Geodatenanalyse 2

Termin: Big Data 2 - Modul 2

Frequenzanalyse von Zeitreihen: Wichtige Eigenschaften der Fourier Analyse

Ca. 20-30 Minuten

Inhalt

- Zusammenhang zwischen Zeitdauer und Frequenzauflösung
- Maximale Frequenzauflösung
- Der Leck-Effekt
- Fensterfunktionen
- Anwendung des Von-Hann Fensters auf die Meeresspiegel-Zeitreihe
- Weitere Methoden der Frequenzanalyse

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Zusammenhang zwischen Zeitdauer und Frequenzauflösung

- Die Frequenzauflösung der FFT ist ein sehr wichtiges Kriterium

- Sie bestimmt, wie genau man Frequenzkomponenten identifizieren kann

Frequenzauflösung

$\Delta F_t = \frac{f_t}{N} = \frac{1}{\Delta f_t N} = \frac{1}{T}$

ΔF_t: Periode in der Frequenzdomaine

f_t: Sampling-Frequenz in der Zeitdomäne

N: Anzahl der Werte in der Zeitdomäne

Δf_t: Periode in der Zeitdomäne

T: Dauer der Zeitreihe in der Zeitdomäne

Wichtig: Die Frequenzauflösung ist nur von der Zeitdauer in der Zeitdomäne abhängig und nicht von der Abtastfrequenz in der Zeitdomäne

Beispiel

Im Folgenden wird der Einfluss der Länge einer Zeitreihe auf die Frequenzauflösung illustriert:

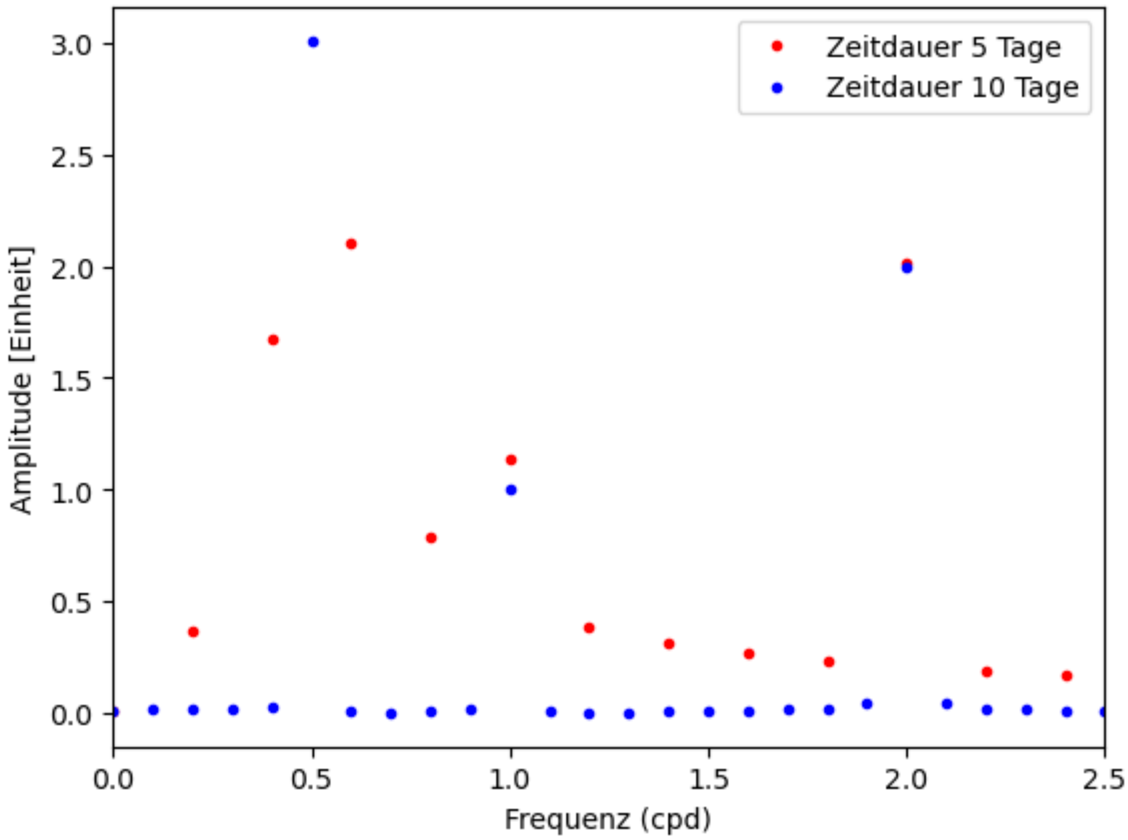
In [2]:
```
time = np.linspace(0, 5, 5*96, endpoint=True)
freq1 = 1
amp1 = 1
comp1 = amp1 * np.cos(time*2*np.pi*freq1)
freq2 = 0.5
amp2 = 3
comp2 = amp2 * np.cos(time*2*np.pi*freq2)
freq3 = 2
amp3 = 2
comp3 = amp3 * np.cos(time*2*np.pi*freq3)
comp_sum1 = comp1 + comp2 + comp3
```

In [3]:
```
N = len(comp_sum1)
T = 96
data_fft1 = np.fft.fft(comp_sum1)[0:int(N/2)]
freqs1 = np.fft.fftfreq(N, d=1/T)[0:int(N/2)]
amplitude1 = (2/N)*np.abs(data_fft1)
```

In [4]:
```
time = np.linspace(0, 10, 10*96, endpoint=True)
freq1 = 1
amp1 = 1
comp1 = amp1 * np.cos(time*2*np.pi*freq1)
freq2 = 0.5
amp2 = 3
comp2 = amp2 * np.cos(time*2*np.pi*freq2)
freq3 = 2
amp3 = 2
comp3 = amp3 * np.cos(time*2*np.pi*freq3)
comp_sum2 = comp1 + comp2 + comp3
```

In [5]:
```
N = len(comp_sum2)
T = 96
data_fft2 = np.fft.fft(comp_sum2)[0:int(N/2)]
freqs2 = np.fft.fftfreq(N, d=1/T)[0:int(N/2)]
amplitude2 = (2/N)*np.abs(data_fft2)
```

In [6]:
```
plt.plot(freqs1, amplitude1, 'r.', label='Zeitdauer 5 Tage')
plt.plot(freqs2, amplitude2, 'b.', label='Zeitdauer 10 Tage')

plt.xlabel('Frequenz (cpd)')
plt.ylabel('Amplitude [Einheit]')
plt.xlim(0,2.5)
plt.legend()

plt.show()
```



FAZIT: Je länger die Zeitreihe, desto besser die Auflösung von einzelnen Frequenzkomponenten!

Maximale Frequenzgrenze

Das obere Frequenzlimit der FFT bestimmt die Obergrenze von detektierbaren Frequenzen

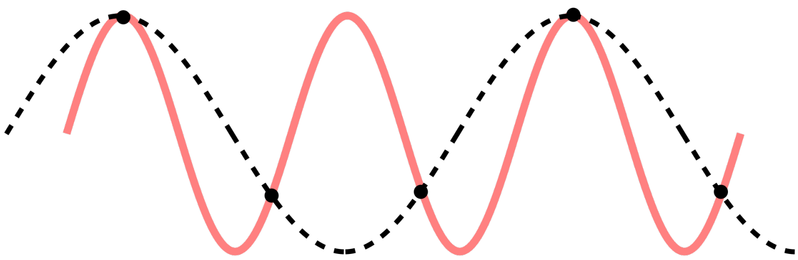
Die Nyquist-Frequenz

Die Obergrenze von detektierbaren Frequenzen ist die sogenannte Nyquist-Frequenz:

$$F_{Nyquist} = \frac{f_t}{2}$$

f_t: Sampling-Frequenz in der Zeitdomäne

Quelle: Wikipedia



Beispiel: Siehe Passing the Nyquist Limit by Jack Schaedler

FAZIT: Für die Bestimmung von Frequenzen muss die Abtastrate in der Zeitdomäne beachtet werden!

Der Leck-Effekt

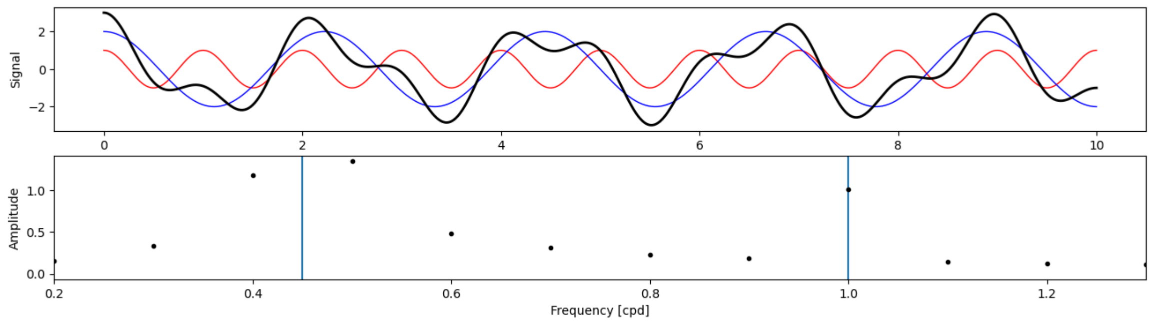
- Die DFT nimmt an, das Frequenzkomponenten zeitlich kontinuierlich (d.h., unendlich lang) sind
- Das bedeutet, dass Frequenzen, welche nicht genau in die Zeitdauer passen eine Diskontinuität hervorrufen
- Diese Diskontinuität führt zum sogenannten **Leck-Effekt**

Beispiel zum Leck-Effekt

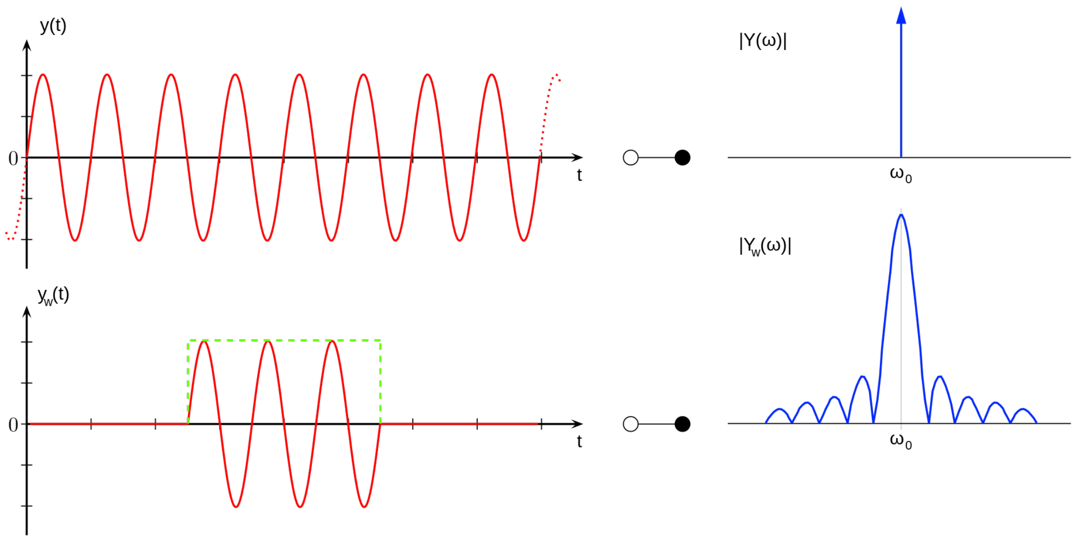
In [7]:
```
time = np.linspace(0, 10, 10*96, endpoint=True)
freq1 = 1
amp1 = 1
comp1 = amp1 * np.cos(time*2*np.pi*freq1)
freq2 = 0.45
amp2 = 2
comp2 = amp2 * np.cos(time*2*np.pi*freq2)
comp_sum = comp1 + comp2
```

In [8]:
```
N = len(comp_sum)
T = 96
data_fft = np.fft.fft(comp_sum)[0:int(N/2)]
freq = np.fft.fftfreq(N, d=1/T)[0:int(N/2)]
amplitude = (2/N)*np.abs(data_fft)
```

In [9]:
```
fig, ax = plt.subplots(2, 1, figsize=(16, 4))

ax[0].plot(time, comp1, 'r', lw=1, label='Component 1')
ax[0].plot(time, comp2, 'b', lw=1, label='Component 2')
ax[0].plot(time, comp_sum, 'k', lw=2, label='Component 2')
ax[0].set_xlabel('Time [days]')
ax[0].set_ylabel('Signal')

ax[1].axvline(0.45)
ax[1].axvline(1)
ax[1].plot(freq, amplitude, 'k.', label='Zeitdauer 10 Tage')
ax[1].set_xlim(0.2, 1.3)
ax[1].set_xlabel('Frequency [cpd]')
ax[1].set_ylabel('Amplitude')

plt.show()
```



Quelle: Wikipedia



Beispiel: Siehe The Phenomenon of Leakage

FAZIT: Der Leck-Effekt kann minimiert werden durch:

- eine Anpassung der Datenlänge in der Zeitdomäne,
- durch die Verwendung einer Fensterfunktion

Fensterfunktionen

- Die Berechnung der DFT für eine Zeitreihe entspricht immer einem Rechteckfenster
- Das Rechteckfenster ist genauso lang wie die Dauer der Zeitreihe
- Es können auch andere Fensterfunktionen verwendet werden
- Diese können auf bestimmte Kriterien ausgerichtet werden
- Zum Beispiel kann man Diskontinuitäten vermeiden

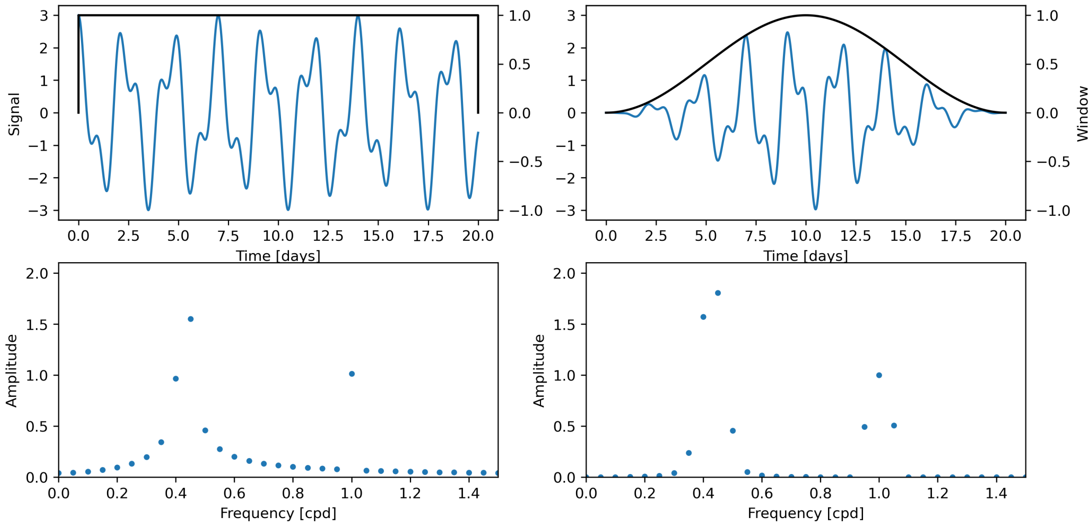
Beispiel: Vergleich von Rechteck und Hanning Fenster

Das **Von-Hann** (oder auch **Hanning**) Fenster ist sehr beliebt:

$$w(n) = \frac{1}{2}\left[1 + cos\left(\frac{2\pi n}{M-1}\right)\right]$$

$w(n)$ ist das Gewicht für den Wert n

M ist die Anzahl der Werte



FAZIT

Ein Fenster:

- ... kann den Leck-Effekt verringern, aber niemals ganz beheben
- ... beeinflusst das Signal, und das Spektrum muss deshalb kompensiert werden (Amplitudenfaktor für Hanning ~ 2)
- ... muss auf die individuellen Umstände agepasst werden

Hinweis: Es gibt sehr viele Fenster mit unterschiedlichen Eigenschaften, siehe auch Fensterfunktion auf Wikipedia

Anwendung des *Von-Hann* Fensters auf die Meeresspiegel Zeitreihe

In [10]:
```
sea_level = pd.read_csv('data/Palau_sea-level.csv', parse_dates=True, index_col=
sea_level
```

Out[10]:

	Sea level [mm]
Datetime[GMT]	
2017-11-25 02:00:00	2124
2017-11-25 03:00:00	2127
2017-11-25 04:00:00	1997
2017-11-25 05:00:00	1886
2017-11-25 06:00:00	1734
...	...
2018-12-31 19:00:00	1566
2018-12-31 20:00:00	1496
2018-12-31 21:00:00	1372
2018-12-31 22:00:00	1218
2018-12-31 23:00:00	1062

9646 rows × 1 columns

In [11]:

```
# the rectangular window
N = len(sea_level)
T = 24

data_fft = np.fft.fft(sea_level['Sea level [mm]'])
data_fft = data_fft[0:int(N/2)]

freq = np.fft.fftfreq(N, d=1/T)
freq = freq[0:int(N/2)]

amp_rect = (2/N)*np.abs(data_fft)
```

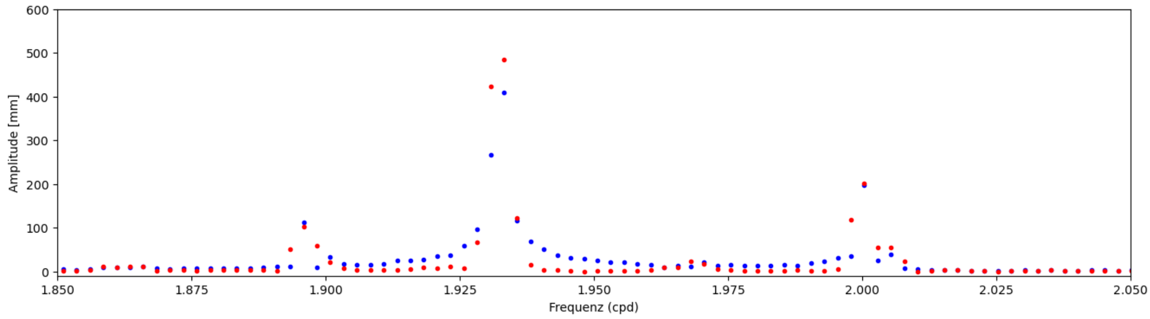
In [12]:

```
# the Hanning window
N = len(sea_level)
T = 24

# generate the window
hanning_data = np.hanning(N)*sea_level['Sea level [mm]']

data_fft = np.fft.fft(hanning_data)
data_fft = data_fft[0:int(N/2)]

freq = np.fft.fftfreq(N, d=1/T)
freq = freq[0:int(N/2)]

amp_hann = 2*(2/N)*np.abs(data_fft)
```

In [13]:

```
fig, ax = plt.subplots(figsize=(16,4))
ax.plot(freq[1:], amp_rect[1:], 'b.', label='Rectangular window')
ax.plot(freq[1:], amp_hann[1:], 'r.', label='Hanning window')
ax.set_xlabel('Frequenz (cpd)')
ax.set_ylabel('Amplitude [mm]')
ax.set_xlim(1.85, 2.05)
ax.set_ylim(-10, 600)
```

Out[13]: (-10.0, 600.0)



Weitere Methoden der Frequenanalyse

- Frequenzanalyse ist ein eigenes Feld der Signalverarbeitung
- Dies erfordert ein eigenes Lehrmodul
- DFT ist sehr gut etabliert und beliebt
- Die DFT ist aber nur eine von vielen Möglichkeiten zur Frequenzanalyse von Zeitreihen
- Neuere Methoden sind mathematisch komplizierter

ENDE