Numerische Methoden der Physik Serie 5

Cedric Sigrist

30. Mai 2025

1 Filterung

Für die Aufgaben wurde der Datensatz `GRCAT07001.ACC` verwendet. Dabei wurden nur die linearen Beschleunigungen in die S-Richtung welche in Einheiten von $\mathrm{mm/s^2}$ gegeben waren betrachtet.

1.1 Urspringliche und Gefilterte Messungen

In Abbildung 1.1 sieht man die Gefilterten und Ungefilterten Daten über den ganzen Zeitraum. Es lässt sich noch nicht sehr viel Erkennen.

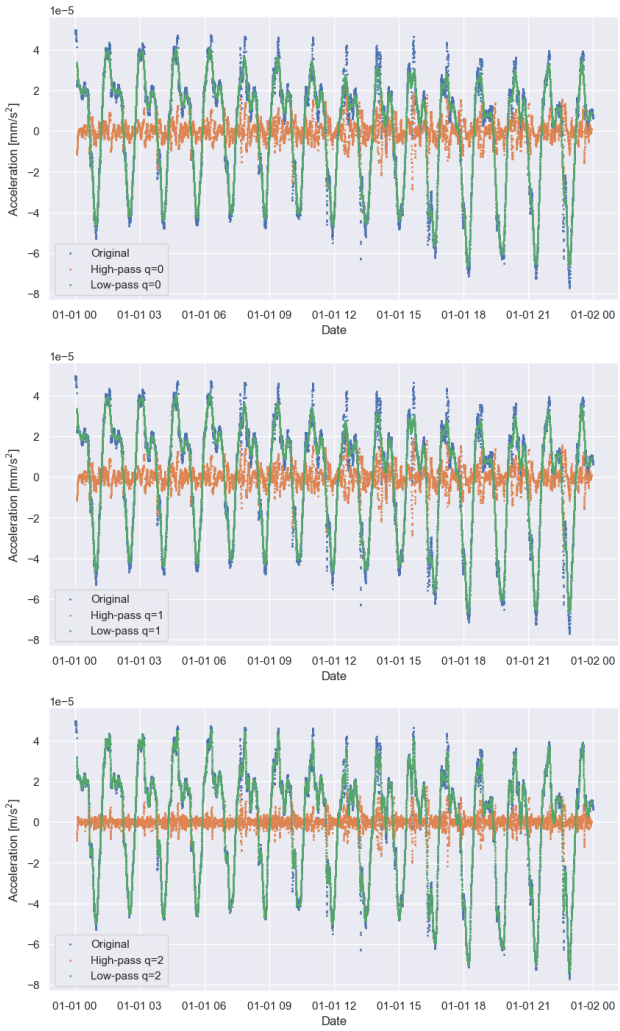

Abbildung 1.1: Die Gefilterten und Ungefilterten Daten für $q \in \{0, 1, 2\}$

1.2 Ursprüngliche und Gefilterte Messungen im Zeitraum von 1 h

Betrachtet man nun nur einen Zeitabschnitt von 1 h, wie in Abbildung 1.2 zu sehen ist, kann man die Unterschiede der verschiedenen Grade des Polynoms erkennen. Da hier die erste Stunde des Datensatzes verwendet wurde, kann man hier auch sehen wie ich entschieden habe die Ränder bei dem Savitzky-Golay-Filter zu behandeln. Meine Idee war folgende: Normalerweise wird bei dem Filter, sofern die Fensterbreite ungerade ist und die Datenpunkte äquidistant, das Polynom immer 'in der Mitte' ausgewertet. Deshalb ist es möglich den Filter als Faltung darzustellen. Man muss das Polynom aber nicht zwingend in der Mitte des Fensters auswerten! Daher verwende ich einfach die Koeffizienten der Polynome welche zu dem ersten und letzten Fenster gehören, und werte dann diese an den Punkten aus, welche sonst nicht abgedeckt wären. In Abbildung 1.2 kann man ausserdem sehen, dass die Gefilterten Daten für $q = 0$ und $q = 1$ grösstenteils gleich aussehen. Das hat damit zu tun, dass polynomiale fits vom Grad 0 und Grad 1 ausgewerten am Zentrum den selben Wert geben. Aufgrund der von mir gewählten Art die Ränder zu behandeln, kann man nun aber die Methoden unterscheiden, da die Polynome 'off-center' ausgewertet werden. Man sieht das die Interpolation am Rand aussehen wie ein Polynom vom Grad q.

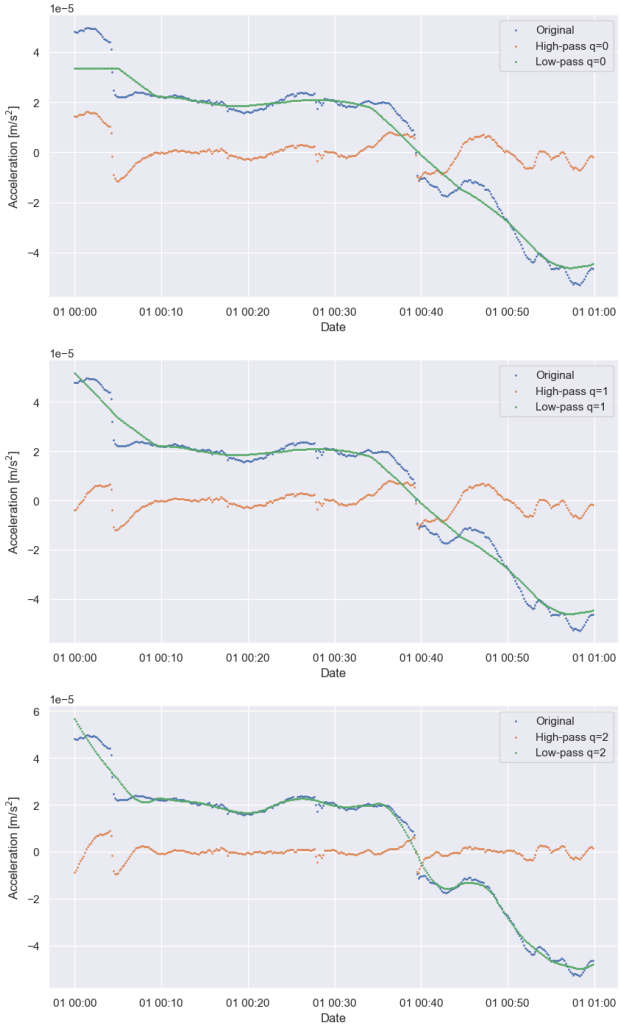

Abbildung 1.2: Gefilterte Daten der ersten Stunde der Messung

1.3 Amplitude & Leistungsspektrum

Mit der numpy funktion `fft` lässt sich nun ein Amplituden und Leistungsspektrum berechenen. Hier ist zu beachten, dass ich die Amplituden und Leistungen mit der Zeitdauer des Datensatzes nomralisiert habe, so dass sie Einheit der Amplituden wieder mm/s^2 ist, so wie es bei einer DFT wäre. Um Platz zu sparen werden hier nur die Amplitudenspektren dargestellt, da die Leistungen keine Zusätzliche Information zeigen. Die Leistungsspektren sind in dem Jupyter-Notebook zu sehen. In den Amplitundenspektren in Abbildung 1.3 lässt sich erkennen, das der beim Tiefpass nur Amplituden mit Langen Perioden enthalten sind und beim Hochpass nur kurze, dies ist zu erwarten.

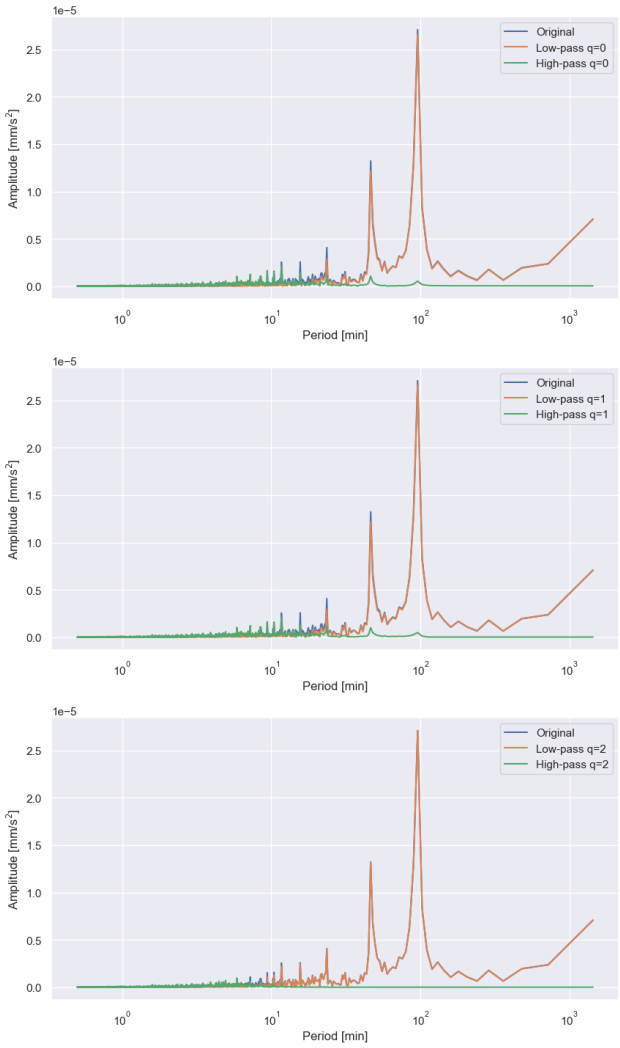

Abbildung 1.3: Amplitudenspektrum

2 Behandlung der Randterme

Um die Zusammenfassung kurz zu halten habe ich direkt von Anfang an die von mir gewählte Behandlung der Randterme verwendet, bei der die Polynome des ersten und letzten Fensters nicht in der Mitte ausgewertet werden.

3 Diskrete Fourier Transformation

Ziel ist es, die Daten in folgender Form darzustellen.

$$a(t) = a_0 + \sum_{i=1}^{m} \left(a_i \cos(i\omega t) + b_i \sin(i\omega t)\right) \qquad \text{mit} \quad \omega = \frac{2\pi}{|t_N - t_1|}$$

Dazu kann mein einfach die Designmatrix Aufstellen und die Normalengleichung lösen. Das ist sehr einfach, da das Modell $a(t; a_0, \ldots, a_m, b_1, \ldots, b_m)$ nur linear von den Parametern $a_0, \ldots, a_m, b_1, \ldots, b_m$ abhängig ist. Man kann ausserdem `numpy.fft` verwenden. Wenn man dann aber die Koeffizienten oder das Amplitudenspektrum herausfinden möchte, muss man aufpassen dass alles korrekt normalisiert ist. Bei korrekter Durchführung sind die Beiden Methoden identisch wie in Abbildung 3.1 zu sehen ist. Es lässt sich ausserdem derjenige Koeffizient mit maximaler Amplitude $A_i^2 = a_i^2 + b_i^2$ bestimmen. Dieser ist für alle m derselbe, nämlich der 15. Koeffizient mit:

$$\begin{aligned} a_{15} &= 2.22 \times 10^{-5}\,\mathrm{mm/s^2} \\ b_{15} &= -1.56 \times 10^{-5}\,\mathrm{mm/s^2} \\ A_{15} &= 2.71 \times 10^{-5}\,\mathrm{mm/s^2} \\ T_{15} &= 96\,\mathrm{min} \end{aligned}$$

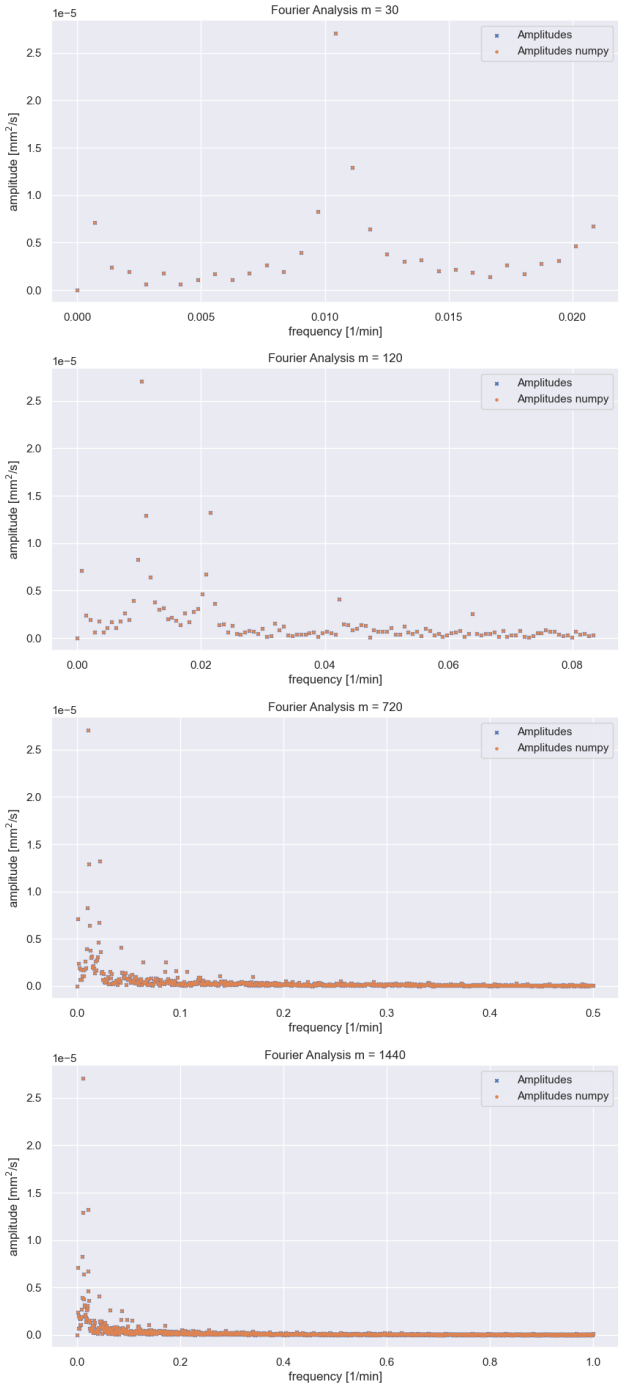

Abbildung 3.1: Diskrete Fourier Transformation mit `numpy` und Designmatrix