

Fourier-Analyse Pegeldaten

June 14, 2021

Analysis II

1 Analyse von Pegeldaten mittels FFT

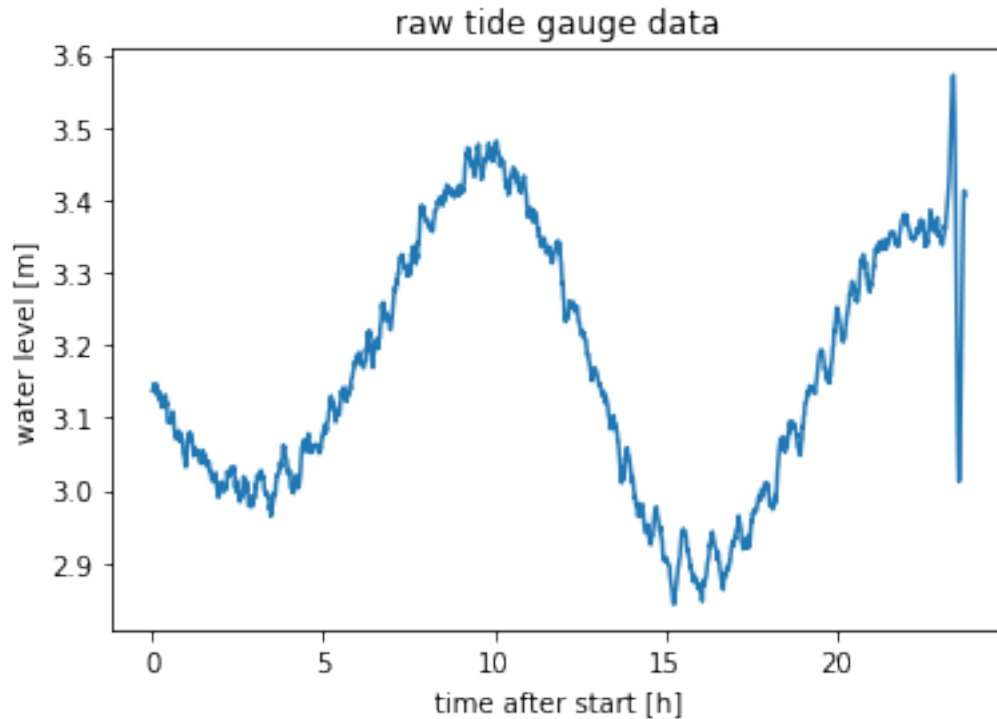
Jörn Behrens (joern.behrens@uni-hamburg.de)

In diesem Beispiel soll die Analyse realer Pegeldaten demonstriert werden. Dazu wenden wir die schnelle Fourier-Transformation auf eine Zeitreihe von Wasserstandsmessungen an, die vom Küstenpegel im Hafen von Padang, West-Sumatra, Indonesien für einen Zeitraum vom 29. Sept. 2009 23:00 Uhr (UTZ) bis 30. Sept. 2009 06:00 Uhr aufgezeichnet wurden. Die Zeiten sind minütlich gegeben, wir skalieren sie auf eine Zeit nach Beginn der Messungen, d.h. unsere Minute 1 beginnt mit der ersten Messung. Außerdem skalieren wir die Werte auf Stunden.

```
[1]: from numpy import loadtxt, array
    gauge=loadtxt('Gauge_data_pagang.txt', usecols=(1,))
    hours=array(range(1,len(gauge)+1))/60.
```

Um einen Überblick zu erhalten stellen wir die Daten graphisch dar.

```
[6]: import matplotlib.pyplot as plt
    %matplotlib inline
    plt.plot(hours,gauge)
    plt.title('raw tide gauge data')
    plt.xlabel('time after start [h]')
    plt.ylabel('water level [m]');
```



Als Menschen sehen wir sofort, dass ca. 23 Stunden nach Beginn der Messung etwas ungewöhnliches passiert ist. Das wollen wir aber mittels Fourier-Transformationen besser (quantitativer) analysieren.

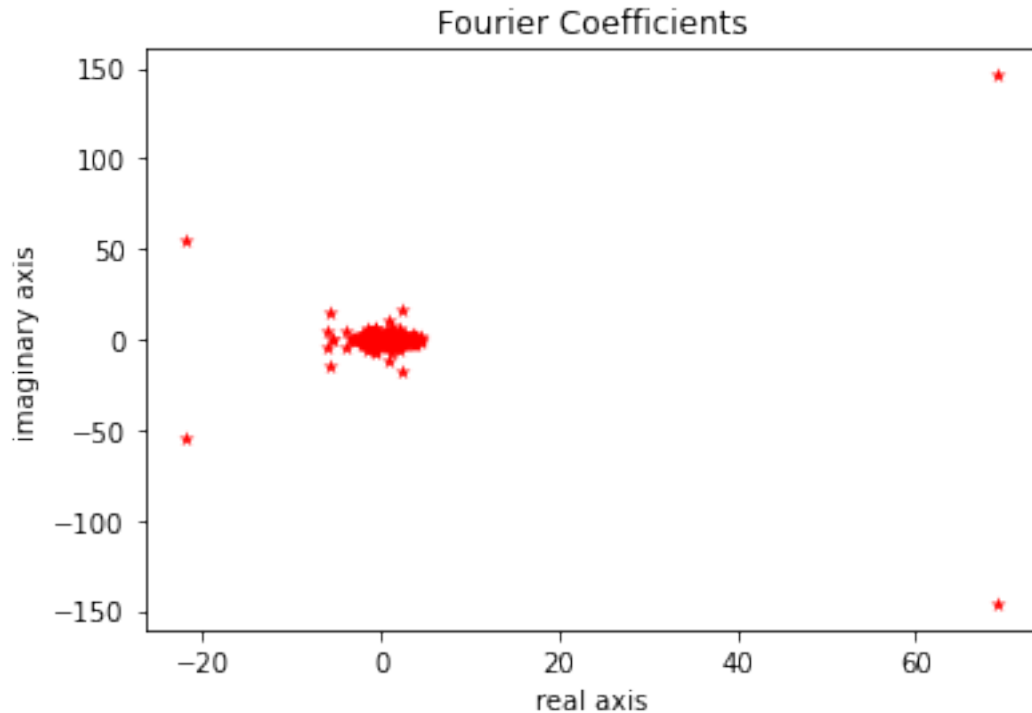
```
[7]: from numpy.fft import fft
      Y=fft(gauge)
      n=len(Y)
```

Der erste Koeffizient in Y enthält den Mittelwert der gesamten Zeitreihe (Parsevallsche Gleichung), daher vernachlässigen wir ihn in der Analyse.

```
[8]: from numpy import delete
      ysav=Y[0]
      Y=delete(Y,0)
```

Wir stellen die Koeffizienten graphisch dar. Beachten Sie, dass sie komplex sind!

```
[9]: plt.scatter(Y.real,Y.imag,marker='*', c='r', linewidths=0)
      plt.title('Fourier Coefficients')
      plt.xlabel('real axis')
      plt.ylabel('imaginary axis');
```



Da die Koeffizienten jeweils auch die komplex Konjugierten umfassen (gespiegelt an der reellen Achse), können wir auf eine Hälfte der Koeffizienten für die Analyse verzichten.

```
[10]: half=int(n/2)
```

Das Powerspektrum besteht aus den quadrierten Beträgen der Koeffizienten. Und diese Werte beschreiben die *Energie* jeder zum jeweiligen Koeffizienten gehörenden Wellenlänge (Frequenz) im Signal.

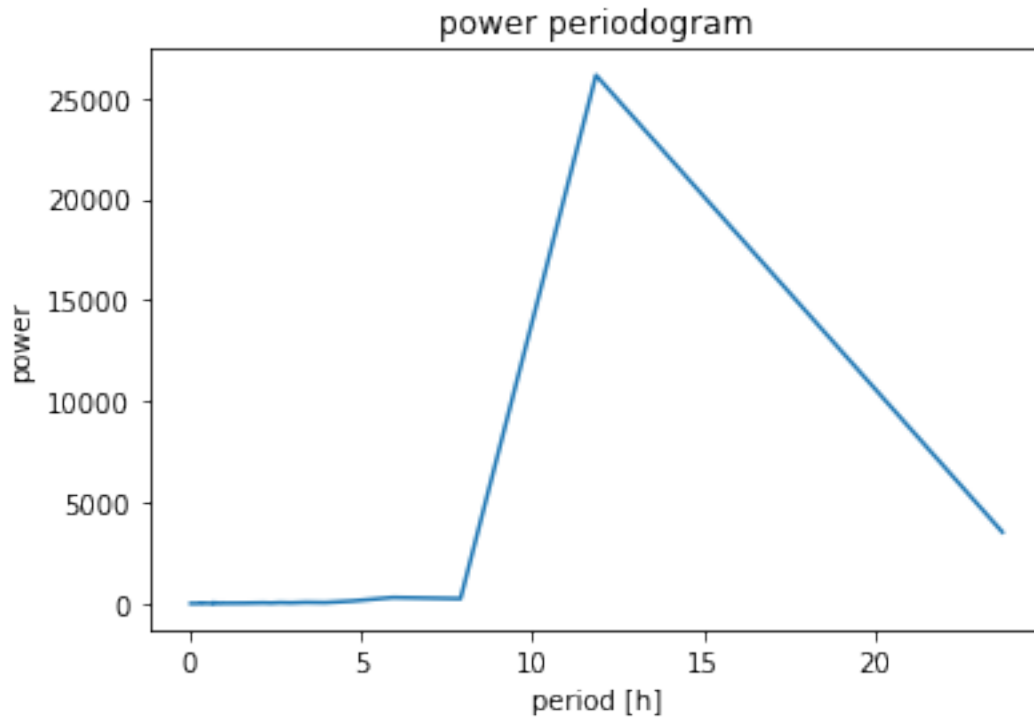
```
[11]: power= abs(Y[0:half])**2
```

Wir sind häufig eher an der Periode als an der Frequenz interessiert, daher berechnen wir sie, indem wir den Kehrwert der Frequenz ermitteln. Außerdem skalieren wir die Periode so, dass sie in Stunden angegeben ist.

```
[12]: freq= array(range(1,int(half+1))).astype(float)/float(n)
      peri= (1./freq)
      perih=peri/60.
```

Das Ergebnis ist das *Power-Periodogramm*, das wir in der folgenden Graphik sehen.

```
[13]: plt.plot(perih,power)
      plt.title('power periodogram')
      plt.xlabel('period [h]')
      plt.ylabel('power');
```



Wir erkennen ein klares sehr ausgeprägtes Maximum bei ca. 11 Stunden Wellenperiode. Das wollen wir aber genauer bestimmen:

```
[14]: from numpy import where, max
ind= where(power == max(power))[0][0]
period= peri[ind]/60.
print('Dominante Periode in Stunden: %0.5g' % (period))
```

Dominante Periode in Stunden: 11.867

Das entspricht in etwa der 12-Stunden Periode des Tidensignals. Weitere signifikante Perioden lassen sich ähnlich aus dem Signal extrahieren. Dazu setzen wir jeweils den gefundenen Koeffizienten zu Null und suchen das nächste Maximum.

```
[15]: pwanalys=power; pwanalys[ind]=0;
ind2=where(pwanalys == max(pwanalys))[0][0]
period2= peri[ind2]/60.
print('2. Dominante Periode in Stunden: %0.5g\n' % (period2))

pwanalys[ind2]=0
ind3=where(pwanalys == max(pwanalys))[0][0]
period3= peri[ind3]/60.
print('3. Dominante Periode in Stunden: %0.5g\n' % (period3))

pwanalys[ind3]=0
```

```
ind4=where(pwanalysis == max(pwanalysis))[0][0]
period4= peri[ind4]/60.
print('4. Dominante Periode in Stunden: %0.5g\n' % (period4))
```

2. Dominante Periode in Stunden: 23.733

3. Dominante Periode in Stunden: 5.9333

4. Dominante Periode in Stunden: 7.9111

Die so gefundenen dominanten Perioden korrespondieren in etwa mit den harmonischen Komponenten des Tidensignals, die unter den Bezeichnungen M2, K1, M4, M3 bekannt sind.

Jetzt wollen wir ein glattes Tidensignal ohne die kleinskaligen Fluktuationen rekonstruieren, dass in etwa einem idealisierten Tidenverlauf entspricht. Wir können das als *sauberes* Tidensignal interpretieren, und alle Abweichungen davon als *Störungen*.

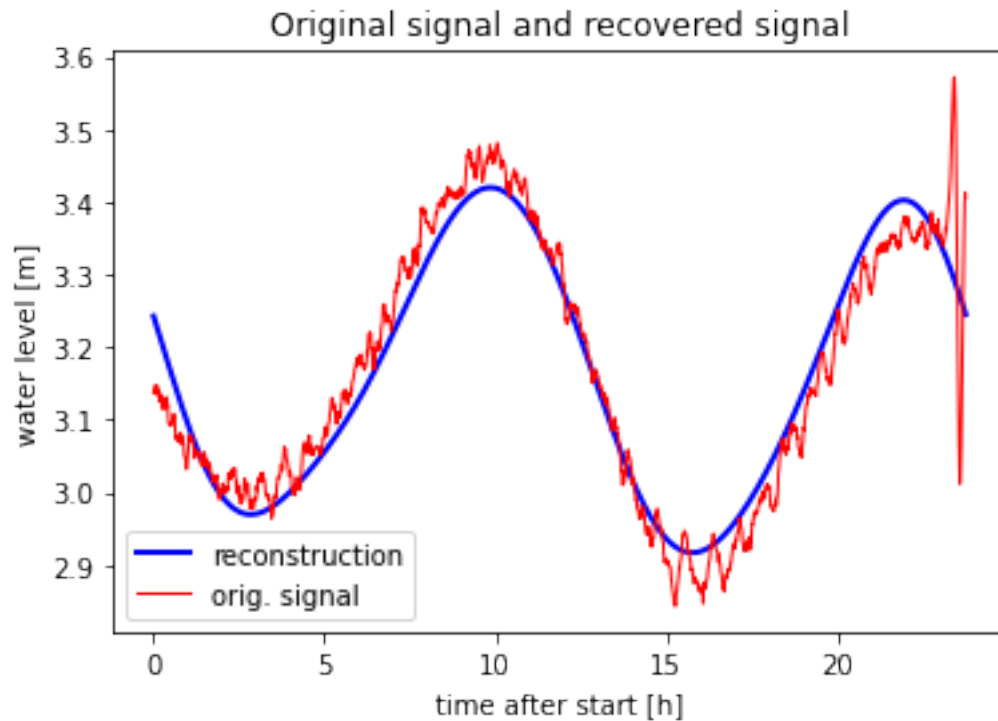
```
[16]: from numpy import zeros, concatenate
Yrecov=zeros(len(Y), dtype=complex)
Yrecov[ind]= Y[ind]; Yrecov[len(Y)-ind-2]= Y[len(Y)-ind-2]
Yrecov[ind2]= Y[ind2]; Yrecov[len(Y)-ind2-2]= Y[len(Y)-ind2-2]
Yrecov[ind3]= Y[ind3]; Yrecov[len(Y)-ind3-2]= Y[len(Y)-ind3-2]
Yrecov[ind4]= Y[ind4]; Yrecov[len(Y)-ind4-2]= Y[len(Y)-ind4-2]
Yrecov= concatenate([ysav], Yrecov)
```

Mit der inversen schnellen Fourier Transformation können wir und das Wellesignal aus den (wenigen) Koeffizienten rekonstruieren.

```
[17]: from numpy.fft import ifft
x=ifft(Yrecov)
```

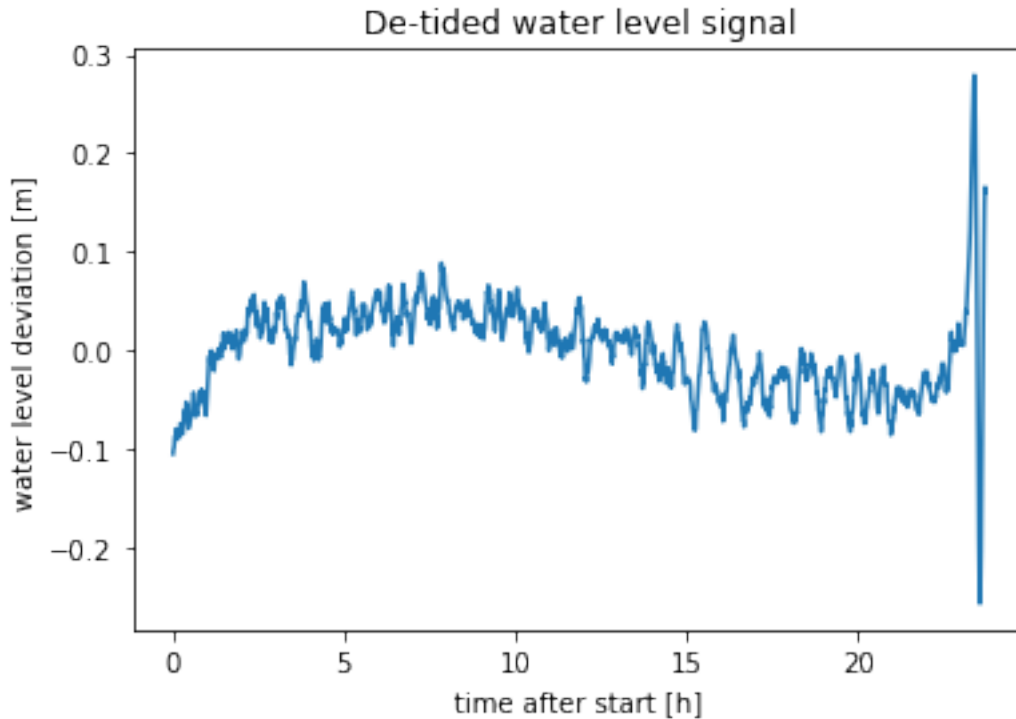
Wir stellen den reellen Anteil des rekonstruierten Signals zusammen mit dem original Signal graphisch dar.

```
[18]: from numpy import real
gaugerecov=real(x)
h2=plt.plot(hours,gaugerecov,linewidth=2, c=[0,0,1], label='reconstruction')
h3=plt.plot(hours,gauge,linewidth=1, c=[1,0,0], label='orig. signal')
plt.legend(loc='lower left')
plt.title('Original signal and recovered signal')
plt.xlabel('time after start [h]')
plt.ylabel('water level [m]');
```



Schließlich können wir diese geglättete Rekonstruktion verwenden, um das Signal vom Tidenverlauf zu befreien und lediglich die Störungen zu erhalten. Die Differenz der beiden Kurven stellt letztlich die Abweichung vom erwarteten Tidenverlauf dar. Diese Differenz könnte beispielsweise für eine automatische Erkennung von Naturereignissen (Tsunami, Sturmflut) verwendet werden.

```
[19]: gaugedetide= gauge-gaugerecov
plt.figure(5)
plt.plot(hours,gaugedetide)
plt.title('De-tided water level signal')
plt.xlabel('time after start [h]')
plt.ylabel('water level deviation [m]');
```



Schlussbemerkung: Die Anomalie im Wellenverlauf, die wir schon mit bloßem Auge im ursprünglichen Signal erkennen konnten hat eine Amplitude von etwa 0,5 m (wie wir der obigen Graphik entnehmen können). Diese Welle war ein “kleiner” Tsunami, der durch ein Erdbeben der Magnitude 7.6 in der Nähe der Stadt Padang ausgelöst wurde (siehe <http://geofon.gfz-potsdam.de/eqinfo/event.php?id=gfz2009tdkv>). Sie finden eine Simulation des Tsunamis in der folgenden Quelle: <http://hdl.handle.net/10013/epic.34035.d001>.

[]: