

Fourier-Analyse Sonnenflecken

June 14, 2021

Analysis II (Sommer 2017)

1 Datenanalyse der Sonnenflecken mit FFT

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

Dieses Beispiel demonstriert die Analyse realer Daten einer Zeitreihe. Die Datei *sunspots.dat* enthält für jedes Jahr zwischen 1700 und 2014 die Wolf-Zahl.

1.1 (a) Daten einlesen und darstellen:

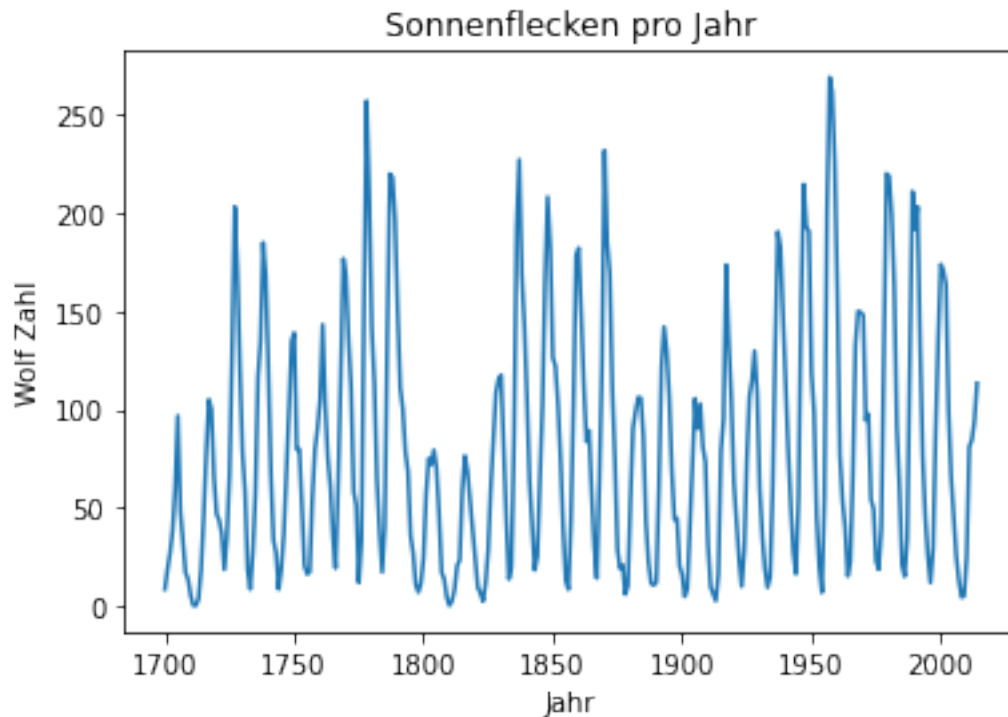
Zunächst laden wir die Daten mit dem `loadtxt` Befehl und weisen sie zwei Variablen (`year` und `wolf`) zu:

```
[1]: from numpy import loadtxt
raw=loadtxt('sunspots.txt')
year= raw[:,0]
wolf= raw[:,1]
```

Jetzt plotten wir die Daten, um einen ersten Eindruck zu erhalten.

```
[2]: import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(year,wolf)
plt.title('Sonnenflecken pro Jahr')
plt.xlabel('Jahr')
plt.ylabel('Wolf Zahl')
```

```
[2]: Text(0, 0.5, 'Wolf Zahl')
```



1.2 (b) Analyse

Um die vorherrschende Periode in den Daten zu ermitteln werden wir dieses Signal nun mit Hilfe der Fourier-Analyse analysieren. Dazu führen wir zuerst eine Transformation mit FFT durch.

```
[3]: from numpy.fft import fft
      Y=fft(wolf)
      n=len(Y)
```

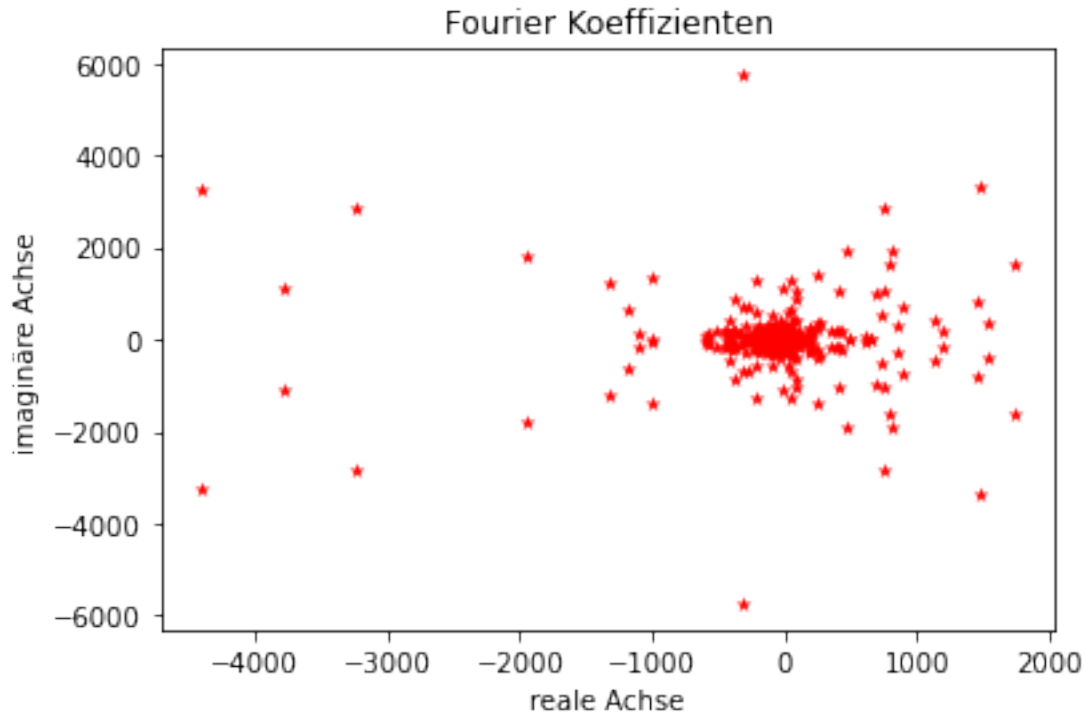
Wir können den ersten Eintrag in Y verwerfen, da es sich um den Koeffizienten zur konstanten Funktion handelt.

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

Die restlichen Koeffizienten plotten wir. Beachten Sie, dass die Koeffizienten komplex sind!

```
[5]: plt.scatter(Y.real,Y.imag,marker='*', c='r', linewidths=0)
      plt.title('Fourier Koeffizienten')
      plt.xlabel('reale Achse')
      plt.ylabel('imaginäre Achse')
```

```
[5]: Text(0, 0.5, 'imaginäre Achse')
```



Wir sehen, dass die Koeffizienten natürlich an der realen Achse gespiegelt sind, denn wir wissen ja, dass $\alpha_{-n} = \overline{\alpha_n}$. Also brauchen wir für die Bestimmung der dominanten Wellen nur noch eine Hälfte der Koeffizienten zu betrachten.

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

Das sogenannte Powerspektrum besteht aus den Beträgen der quadrierten Koeffizienten. Es beschreibt die *Energie* in jeder einzelnen Wellenlänge der Fourier-Reihe.

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

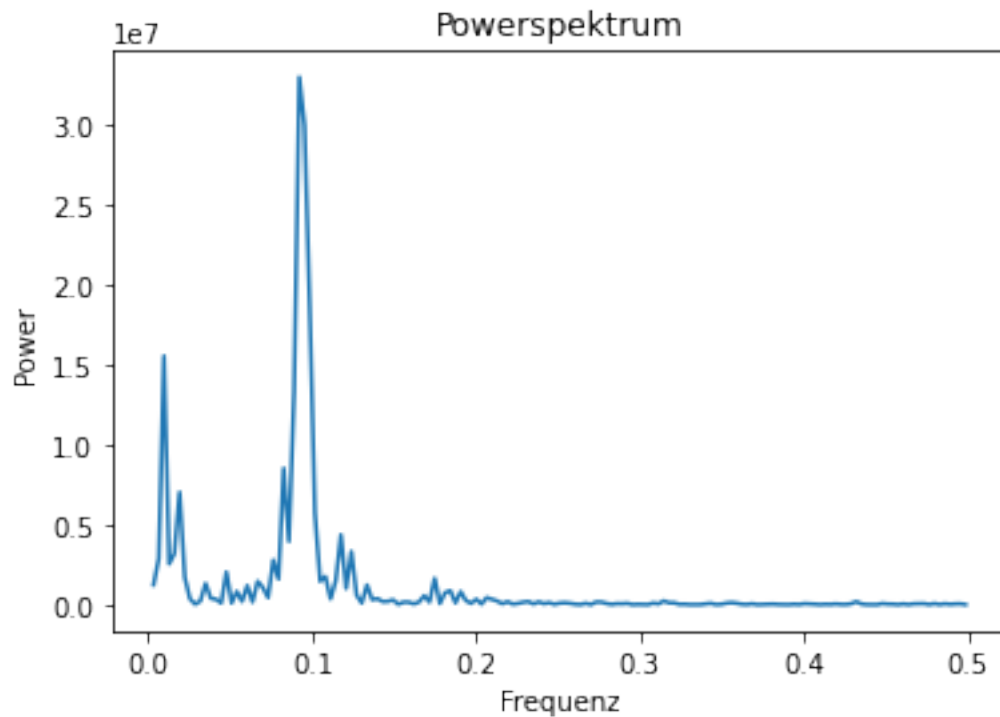
Die zugehörige Frequenz ist gerade die Wellenzahl, während die Periode das Inverse der Frequenz ist.

```
[8]: from numpy import array
freq= array(range(1,int(half+1))).astype(float)/float(n)
peri= (1./freq)
```

Wir plotten das Powerspektrum bezüglich der Frequenzen.

```
[9]: plt.plot(freq,power)
plt.title('Powerspektrum')
plt.xlabel('Frequenz')
plt.ylabel('Power')
```

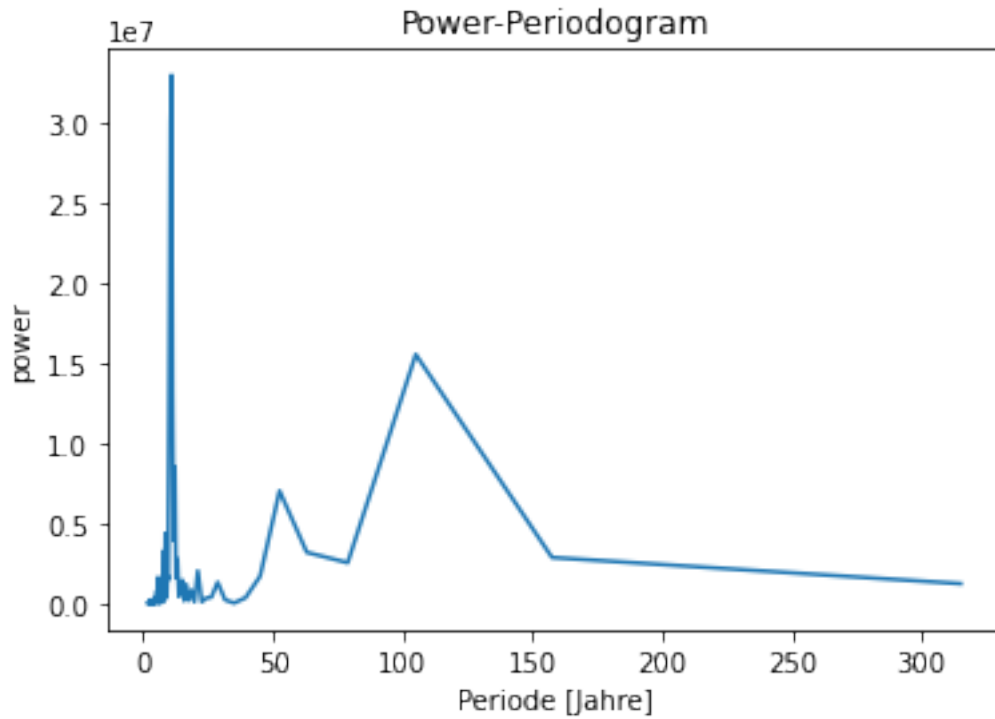
```
[9]: Text(0, 0.5, 'Power')
```



Bei der Frequenz von ca. 0.1 gibt es ein klares und auffällige Maximum. Allerdings ist die Frequenz in diesem Fall nicht sehr aussagekräftig, denn wir wollen ja die dominante Periode kennen lernen. Also stellen wir das Power-Periodogramm dar:

```
[10]: plt.plot(peri,power)
plt.title('Power-Periodogram')
plt.xlabel('Periode [Jahre]')
plt.ylabel('power')
```

```
[10]: Text(0, 0.5, 'power')
```

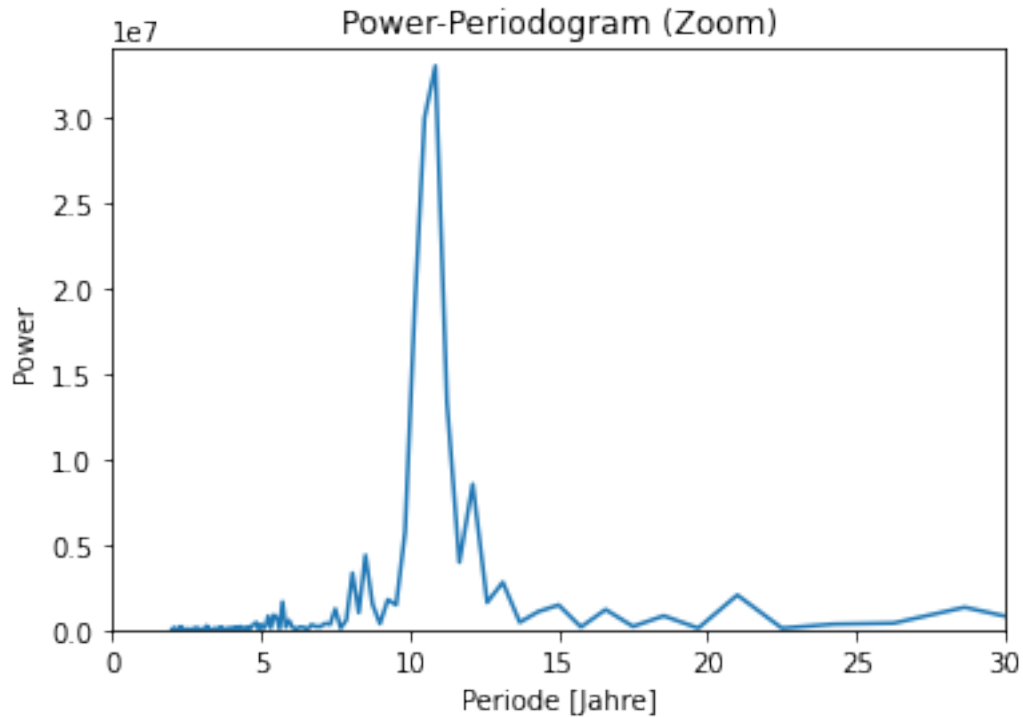


1.3 (c) Achsenskalierung

Die größte Spitze ist in dieser Darstellung schwer auszumachen, daher zoomen wir etwas in den Bereich zwischen 0 und 30 Jahren:

```
[11]: plt.plot(peri,power)
plt.axis([0, 30, 0, 3.4e+7])
plt.title('Power-Periodogram (Zoom)')
plt.xlabel('Periode [Jahre]')
plt.ylabel('Power')
```

```
[11]: Text(0, 0.5, 'Power')
```



Nun ermitteln wir noch den exakten Wert der maximalen Periode:

```
[12]: from numpy import where, max
ind= where(power == max(power))[0][0]
period= peri[ind]
print('Dominante Periode in Jahren: %0.5g' % (period))
```

Dominante Periode in Jahren: 10.862

1.4 (d) Fehlerbetrachtung

Um den Fehler der Berechnung zu bestimmen können wir die Original-Daten mit den einmal hin und zurück transformierten Daten vergleichen.

```
[13]: from numpy.fft import fft,ifft
from numpy.linalg import norm
x=wolf
xhat= ifft(fft(wolf))
err= norm(xhat-x)/norm(x)
print('Der relative vorwärts Fehler der FFT: %12.8e' % (err))
```

Der relative vorwärts Fehler der FFT: 2.64627543e-16

```
[ ]:
```