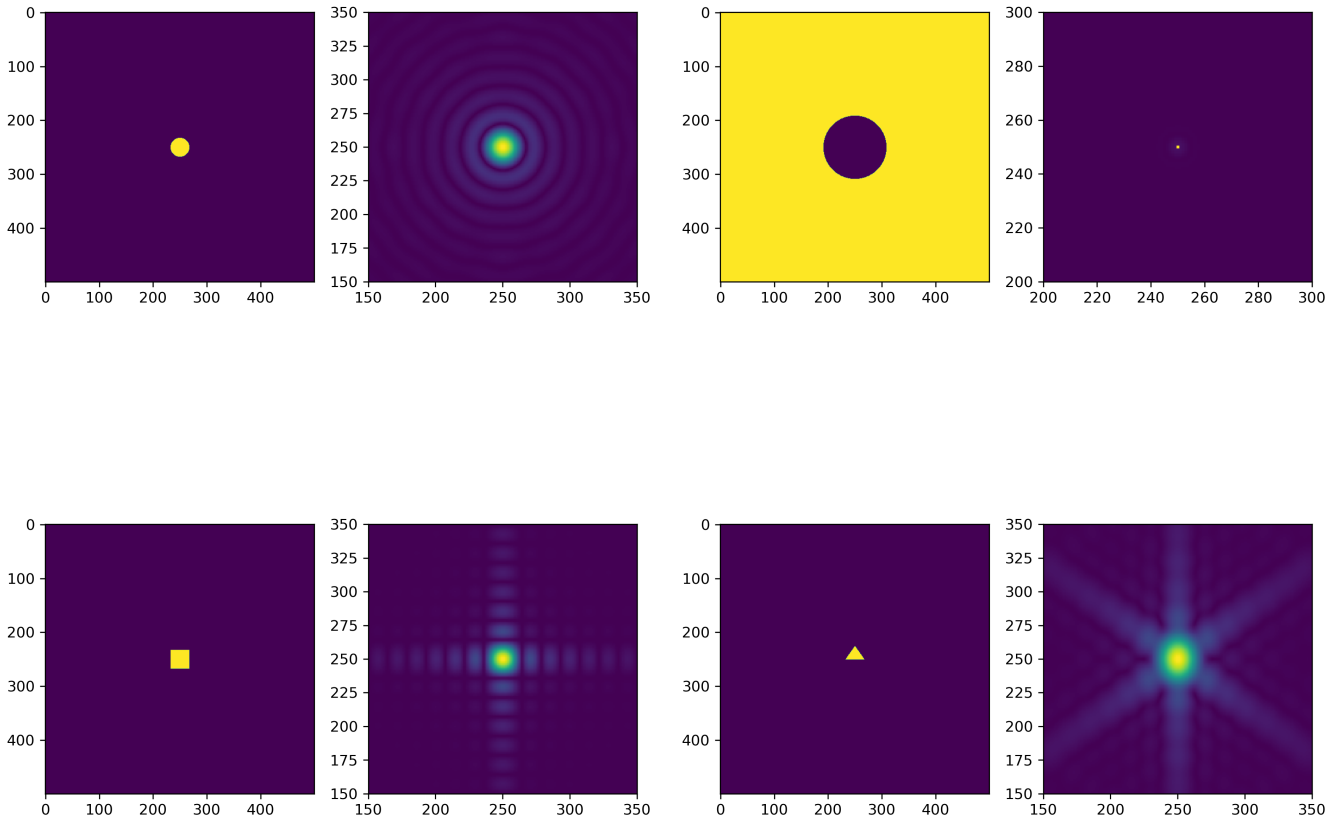
Anwendung: Beugung



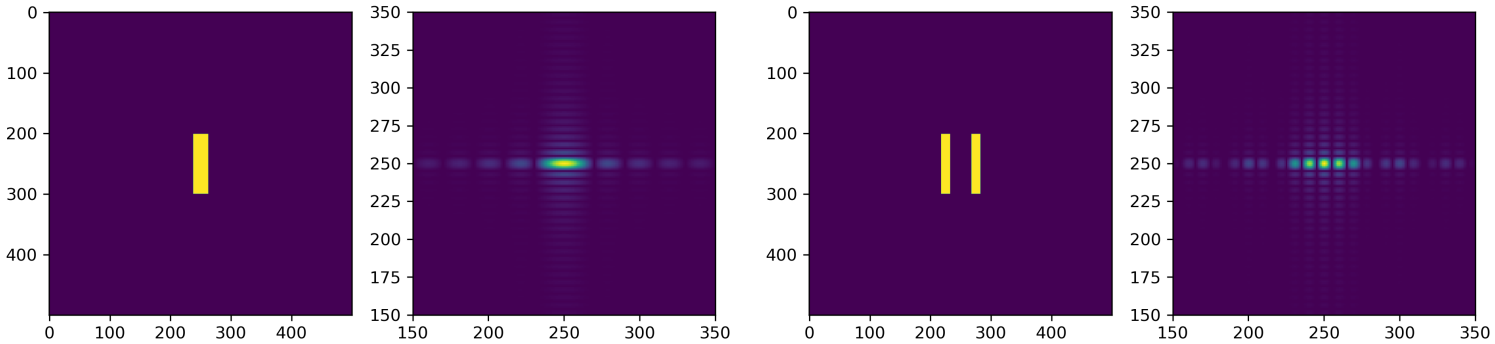
Anwendung: 2D Beugung

2D-DFT:

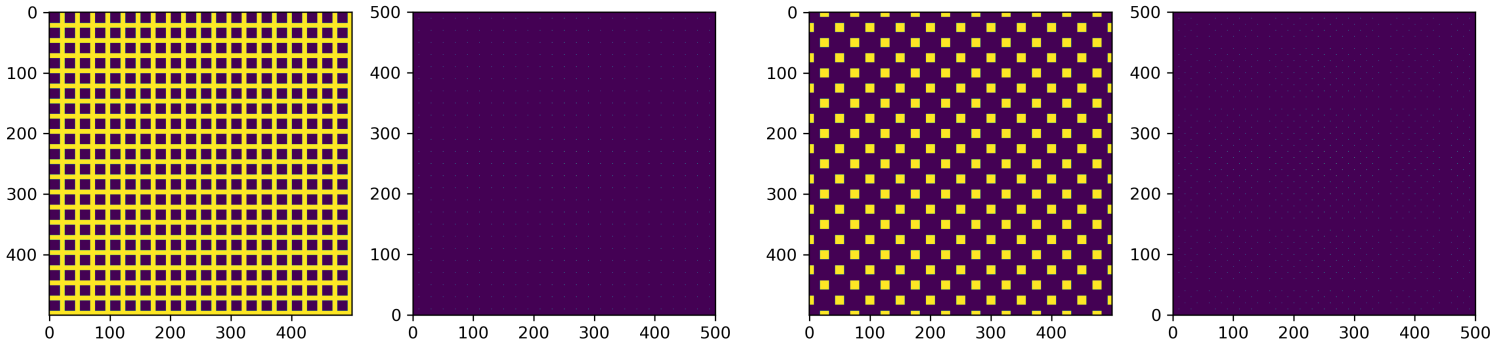
$$\hat{a}_{kl} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} a_{mn} e^{-i2\pi\left(\frac{mk}{M} + \frac{nl}{N}\right)}$$



Anwendung: 2D Beugung



`log |Y + 1|` (zoom!) :



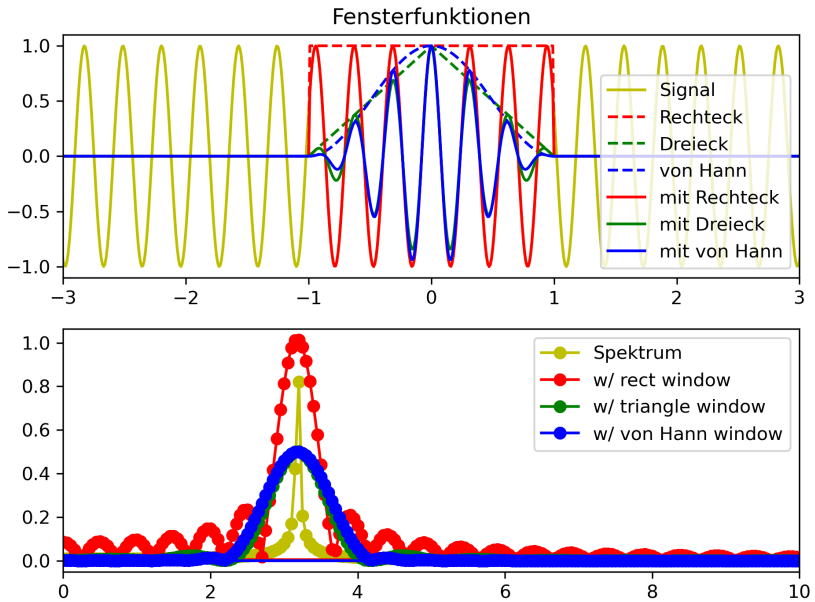
Übersicht DFT berechnen

Python:
`fft()/ifft()`, `fft2()/ifft2()`, `fftn()/ifftn()`
`fftshift()/ifftshift()`, `fftfreq()`
Optimiert: `numpy.fft`, `scipy.fftpack`, `pyfftw`
C:
FFTW - Fastest Fourier Trafo in the West
MKL - Math Kernel Library von Intel
@ GPU: cufft (CUDA)

Signalverarbeitung und -analyse: Wichtige Effekte

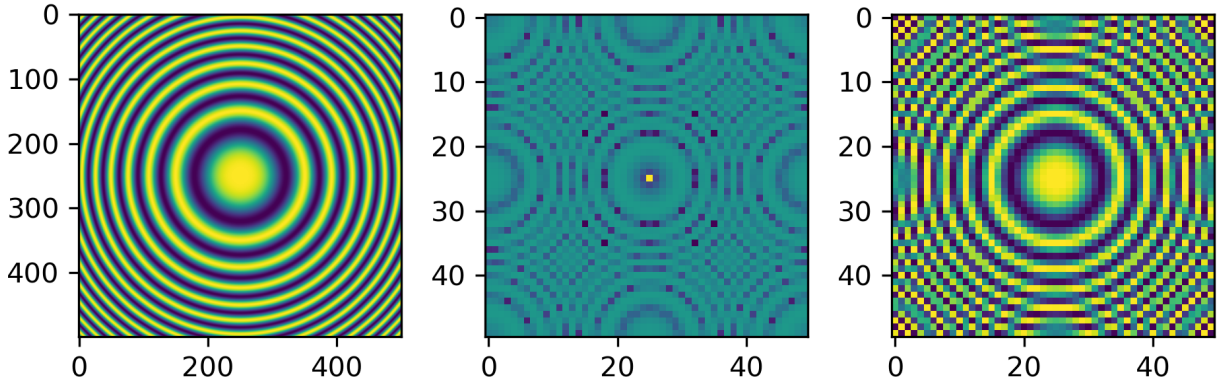
BUCH: Kapitel 18.3

Leck-Effekt: In der Realität sind Signale (Messungen) immer endlich und damit z.B. im Frequenzbereich über einen Bereich verteilt. Dies entspricht praktisch einem Rechteckfilter in der Zeit, der zu einem schmalen Hauptmaximum, aber auch vielen Nebenmaxima im Frequenzbereich führt. Allerdings lassen sich diese sog. Fensterfilter optimieren. Beispiele: Dreieckfenster, von-Hann-Fenster, ...



Signalverarbeitung und -analyse: Wichtige Effekte

Alias-Effekt (Stroboskop-Effekt): Endliche Samplerate führt zum Abschneiden hoher Frequenzen (Abtasttheorem, vgl. Nyquist-Frequenz)
$\rightarrow$ Interferenz zw. Signal und Sample-/Abtastfrequenz
Bei Resonanz: Stroboskop-Effekt, Moire-Muster, etc.
Auch bekannt aus Filmen: Wagon-Wheel-Effekt (Einfach mal nach Videos unter dem Namen suchen)

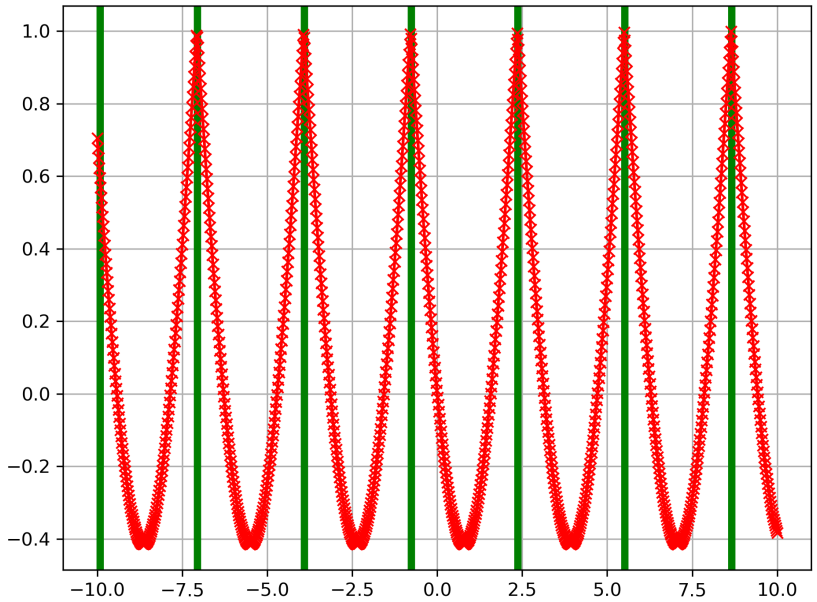


Signalverarbeitung und -analyse: Beispielmethoden

"Detrend": Entferne konstanten Untergrund bzw. Trend in den Daten. Dafür einfach lineare Anpassung anziehen: scipy.signal.detrend()

"Peak Find": Viele Signale enthalten Resonanzen (Spektren, etc.). Oft gesucht: Linienposition und -breite

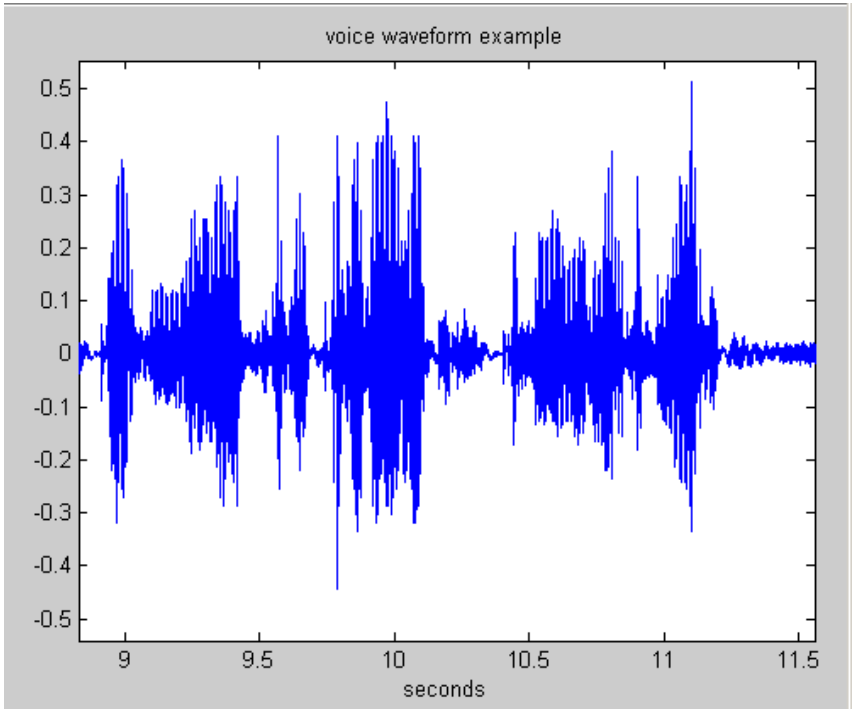
Dazu: jeweilige Anpassung mit Linienform oder andere Methoden (s. `scipy.signal.find_peaks_cwt()`)

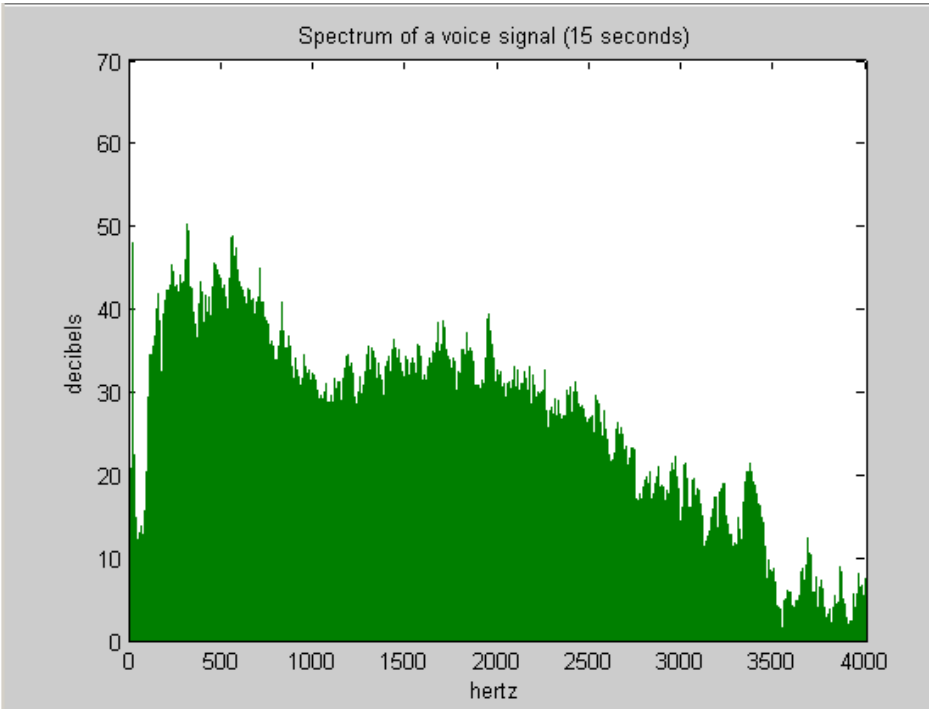


Signalverarbeitung und -analyse: Beispielmethoden

Periodigramm: gleitende DFT eines Zeitsignals (mit Fenster) $\rightarrow$ power-spectral-density ($= \log |Amplitude|^2$)

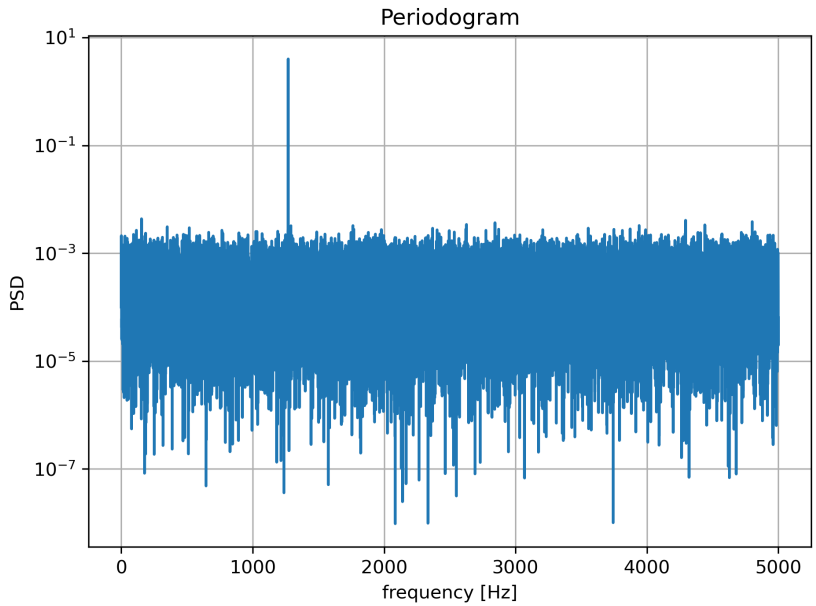
`scipy.signal.periodigram()`

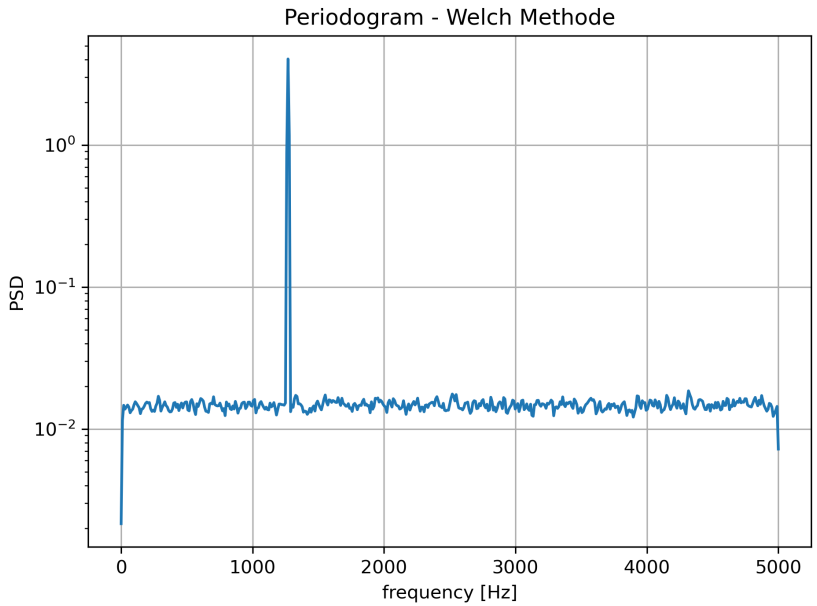




Signalverarbeitung und -analyse: Beispielmethoden

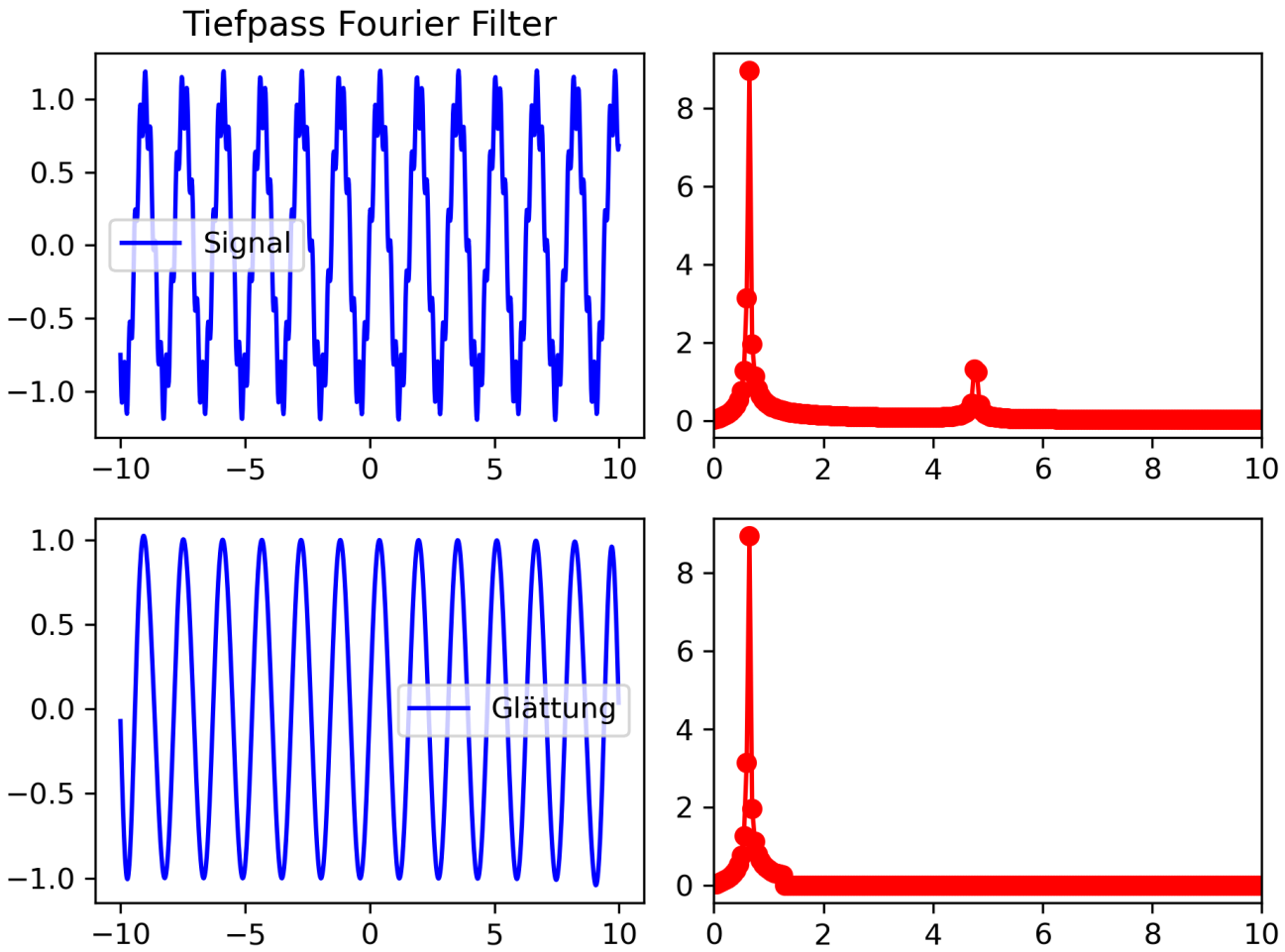
Optimiertes Periodigram: **Welch-Methode**:
Überlappende Fensterfunktionen und Mittelung



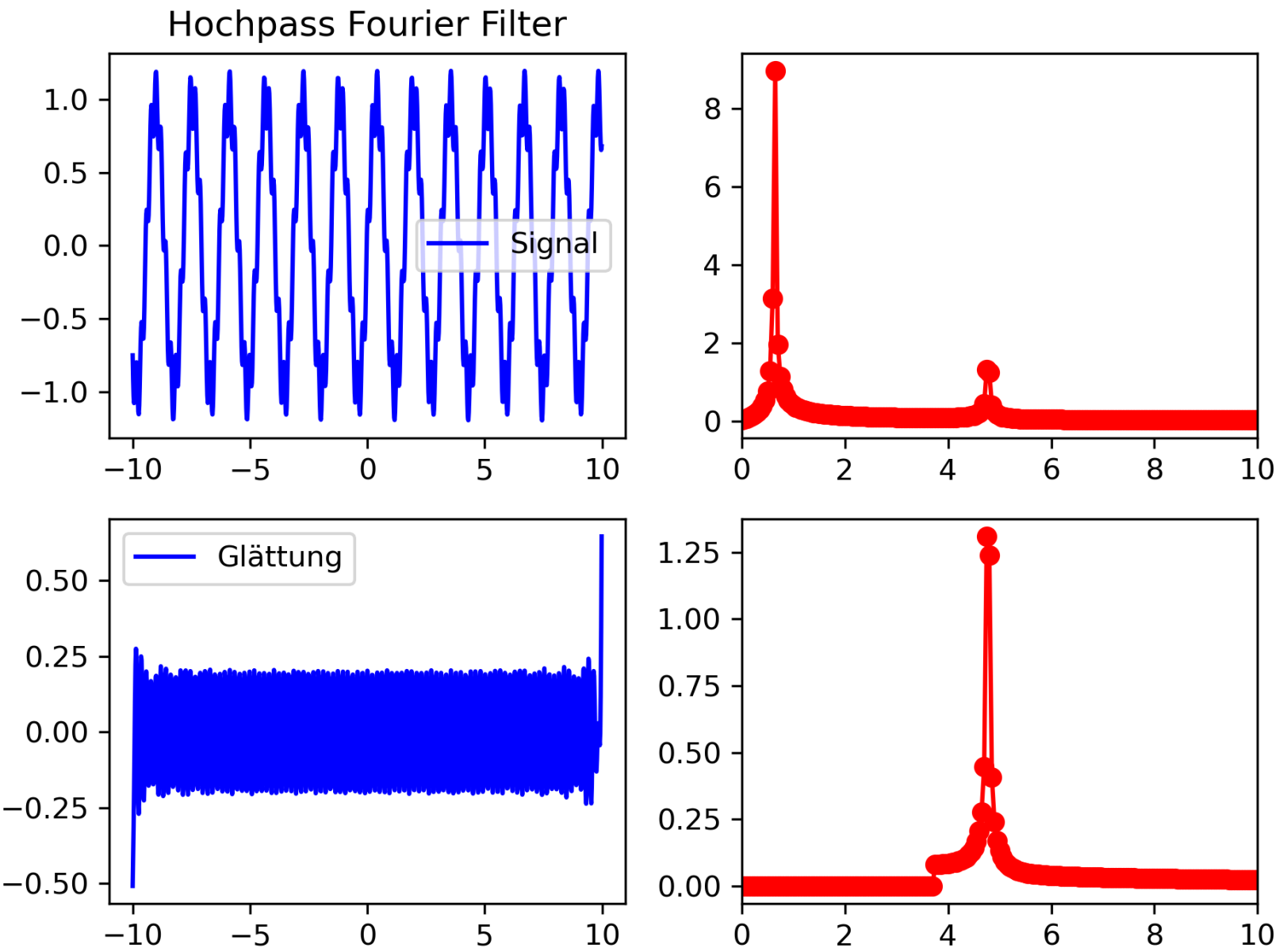


Signalverarbeitung und -analyse: Fourier-Filter

Idee: Verwende Filter im Fourier-Raum (Frequenzbereich)
Anwendungen: Hochpass-, Tiefpass-, Bandpass-, Bandblock-Filter

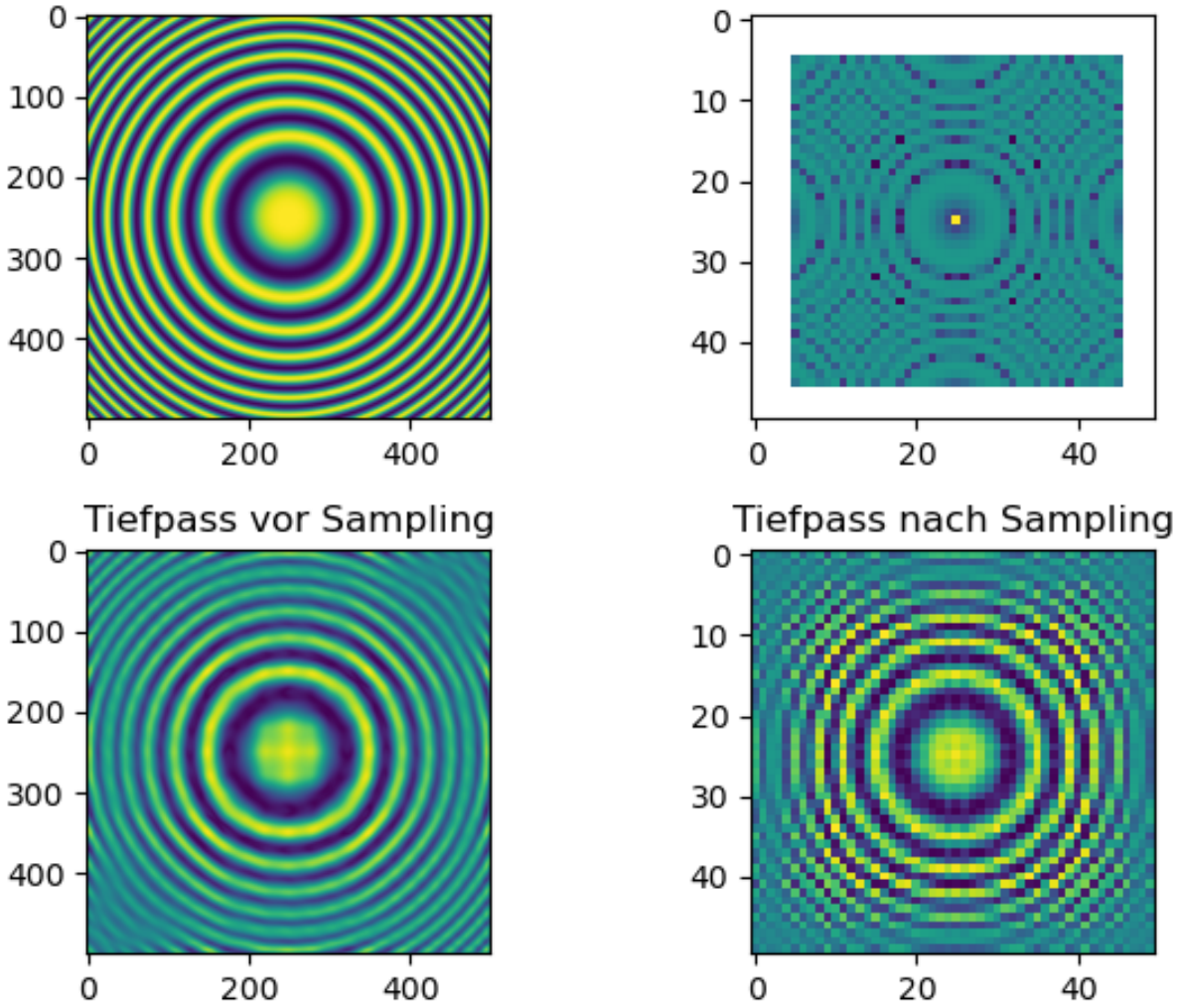


Signalverarbeitung und -analyse: Fourier-Filter



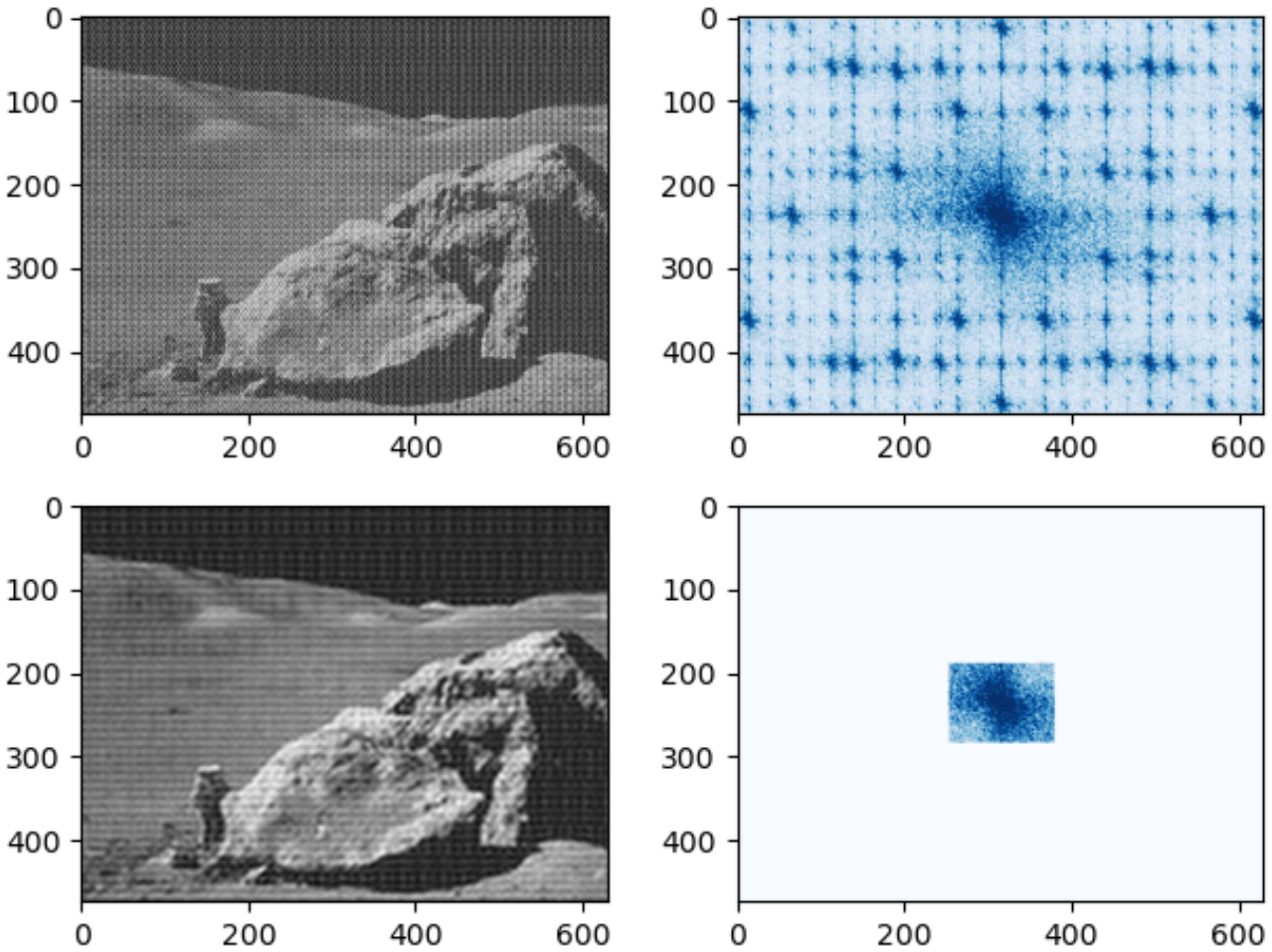
Signalverarbeitung und -analyse: Fourier-Filter

Anti-Alias: Aliaseffekt durch Filter verringern
Auch bekannt bei Schriften (Raster) oder Monitoren (Maske)



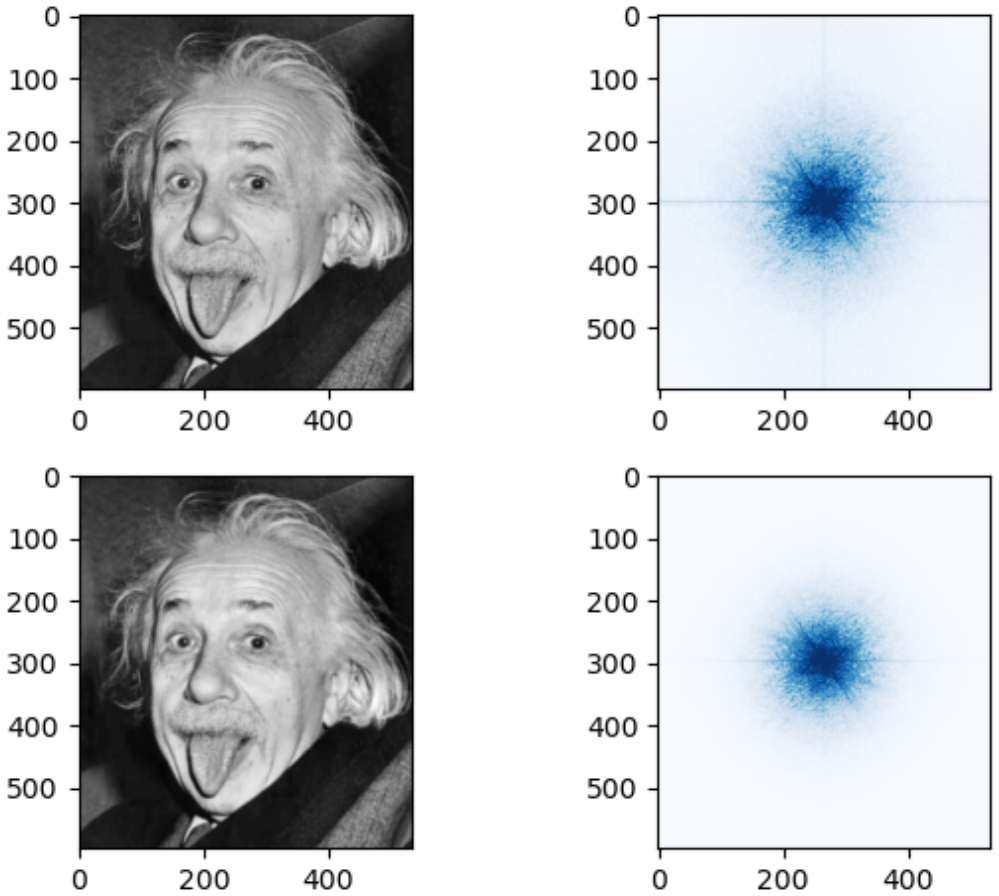
Signalverarbeitung und -analyse: Bildbearbeitung

Viele Methoden, die Fourierfilter verwenden:
Beispiel 1: **Streifenfilter**
Entferne die Anteile im Fourierbild, die zu den Streifen führen



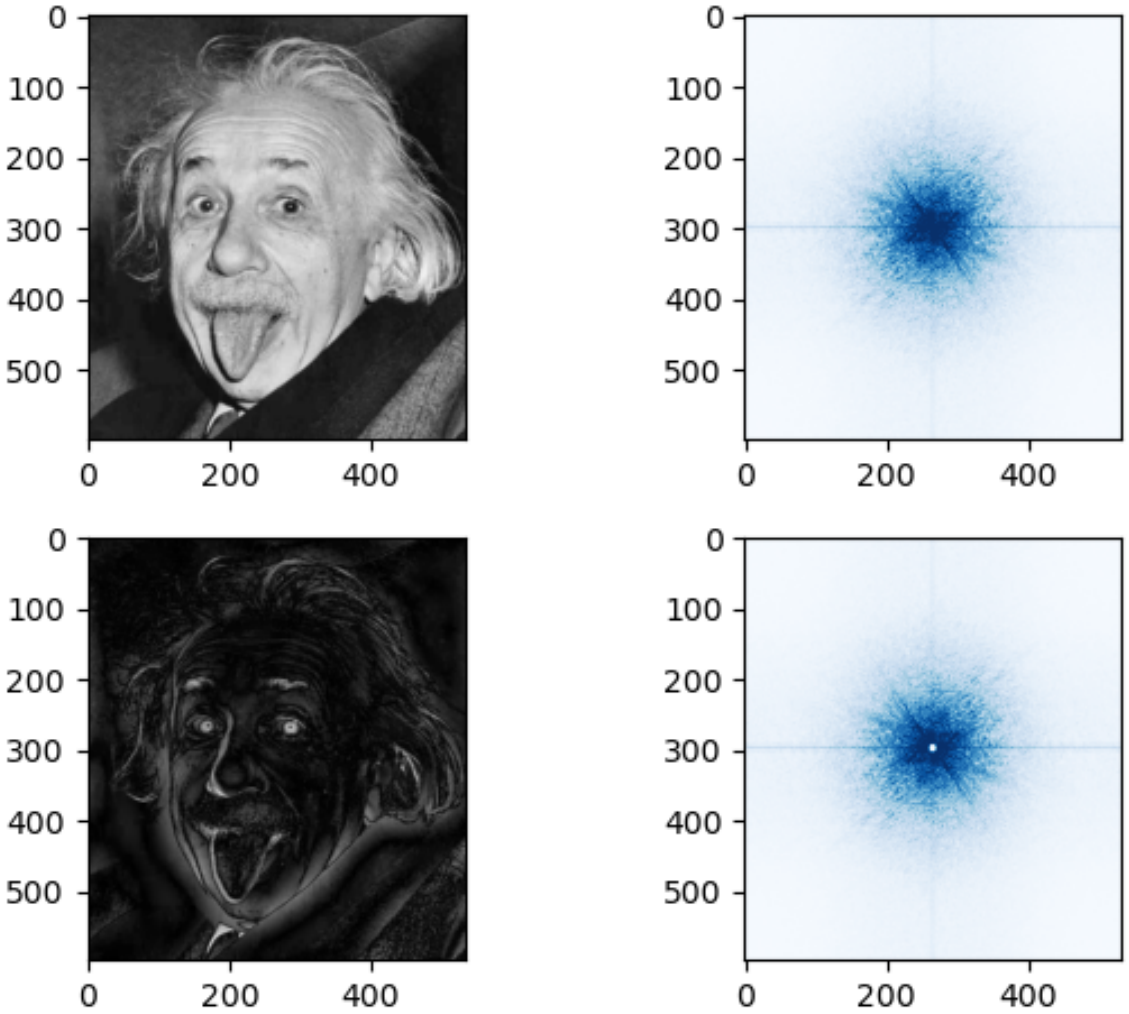
Signalverarbeitung und -analyse: Bildbearbeitung

Beispiel 2: **Weichzeichnen**
Entferne hochfrequente Anteile (Kanten, kleine Strukturen, etc.)
Um Aliaseffekte zu vermeiden ("Ringing"): verwende "weiche" Filter, z.B. Gauss



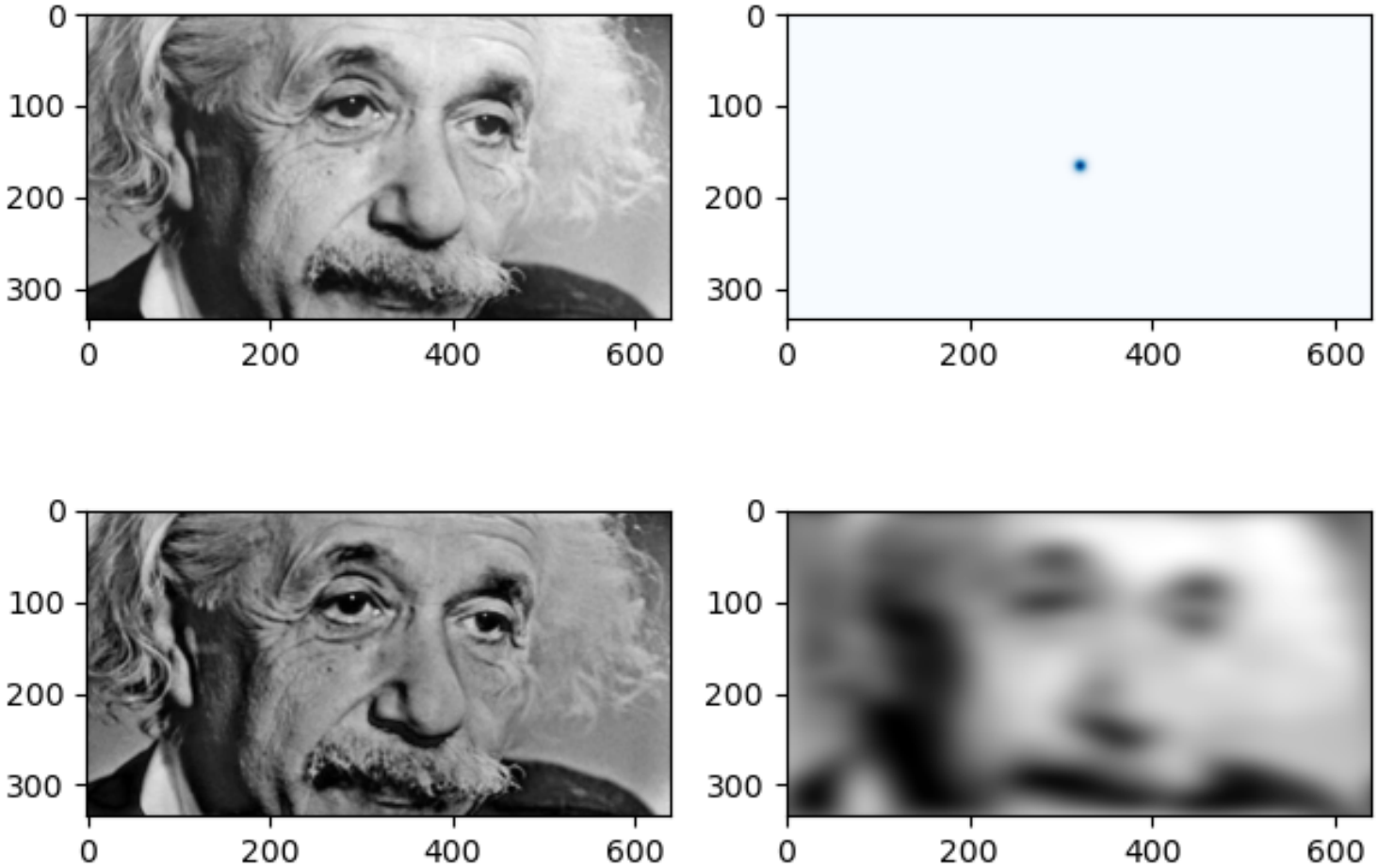
Beispiel 3: **Kantenfilter**

Verwende Hochpassfilter um Umrisse (Kanten) zu sehen



Beispiel 4: **Schärfen**

Idee: Weichzeichen des Bildes und Abziehen vom Originalbild



Signalverarbeitung und -analyse: Faltung und Anwendung

Faltung:

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

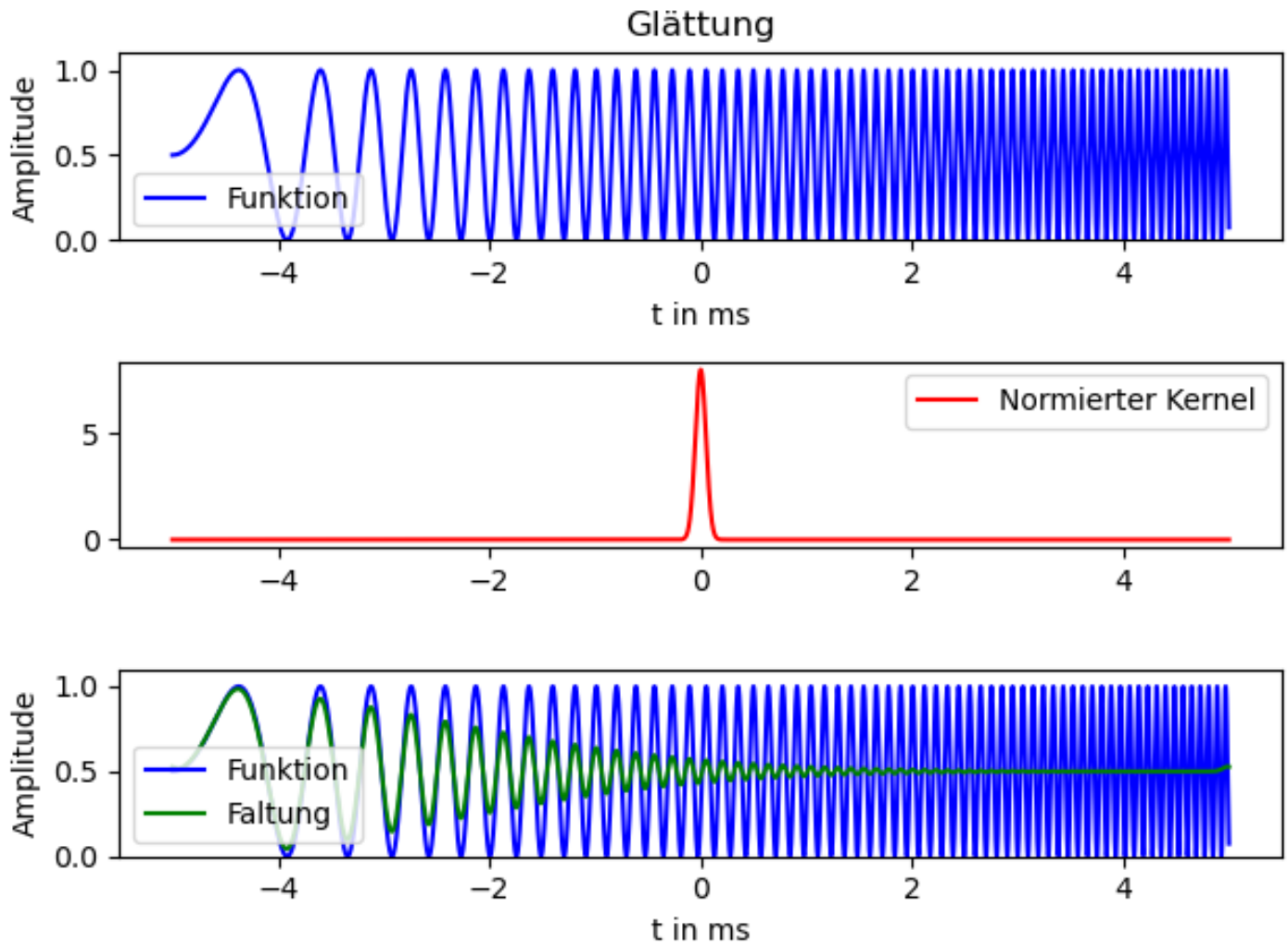
$$(f * g)_j = \sum_k f_k g_{j-k}$$

Anschaulich: Eine (gespiegelte) Gewichtsfunktion ("kernel") g wird über die Funktion f geschoben. (siehe Animation auf `https://de.wikipedia.org/wiki/Faltung_(Mathematik)`)
Zur Berechnung: nutze **Faltungstheorem**

$$f * g = \mathcal{F}^{-1}(k\mathcal{F}\{f\} \cdot \mathcal{F}\{g\})$$

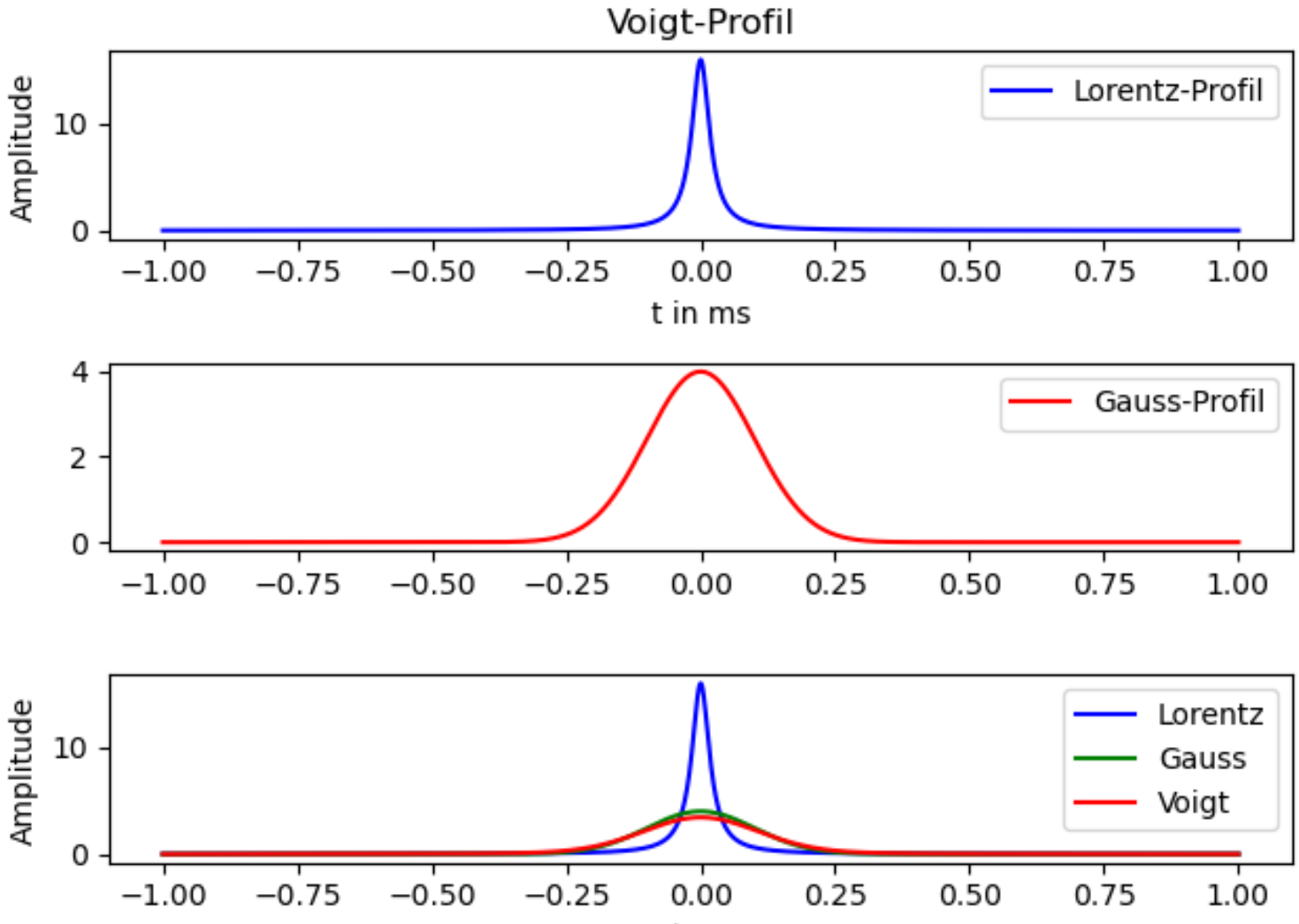
FFT ist mit $\mathcal{O}(N \log N)$ deutlich schneller als direkte Summe (s.o.) mit $\mathcal{O}(N^2)$
Typische Anwendung: Glättung durch Faltung mit Gauss-Kernel

Signalverarbeitung und -analyse: Faltung und Anwendung



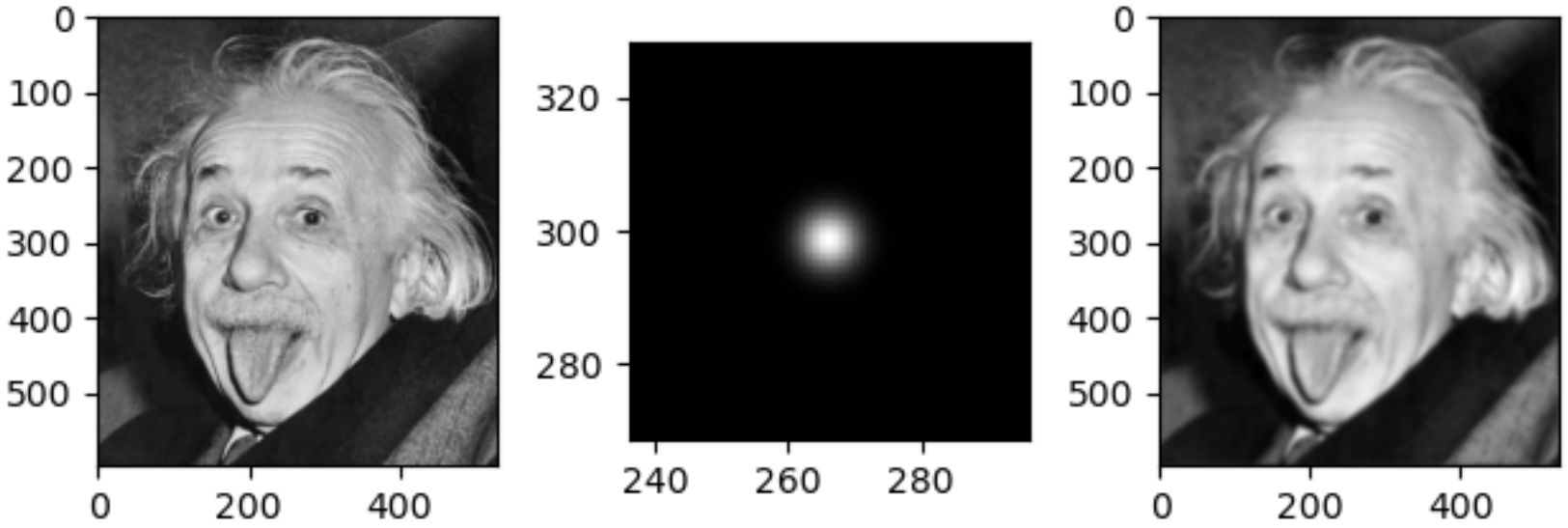
`scipy.signal.convolve()`, `scipy.signal.fftconvolve()`

Voigt-Profil: Faltung von Dopplerverbreiterung (Gauss) mit natürlicher Linienbreite (Lorentz)



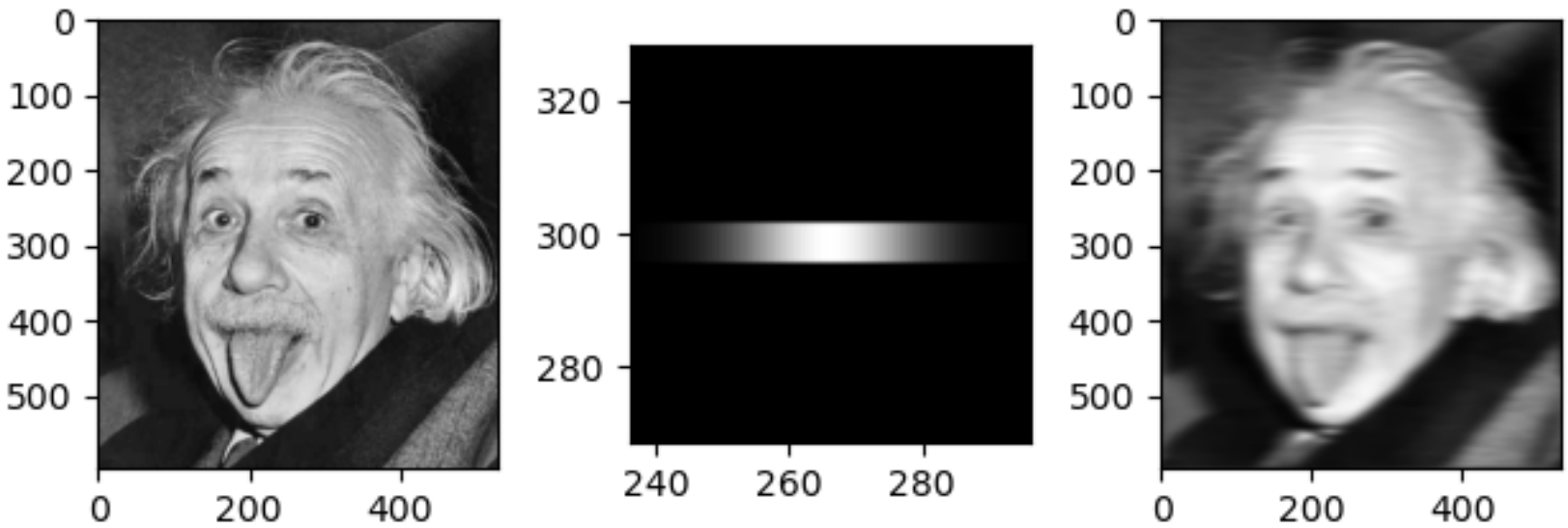
Signalverarbeitung und -analyse: Faltung und Anwendung

In 2D: Kernel wird auch "point spread function" (PSF) genannt, da jeder Punkt mit dem Kernel multipliziert wird.
2D-Gauss-Kernel: **Gauss-Glättung**



Signalverarbeitung und -analyse: Faltung und Anwendung

Verschiedene Kernel führen zu unterschiedlichen Effekten. Beispiel: **Blur**



Signalverarbeitung und -analyse: Faltung und Anwendung

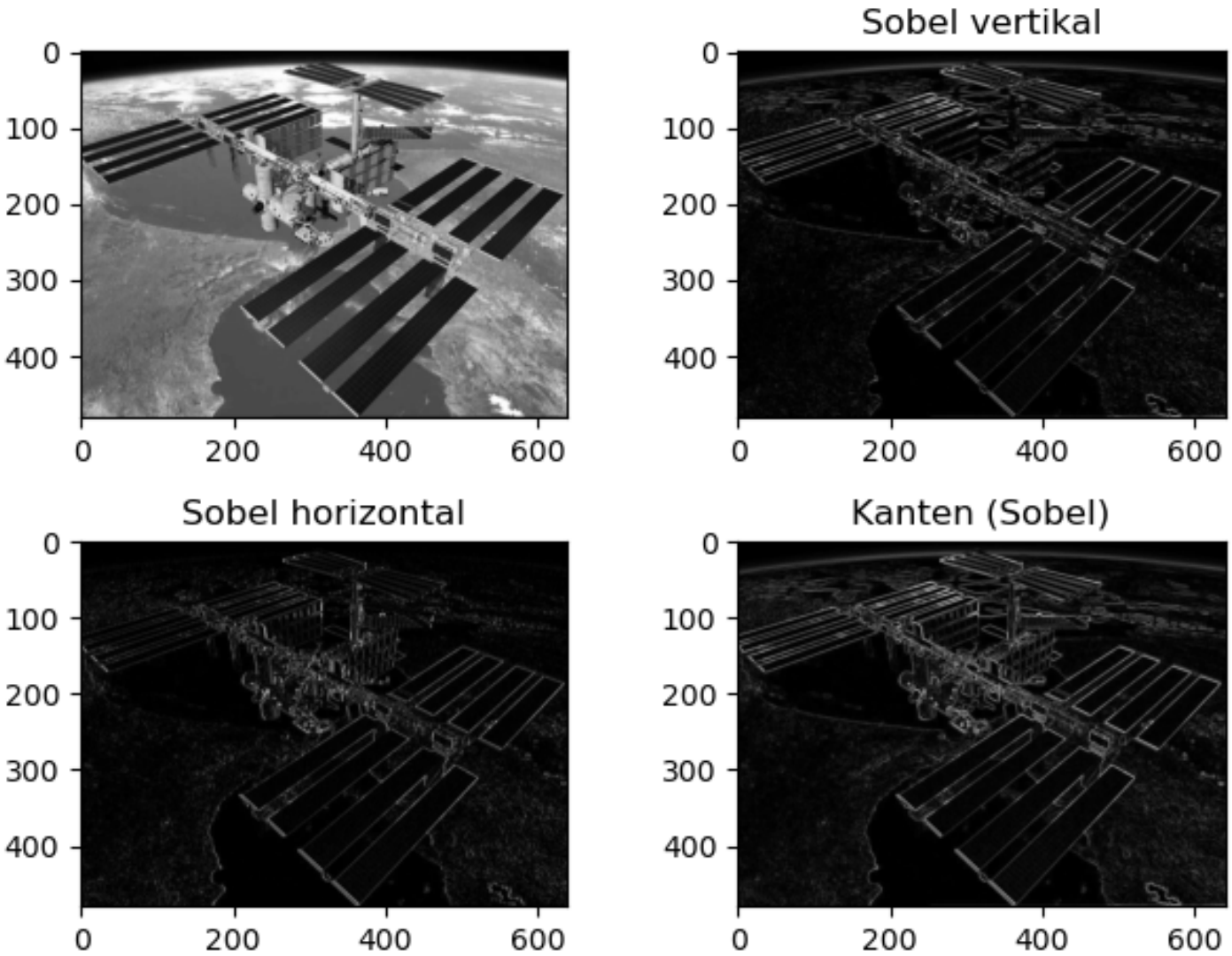
Kernel $(1\ -1) \equiv (f * g)_n = f_n - f_{n-1}$: Vorwärtsableitung
Kernel $(\frac{1}{2}0\ -\frac{1}{2}) \equiv (f * g)_n = \frac{1}{2}(f_{n+1} - f_{n-1})$: Zentrale Ableitung
Kernel $(\frac{1}{3}\frac{1}{3}\frac{1}{3}) \equiv (f * g)_n = \frac{1}{3}(f_{n+1} + f_n + f_{n-1})$: Mittelung/Glättung

Kantenerkennung: Faltung mit Kernel $(1\ -1)$ bzw. $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$ oder besser **Sobelfilter** (Ableitung + Glättung)

$$S = \sqrt{S_x^2 + S_y^2}, S_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}, S_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

Signalverarbeitung und -analyse: Faltung und Anwendung

Sobelfilter (s. `ndimage.sobel()`):

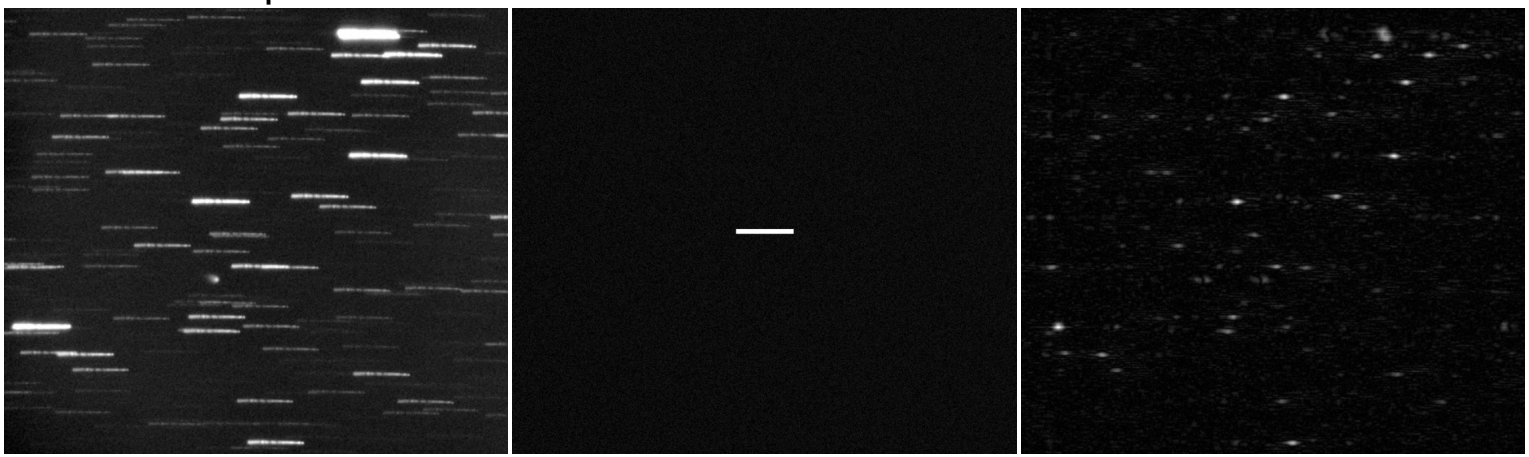


Dekonvolution: Zurückrechnen des Originalbildes bei bekanntem/geratenem Kernel

$$B = \mathcal{F}^{-1}\left\{\frac{\mathcal{F}\{S\}}{\mathcal{F}\{K\}}\right\}.$$

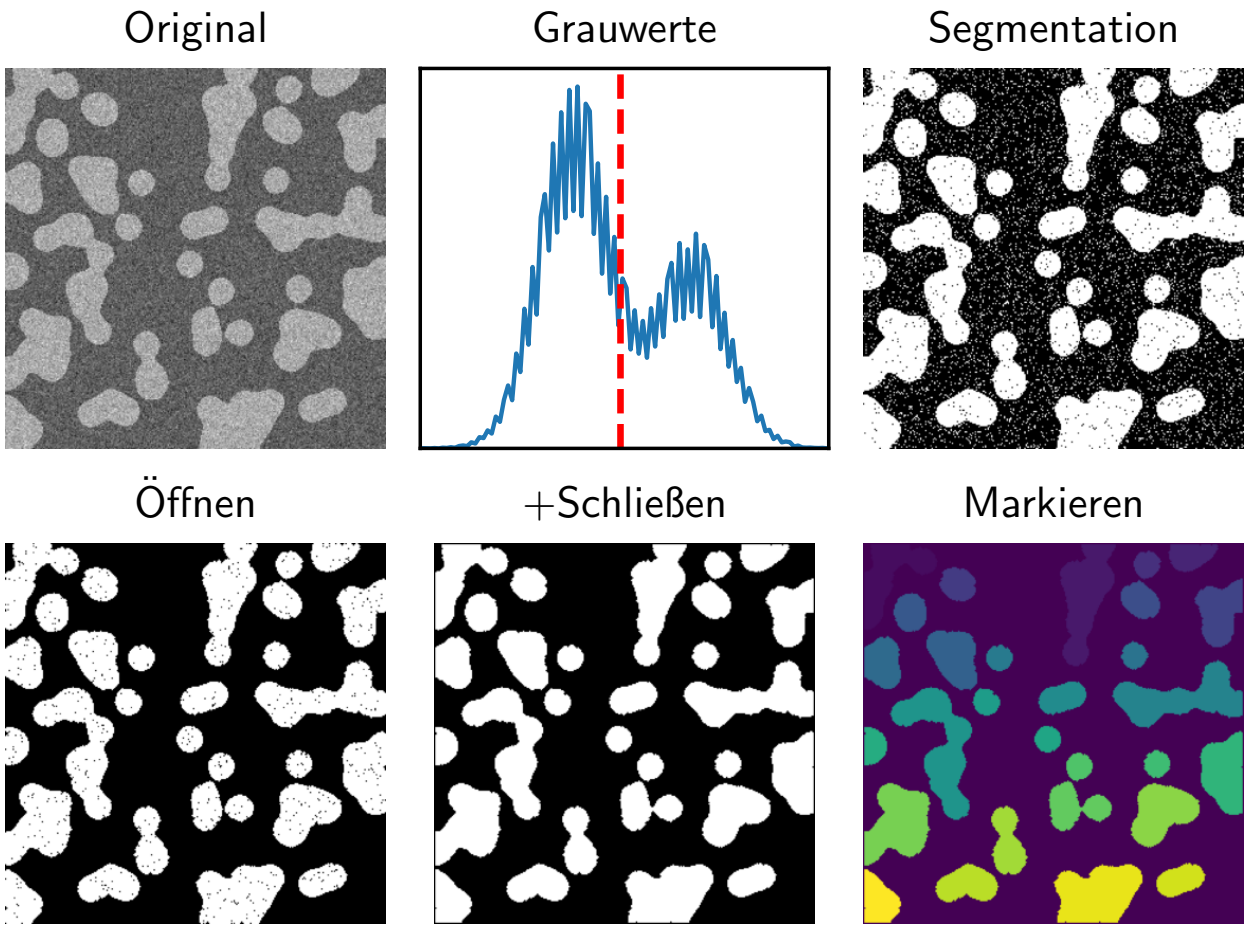
Beispiele: Bewegungskorrektur (s. Übung)

Sternspuren | Kernel | Dekonvolution



Signalverarbeitung und -analyse: Bildanalyse

Informationen aus Bildern extrahieren. Beispiel: Segmentation, Morphologie und Granulometrie
$\rightarrow$ s. Buch Kap. 18.3.4



Korrelation:

$$(f \star g)(t) = \int_{-\infty}^{\infty} f^*(\tau) g(\tau + t)\, \mathrm{d}\tau$$

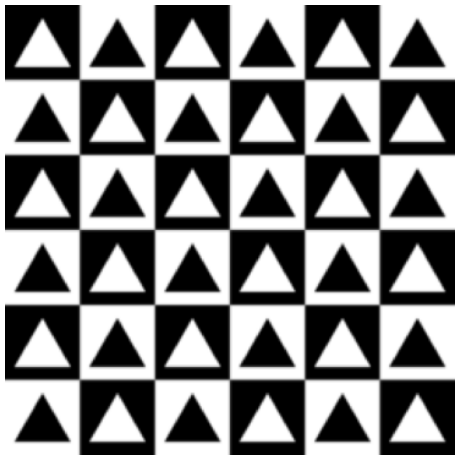
$$(f \star g)_j = \sum_k f_k^* g_{k+j}.$$

Gleitender Mittelwert von f mit Gewichtung g. Hohe Werte bei Übereinstimmung: **Mustererkennung**
$\mathcal{F}\{f \star g\} = k\mathcal{F}\{f^*(-t)\} \cdot \mathcal{F}\{g\} = k\mathcal{F}^*\{f\} \cdot \mathcal{F}\{g\}$.
Autokorrelation: Korrelation mit sich selbst, d.h. Erkennung von periodischen Mustern
`scipy.signal.correlate()`, `scipy.signal.correlate2d()`

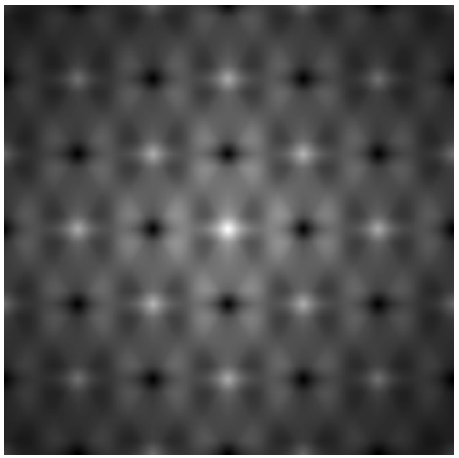
Signalverarbeitung und -analyse: Bildanalyse

Beispiel:

Bild

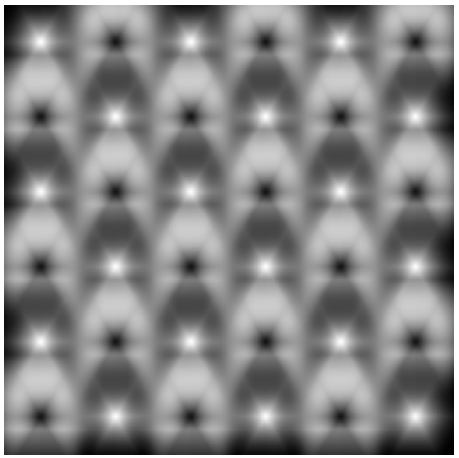


Autokorrelation



Muster

Kreuzkorrelation



Signalverarbeitung und -analyse: Bildanalyse

Anwendung: Mustererkennung durch Kreuzkorrelation