Universität Passau • Fakultät für Informatik und Mathematik

Superresolution durch Kettenbrüche

Bachelorarbeit

im Studiengang Bachelor Mathematik

vorgelegt von

Markus GEORGI

Tag der Einreichung: 23. September 2020

Prof. Dr. Tomas Sauer (Lehrstuhl für Mathematik mit Schwerpunkt Digitale Bildverarbeitung)

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Theorie

1.1 Einführung

Die Superresolution beschreibt die Vergrößerung der Auflösung von Bildern, womit der Informationsgehalt für den Betrachter erhöht wird. Dafür gibt es bereits einige etablierte Methoden. Ziel der Arbeit ist es allerdings eine Alternativlösung für das Superresolution-Problem vorzustellen, welche in keinerlei Wettbewerb zu den klassischen Methoden stehen soll. In dieser Arbeit wird das Bild bereits abstahiert als Signal vorliegen (dazu später mehr), von dem bestimmte Spitzenwerte ermittelt werden sollen. Man stelle sich z.B. das Bild von Sternen im Nachthimmel als graue Flecken auf schwarzem Grund vor. Das Ziel besteht darin, die hellsten Stellen, also die Zentren der Flecken und damit die genauen Standpunkte der Sterne zu ermitteln. Als Ausgabe werden wir eben jene Werte erhalten, an denen diese Spitzenwerte im Signal vorliegen. Das Problem soll auf numerische Art, ausschließlich mithilfe von Kettenbrüchen gelöst werden. Dabei besteht die Hoffnung, dass der Prozess möglichst stabil ist. Um das Problem zu lösen muss allerdings erst etwas Vorarbeit geleistet werden, um zu verstehen, wie die besagte Methode funktioniert. Dazu wird in den ersten Kapiteln die nötige Theorie behandelt. Außerdem wird die Vorgehensweise, die das Problem lösen soll (ursprünglich erdacht vom Mathematiker Gaspard de Prony), zunächst allgemein behandelt. Danach wird anhand der Theorie eine algorithmische Lösung des Superresolution-Problems erarbeitet, welche im Anschluss in Octave programmiert wird. Zum Schluss wird der Algorithmus mithilfe von ausgewählten Versuchen auf Stabilität untersucht und bewertet. Die Beweise der in dieser Arbeit benötigten Sätze werden zwar ausgelassen, da sie wenig zum Ziel der Arbeit beitragen, können allerdings in [Sau20] nachgelesen werden.

In diesem ersten Kapitel werden zunächst Kettenbrüche eingeführt und anschließend das Grenzwertverhalten unendlicher Kettenbrüche betrachtet, da wir über diesen Ansatz später unser Eingangssignal verarbeiten werden, was allerdings erst in Kapitel 1.2 näher beleuchtet wird.

Beginnen wir, indem wir zunächst eine etwas kompaktere Schreibweise für Kettenbrüche einführen.

1 Definition (zugehöriger Kettenbruch)**.**
Für $n \in \mathbb{N}$ und $a_0, ..., a_n \in \mathbb{Z}$ sei der **zugehörige Kettenbruch** die rationale Zahl

$$[a_0; a_1, ..., a_n] = a_0 + \cfrac{1}{a_1 + \cfrac{1}{\ddots + \cfrac{1}{a_{n-1} + \cfrac{1}{a_n}}}}. \tag{1.1}$$

Hiermit gibt es noch einige Probleme. Betrachte z.B. die Fälle $a_n = 0$, oder $a_{n+1} = -\frac{1}{a_n}$. In beiden Fällen würden wir durch 0 teilen. Um das zu vermeiden wählen wir zusätzlich $a_0 \in \mathbb{Z}$ und $a_j \in \mathbb{N}$ für $j \in \mathbb{N}$. Da die Pünktchen-Notation in (1.1) noch etwas vage ist, betrachten wir zusätzlich die rekursive Definition, bei der der Nenner des „großen“ Bruches selbst wieder einen Kettenbruch darstellt.

$$[a_0; a_1] = a_0 + \frac{1}{a_1}, \qquad [a_0; a_1, ..., a_n] = a_0 + \frac{1}{[a_1; a_2, ..., a_n]}$$

2 Beispiel.

$$\begin{aligned}\frac{26}{7} &= 3 + \frac{5}{7} = 3 + \frac{1}{\frac{7}{5}} = 3 + \frac{1}{1 + \frac{2}{5}} = 3 + \frac{1}{1 + \frac{1}{\frac{5}{2}}} = 3 + \frac{1}{1 + \frac{1}{2 + \frac{1}{2}}} = 3 + \frac{1}{1 + \frac{1}{[2;2]}} \\ &= 3 + \frac{1}{[1; 2, 2]} = [3; 1, 2, 2]\end{aligned}$$

Damit können wir auch rationale Kettenbrüche definieren.

3 Definition (rationale Kettenbrüche)**.**
Für $r_0 \in \mathbb{Q}$ und $r_j \in \mathbb{Q} \setminus \{0\}$, $j \in \mathbb{N}$ ist

$$\begin{aligned}[a_0; a_1, ...a_k, ..., a_n] &= a_0 + \cfrac{1}{a_1 + \cfrac{1}{\ddots + \cfrac{1}{\underbrace{\boxed{a_k + \cfrac{1}{\ddots + \cfrac{1}{a_{n-1} + \cfrac{1}{a_n}}}}}_{r_k}}}} \\ &= \left[a_0; a_1, ..., a_{k-1}, \underbrace{[a_k; a_{k+1}, ..., a_n]}_{r_k}\right] \\ &= [a_0; a_1, ..., a_{k-1}, r_k]\end{aligned}$$

ein **rationaler Kettenbruch**.

Jede endliche Folge $a_0, ..., a_n$ von Zahlen ist die initiale Folge einer unendlichen Folge $a = (a_j)_{j\in\mathbb{N}}$. Das ermöglicht es uns, unendliche Kettenbrüche zu betrachten.

4 Definition (unendliche Kettenbrüche)**.**
Zu einer Folge $a = (a_j)_{j\in\mathbb{N}}$ sei

$$[a_0; a_1, a_2, ...] = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + ...}}$$

der **unendliche Kettenbruch**.

An dieser Stelle müssen wir uns allerdings fragen, was der Wert eines solchen unendlichen Ausdrucks eigentlich sein soll. Intuitiv hängt der Grenzwert des Kettenbruches mit dem ursprünglichen, endlichen Segment zusammen:

$$[a_0; a_1, a_2, ...] = \lim_{n\to\infty} [a_0; a_1, ..., a_n]$$

Auf dem aktuellen Stand wissen wir noch nicht, ob, oder wann ein solcher Grenzwert überhaupt existiert. Dafür betrachten wir folgendes Kriterium, welches sogar für Kettenbrüche mit rationalen Koeffizienten funktioniert.

5 Satz (Konvergenzkriterium für Kettenbrüche)**.**
Für $r_j \in \mathbb{Q}$, $r_j > 0$ und $j \in \mathbb{N}_0$ konvergiert der Kettenbruch $[r_0; r_1, r_2, ...]$ genau dann, wenn

$$\sum_{j=0}^{\infty} r_j = \infty.$$

Im Spezialfall $r_j \in \mathbb{N}$ ist dies erfüllt.

Wir sehen bereits, dass diese unendlichen Kettenbrüche insbesondere dann „zahm“ sind, wenn $a_1, a_2, ...$ positive, ganze Zahlen sind. Dann ist der Kettenbruch $[a_1, a_2, ...]$ positiv und mit $a_0 \in \mathbb{Z}$ können wir auch negative Zahlen repräsentieren. Außerdem gilt:

6 Satz.
Jede reelle Zahl $x \in \mathbb{R}$ kann als Kettenbruch $[a_0; a_1, ...]$ mit $a_0 \in \mathbb{Z}$ und $a_j \in \mathbb{N}$, $j \in \mathbb{N}$ geschrieben werden. Dieser Kettenbruch ist endlich, genau dann, wenn $x \in \mathbb{Q}$.

Damit haben wir bereits einen ersten Eindruck zu Kettenbrüchen gewonnen und werden uns im nächsten Kapitel Kettenbrüchen mit Polynomen (anstelle von natürlichen, oder ganzen Zahlen) in den Faktoren widmen. Dabei wollen wir das Grenzwertverhalten von Kettenbrüchen etwas näher betrachten, wodurch wir auf ein Ergebnis stoßen werden, welches für unser Superresolution-Problem bei der Verarbeitung des Eingangssignales sehr nützlich ist.

1.2 Kettenbrüche und Laurent-Reihen

Wir betrachten im Folgenden den Ring $R = \Pi = \mathbb{R}[x]$ von Polynomen in nur einer Variable mit reellen Koeffizienten sowie für $n \in \mathbb{N}_0$ den Vektorraum

$$\Pi_n = \{1, x, ..., x^n\} = \{f \in \Pi : \deg f \leq n\}.$$

Um das Grenzwertverhalten von Kettenbrüchen mit Polynomen als Faktoren zu untersuchen benötigen wir zunächst die Laurent-Reihe, mit deren Hilfe wir später die Konvergenz definieren können. Darüber hinaus können wir wiederum über Kettenbrüche besondere Laurent-Reihen erhalten, mit denen wir es dann im auch Superresolution-Problem zu tun bekommen.

7 Definition.

- Die **Laurent-Reihe** $\lambda(x)$, assoziiert mit der Folge $(\lambda_j)_{j\in\mathbb{N}_0}$ ist definiert als

$$\lambda(x) = \sum_{j=0}^{\infty} \lambda_j x^{-j}$$

- Eine Folge $(\lambda_n(x))_{n\in\mathbb{N}}$, von Laurent-Reihen **konvergiert gegen eine Laurent-Reihe** $\lambda^*(x)$, falls für alle $k \in \mathbb{N}_0$ ein $n_0 \in \mathbb{N}$ existiert, sodass für alle $n \geq n_0$ gilt

$$\lambda_n(x) - \lambda^*(x) = x^{-k-1}\tilde{\lambda_n}(x), \quad \text{d.h.} \quad \lambda_{n,j} = \lambda_j^*,\ j = 0, ..., k-1.$$

  Das bedeutet, dass die ersten k Summanden jeder Reihe $\lambda_n(x)$ mit den ersten k Summanden des Grenzwerts $\lambda^*(x)$ übereinstimmen müssen.

Eine erste, sehr einfache Feststellung ist, dass sich der Kehrwert jedes Polynoms zu einer Potenzreihe erweitert lässt, bei der die ersten Koeffizienten null sind. Diese Beobachtung wird gegen Ende dieses Kapitels noch nützlich sein, in der Beweisskizze zum Fundamentalsatz für Kettenbrüche für Laurent-Reihen, Satz 12.

8 Lemma.
Für $p \in \Pi_n$ mit $p_n \neq 0$ erhält man

$$\frac{1}{p(x)} = \sum_{j=n}^{\infty} \lambda_j x^{-j} =: \lambda(x). \tag{1.2}$$

Außerdem können wir jetzt mithilfe von Laurent-Reihen die Konvergenz unendlicher Kettenbrüchen definieren.

9 Definition.
Ein unendlicher Kettenbruch $[0; a_1, a_2, ...]$, $a_j \in \Pi \setminus \Pi_0$ heißt **konvergent**, falls eine Laurent-Reihe $\lambda(x)$ existiert, sodass

$$\lim_{n\to\infty} \frac{p_n(x)}{q_n(x)} = \lambda(x),$$

wobei $\frac{p_n(x)}{q_n(x)} := [0; a_1, ..., a_n]$.

Es gilt $p_0 = 0$ sowie $p_1 = 1$, da $a_0 = 0$. Es folgt also $\deg q_n > \deg p_n$ und damit

$$\frac{p_n(x)}{q_n(x)} = \sum_{j=\deg q_n - \deg p_n}^{\infty} \underbrace{\tilde{\lambda_j}}_{\text{entstehen durch Division von } p_n \text{ durch } q_n} x^{-j},$$

womit wir wieder eine Laurent-Reihe erhalten. Näheres zur Sinnhaftigkeit der Definition ist allerdings zu finden in [Sau20] - Remark 4.11. Nun können wir die Idee hinter der Konvergenz von Kettenbrüchen illustrieren, indem wir uns daran erinnern, wie die Objekte generiert werden: Wir transferieren das endliche Segment in eine Laurent-Reihe und betrachten den Grenzwert der Folge von Laurent-Reihen im Sinne von Definition 7.

$$[0; a_1, a_2, ...] \leftarrow [0; a_1, ..., a_n] = \frac{p_n}{q_n} = \lambda_n \to \lambda \quad n \to \infty$$

Betrachten wir Kettenbrüche mit Faktoren von Grad 1, so ergibt sich eine recht simple Konvergenz:

10 Satz.
Jeder Kettenbruch der Form $[0; r_1, r_2, ...]$, *wobei* $r_j \in \Pi_1 \setminus \Pi_0$ *für* $j \in \mathbb{N}$, *konvergiert zu einer Laurent-Reihe* $\lambda(x)$, *sodass*

$$\lambda(x) - \frac{p_n(x)}{q_n(x)} = \underbrace{O(x^{-2n-1})}_{\substack{\textit{Rest, der sich von der Laurent-Reihe unterscheidet,} \\ \textit{erst ab Summand mit Koeffizient } \lambda_{-2n-1}}}.$$

Da $r_j \in \Pi_1 \setminus \Pi_0$, also $r_j(x) = \alpha_j x + \beta_j$, $\alpha_j \neq 0$, erhalten wir $\deg(q_n) = \deg(p_n) + 1 = n$. Die Anzahl der erfassten Koeffizienten entspricht also dem doppelten Grad des Nenners, womit der Kettenbruch sehr schnell sehr gut zu der Laurent-Reihe λ passt. Gute Repräsentationen dieser Art für eine gegebene Laurent-Reihe bekommen einen besonderen Namen, auf den wir auch später im Superresolution-Problem stoßen werden:

11 Definition.
Ein unendlicher Kettenbruch heißt **einfach**, wenn er von der Gestalt

$$[0; r_1, r_2, ...] = \frac{1|}{|r_1} + \frac{1|}{|r_2} + ... = \sum_{k=1}^{\infty} \frac{1|}{|r_k}$$

ist, mit $r_j \in \Pi_1 \setminus \Pi_0$. Das heißt, wir arbeiten mit Brüchen

$$\frac{\textcircled{1}|}{|r_1} + \frac{\textcircled{1}|}{|r_2} + ...,$$

die jeweils 1-er in Nenner haben. Dieser unendliche, einfache Kettenbruch heißt **assoziiert** zu der Laurent-Reihe $\lambda(x)$, falls

$$\lambda(x) - \frac{p_n(x)}{q_n(x)} = O(x^{-2n-1}), \quad n \in \mathbb{N}.$$

Das bedeutet wieder

$$\frac{p_n(x)}{q_n(x)} = \sum_{j=0}^{2n} \lambda_j x^{-j} + \sum_{j=2n+1}^{\infty} \underbrace{\gamma_{n,j}}_{\substack{\text{die Koeffizienten, die sich} \\ \text{von } \lambda_j \text{ unterscheiden}}} x^{-j}, \quad n \in \mathbb{N}$$

Nicht jede Laurent-Reihe besitzt einen assoziierten Kettenbruch. Es wird sich allerdings herausstellen, dass die Beschreibung einer Laurent-Reihe, für die ein assoziierter Kettenbruch existiert, noch interessanter ist, da uns diese im Superresolution-Problem die Darstellung des Eingangssignales als Kettenbruch ermöglicht. Das führt uns zum nächsten Resultat.

12 Satz (Fundamentalsatz für Kettenbrüche für Laurent-Reihen)**.**
Eine Laurent-Reihe $\lambda(x)$ besitzt einen assoziierten Kettenbruch $[0; r_1, r_2, ...]$ mit $r_j := \alpha_j x + \beta_j \in \Pi_1 \setminus \Pi_0$ genau dann, wenn $\lambda_0 = 0$ und

$$\det(\Lambda_n) \neq 0, \quad \textit{wobei} \quad \Lambda_n = \begin{bmatrix} \lambda_1 & \lambda_2 & \cdots & \lambda_n \\ \lambda_2 & \lambda_3 & & \vdots \\ \vdots & & \ddots & \lambda_{2n-2} \\ \lambda_n & \cdots & \lambda_{2n-2} & \lambda_{2n-1} \end{bmatrix}, \quad n \in \mathbb{N}.$$

Aus dem Beweis von Satz 12 geht eine explizite Berechnung der Rekursionskoeffizienten α_j und β_j hervor, mit denen der assoziierte Kettenbruch bestimmt werden kann. Diese Herleitung wollen wir etwas genauer betrachten, weshalb zunächst eine nützliche Aussage zu den Konvergenten von endlichen Kettenbrüchen folgt, um den angesprochenen Beweis zumindest skizzieren zu können.

13 Satz.
Für $k \leq n$ erfüllen die Konvergenten $\frac{p_k}{q_k}$ von endlichen Kettenbrüchen $[r_0; r_1, ..., r_n]$ mit $r_j \in R$ die Rekursionsrelation

$$\begin{aligned} p_k &= r_k p_{k-1} + p_{k-2}, \quad \textit{mit} \quad p_{-1} = 1,\ p_0 = r_0, \\ q_k &= r_k q_{k-1} + q_{k-2}, \quad \textit{mit} \quad q_{-1} = 0,\ p_0 = 1, \end{aligned} \tag{1.3}$$

d.h., $\frac{p_0}{q_0} = \frac{r_0}{1} = r_0$ sowie

$$\frac{p_{k-1}}{q_{k-1}} - \frac{p_k}{q_k} = \frac{(-1)^k}{q_{k-1}q_k}, \quad \frac{p_k}{q_k} - \frac{p_{k-2}}{q_{k-2}} = \frac{(-1)^k r_k}{q_{k-2}q_k}, \tag{1.4}$$

sind also teilerfremd.

Beweisskizze zu Satz 12.
Der Kettenbruch ist assoziiert zur Laurent-Reihe $\lambda(x)$, genau dann wenn für jedes $n \in \mathbb{N}$ gilt

$$\begin{aligned} \frac{p_n(x)}{q_n(x)} &= \lambda_0 + \ldots + \lambda_{2n}x^{-2n} + \overbrace{\gamma_{n,2n+1}}^{\text{Bez. für einen Wert, der sich von } \lambda_{2n+1} \text{ unterscheidet. Wert dieses Koeffizienten ist vorerst egal.}} x^{-(2n+1)} + \gamma_{n,2n+2}x^{-(2n+2)} + \ldots, \\ \frac{p_{n+1}(x)}{q_{n+1}(x)} &= \lambda_0 + \ldots + \lambda_{2n+2}x^{-(2n+2)} + \gamma_{n+1,2n+3}x^{-(2n+3)} + \gamma_{n+1,2n+4}x^{-(2n+4)} + \ldots. \end{aligned} \tag{1.5}$$

Mit Satz 13 gilt

$$\begin{aligned} \frac{(-1)^{n+1}}{q_n(x)q_{n+1}(x)} &= \frac{p_n(x)}{q_n(x)} - \frac{p_{n+1}(x)}{q_{n+1}(x)} \\ &= (\gamma_{n,2n+1} - \lambda_{2n+1})x^{-(2n+1)} + (\gamma_{n,2n+2} - \lambda_{2n+2})x^{-(2n+2)} \\ &\quad + (\gamma_{n,2n+3} - \gamma_{n+1,2n+3})x^{-(2n+3)} + \underbrace{\ldots}_{\substack{\text{Hier werden, wie auch beim vorherigen Summanden,} \\ \text{nur noch } \gamma's \text{ voneinander abgezogen.}}}. \end{aligned} \tag{1.6}$$

Als nächstes können wir die Rekursionsformel (1.3) verwenden, womit wir erhalten, dass die Nenner der Konvergenten einfacher Kettenbrüche der Form

$$q_n(x) = r_n(x)q_{n-1}(x) + q_{n-2}(x) = (\alpha_n x + \beta_n)q_{n-1}(x) + q_{n-2}(x), \quad n \in \mathbb{N}$$

sein müssen. Über Induktion lässt sich weiter zeigen, dass

$$q_n(x) = \left(\prod_{j=1}^{n} \alpha_j\right)\left(x^n + x^{n-1}\sum_{j=1}^{n}\frac{\beta_j}{\alpha_j}\right) + r_{n-2}.$$

für ein $r_{n-2} \in \Pi_{n-2}$ Eine genauere Ausführung davon ist zu finden in [Sau20] im Beweis zu Theorem 4.4.

Damit erhalten wir jetzt aber

$$
\begin{aligned}
&q_{n+1}(x)q_n(x)\\
&= \left(\left(\prod_{j=1}^{n+1}\alpha_j\right)\left(x^{n+1}+x^n\sum_{j=1}^{n+1}\frac{\beta_j}{\alpha_j}\right)+r_{n-1}\right)\\
&\quad\cdot\left(\left(\prod_{j=1}^{n}\alpha_j\right)\left(x^{n}+x^{n-1}\sum_{j=1}^{n}\frac{\beta_j}{\alpha_j}\right)+r_{n-2}\right)\\
&= \alpha_{n+1}\left(\prod_{j=1}^{n}\alpha_j\right)^2 x^{2n+1}+\left(\prod_{j=1}^{n+1}\alpha_j\sum_{j=1}^{n+1}\frac{\beta_j}{\alpha_j}\prod_{j=1}^{n}\alpha_j+\prod_{j=1}^{n+1}\alpha_j\prod_{j=1}^{n}\alpha_j\sum_{j=1}^{n}\frac{\beta_j}{\alpha_j}\right)x^{2n}+\underbrace{\ldots}_{\substack{\text{hier uninteressanter Rest mit}\\ \text{Koeffizienten für Grade} \le 2n-1}}\\
&= \alpha_{n+1}\left(\prod_{j=1}^{n}\alpha_j\right)^2 x^{2n+1}+\left(\prod_{j=1}^{n}\alpha_j\right)^2\left(\beta_{n+1}+2\alpha_{n+1}\sum_{j=1}^{n}\frac{\beta_j}{\alpha_j}\right)x^{2n}+\ldots .
\end{aligned}
\tag{1.7}
$$

Lemma 8 bzw. der Beweis dieses Lemmas (nachzulesen in [Sau20] - Beweis zu Lemma 4.1) liefert $\frac{1}{p_n}=\lambda_n$ in (1.2) und damit

$$
\frac{1}{q_{n+1}(x)q_n(x)} = \alpha_{n+1}^{-1}\left(\prod_{j=1}^{n}\alpha_j\right)^{-2} x^{-(2n+1)}+\ldots . \tag{1.8}
$$

Vergleichen wir (1.6) mit (1.8) erhalten wir

$$
(-1)^{n+1}\left(\gamma_{n,2n+1}-\lambda_{2n+1}\right)x^{-(2n+1)} = \alpha_{n+1}^{-1}\left(\prod_{j=1}^{n}\alpha_j\right)^{-2} x^{-(2n+1)},
$$

was äquivalent ist zu

$$
\begin{aligned}
\alpha_{n+1} &= \frac{(-1)^{n+1}}{\prod_{j=1}^{n}\alpha_j^2\left(\gamma_{n,2n+1}-\lambda_{2n+1}\right)}, \quad \text{oder}\\
\gamma_{n,2n+1}-\lambda_{2n+1} &= \frac{(-1)^{n+1}}{\alpha_{n+1}\cdot\prod_{j=1}^{n}\alpha_j^2}
\end{aligned}
\quad . \tag{1.9}
$$

Fassen wir zusammen: Die Existenz eines assoziierten Kettenbruches mit $r_j \in \Pi_1\backslash\Pi_0$ ist äquivalent zur Gültigkeit von (1.9) mit $\alpha_j \neq 0$ für alle j, was wiederum äquivalent ist zu $\gamma_{n,2n+1} \neq \lambda_{2n+1}$.

Um zu sehen, was das bedeutet, multiplizieren wir die erste Zeile von (1.5) mit $q_n(x)$:

$$
p_n(x) = \left(\sum_{j=0}^{2n}\lambda_j x^{-j}+\sum_{j=2n+1}^{\infty}\gamma_{n,j}x^{-j}\right)\left(\sum_{k=0}^{n}q_{n,k}x^k\right).
$$

Durch Umformungen, welche in [Sau20] genauer ausgeführt werden, erhalten wir

$$p_n(x) = \underbrace{\sum_{j=0}^{n} \eta_{-j} x^j}_{\substack{\text{nicht-negative} \\ \text{Exponenten}}} + \underbrace{\sum_{j=1}^{n} \eta_j x^{-j} + \eta_{n+1} x^{-(n+1)}}_{\substack{\text{negative} \\ \text{Exponenten}}} + \underbrace{O(x^{-(n+2)})}_{\text{Rest}},$$

mit

$$\begin{bmatrix} p_{n,n} \\ \vdots \\ p_{n,0} \end{bmatrix} = \begin{bmatrix} \eta_{-n} \\ \vdots \\ \eta_0 \end{bmatrix} = \begin{bmatrix} \lambda_0 & & \\ \vdots & \ddots & \\ \lambda_n & \cdots & \lambda_0 \end{bmatrix} \begin{bmatrix} q_{n,n} \\ \vdots \\ q_{n,0} \end{bmatrix} \quad \text{und} \tag{1.10}$$

$$\begin{bmatrix} \eta_1 \\ \vdots \\ \eta_n \\ \eta_{n+1} \end{bmatrix} = \underbrace{\begin{bmatrix} \lambda_{n+1} & \lambda_n & \cdots & \lambda_1 \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{2n} & \lambda_{2n-1} & \cdots & \lambda_n \\ \gamma_{n,2n+1} & \lambda_{2n} & \cdots & \lambda_{n+1} \end{bmatrix}}_{=:M} \begin{bmatrix} q_{n,n} \\ \vdots \\ q_{n,0} \end{bmatrix}. \tag{1.11}$$

$p_n(x)$ ist ein Polynom, d.h. es muss gelten $\eta_1 = \ldots = \eta_{n+1} = 0$. Da $q \neq 0$, muss gelten $\det(M) = 0$. Wir erinnern uns an die zweite Gleichung von (1.9):

$$\gamma_{n,2n+1} = \lambda_{2n+1} + \frac{(-1)^{n+1}}{\alpha_{n+1} \cdot \prod_{j=1}^{n} \alpha_j^2},$$

substituieren dies in die Marix M und ordnen die Spalten neu an (wodurch sich der Wert der Determinante nicht verändert):

$$0 = \det M = \det \begin{bmatrix} \lambda_1 & \cdots & \lambda_n & \lambda_{n+1} \\ \vdots & \ddots & \vdots & \vdots \\ \lambda_n & \cdots & \lambda_{2n-1} & \lambda_{2n} \\ \lambda_{n+1} & \cdots & \lambda_{2n} & \gamma_{n,2n+1} \end{bmatrix}.$$

Mithilfe der Multilinearität der Determinante und dem Entwicklungssatz (Entwicklung nach der $(n+1)$-ten Spalte) erhalten wir

$$\begin{aligned} 0 &= \det \begin{bmatrix} \lambda_1 & \cdots & \lambda_n & \lambda_{n+1} \\ \vdots & \ddots & \vdots & \vdots \\ \lambda_n & \cdots & \lambda_{2n-1} & \lambda_{2n} \\ \lambda_{n+1} & \cdots & \lambda_{2n} & \lambda_{2n+1} \end{bmatrix} \\ &\quad + \left(\alpha_{n+1} \prod_{j=1}^{n} \alpha_j^2 \right)^{-1} \det \begin{bmatrix} \lambda_1 & \cdots & \lambda_n & 0 \\ \vdots & \ddots & \vdots & \vdots \\ \lambda_n & \cdots & \lambda_{2n-1} & 0 \\ \lambda_{n+1} & \cdots & \lambda_{2n} & (-1)^{n+1} \end{bmatrix} \\ &= \det \Lambda_{n+1} + \left(\alpha_{n+1} \prod_{j=1}^{n} \alpha_j^2 \right)^{-1} \underbrace{(-1)^{n+1+n+1}}_{=1} (-1)^{n+1} \det(\Lambda_n). \end{aligned}$$

Damit gilt also

$$\det \Lambda_{n+1} = (-1)^n \frac{\det \Lambda_n}{\alpha_{n+1} \prod_{j=1}^{n} \alpha_j^2} \tag{1.12}$$

sowie

$$\alpha_{n+1} = (-1)^n \left(\prod_{j=1}^{n} \alpha_j^2 \right)^{-1} \frac{\det \Lambda_n}{\det \Lambda_{n+1}}, \tag{1.13}$$

was uns bereits die Parameter α_j für (1.15) liefert.

Ziehen wir allerdings noch die finale Schlussfolgerung. Ist also der Kettenbruch mit Koeffizienten $r_j(x) = \alpha_j x + \beta_j$ und $\alpha_j \neq 0$ assoziiert zur Laurent-Reihe $\lambda(x)$, dann liefert uns (1.12) induktiv über n, dass $\det \Lambda_n \neq 0$.

Für die Rückrichtung begründen wir zunächst die Notwendigkeit von $\lambda_0 = 0$. Betrachten wir (1.10) unter dem Aspekt, dass $\deg p_n = n - 1$, also $0 = p_{n,n} = \lambda_0 q_{n,n}$ und $\deg q_n = n$, also $q_{n,n} \neq 0$, stellen wir fest, dass $\lambda_0 = 0$ gelten muss. Weiter gibt uns (1.13) eine Formel zur expliziten Berechnung der Koeffizienten α_n. Die rechte Seite der Gleichung zeigt, dass $\alpha_n \neq 0$ und somit die Komponenten $r_j = \alpha_j x + \beta_j$ nichtkonstante Polynome sind, solange die Bedingung $\det \Lambda_n \neq 0$ erfüllt ist. □

Für die Parameter β_j betrachten wir den zweiten Term $\neq 0$ in der Laurent-Erweiterung von $\frac{1}{q_{n+1} q_n}$, welcher nach (1.7) und Lemma 8 bestimmt werden kann, indem das System

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} p_{2n+1} & 0 \\ p_{2n} & p_{2n+1} \end{bmatrix} \begin{bmatrix} \theta_{2n+1} \\ \theta_{2n+2} \end{bmatrix}$$

$$\text{mit} \quad p_{2n+1} = \alpha_{n+1} \prod_{j=1}^{n} \alpha_j^2 \quad \text{und} \quad p_{2n} = \prod_{j=1}^{n} \alpha_j^2 \left(\beta_{n+1} + 2\alpha_{n+1} \sum_{j=1}^{n} \frac{\beta_j}{\alpha_j} \right)$$

gelöst wird. Es gilt also

$$\theta_{2n+1} = \frac{1}{p_{2n+1}} \quad \text{und}$$

$$\theta_{2n+2} = -\frac{p_{2n}}{p_{2n+1}} \theta_{2n+1} = -\left(\prod_{j=1}^{n+1} \alpha_j \right)^{-2} \left(\beta_{n+1} + 2\alpha_{n+1} \sum_{j=1}^{n} \frac{\beta_j}{\alpha_j} \right).$$

Koeffizientenvergleich liefert, wie bereits in der Beweisskizze zu Satz 12,

$$(-1)^{n+1} (\gamma_{n,2n+2} - \lambda_{2n+2}) = -\left(\prod_{j=1}^{n+1} \alpha_j \right)^{-2} \left(\beta_{n+1} + 2\alpha_{n+1} \sum_{j=1}^{n} \frac{\beta_j}{\alpha_j} \right),$$

bzw.

$$\gamma_{n,2n+2} = \lambda_{2n+2} + (-1)^n \left(\prod_{j=1}^{n+1} \alpha_j \right)^{-2} \left(\beta_{n+1} + 2\alpha_{n+1} \sum_{j=1}^{n} \frac{\beta_j}{\alpha_j} \right). \tag{1.14}$$

Nach etwas Umstellen ergibt sich

$$\beta_{n+1} = (-1)^n \left(\prod_{j=1}^{n+1} \alpha_j \right)^2 (\gamma_{n,2n+2} - \lambda_{2n+2}) - 2\alpha_{n+1} \sum_{j=1}^{n} \frac{\beta_j}{\alpha_j}.$$

Für eine explizite Bestimmung der β_j erweitern wir die Idee aus der Beweisskizze und nutzen aus, dass

$$0 = \begin{bmatrix} \eta_1 \\ \vdots \\ \eta_n \\ \eta_{n+2} \end{bmatrix} = \underbrace{\begin{bmatrix} \lambda_1 & \cdots & \lambda_n & \lambda_{n+1} \\ \vdots & & \vdots & \vdots \\ \lambda_n & \cdots & \lambda_{2n-1} & \lambda_{2n} \\ \lambda_{n+2} & \cdots & \gamma_{n,2n+1} & \gamma_{n,2n+2} \end{bmatrix}}_{=:M} \begin{bmatrix} q_{n,0} \\ \vdots \\ q_{n,n} \end{bmatrix}$$

und die gleiche Argumentation für $det(M) = 0$.

Durch mehrfaches Verwenden der Multilinearität der Determinante, der Eigenschaft (1.14), der zweiten Gleichung in (1.9) und des Entwicklungssatzes erhält man schließlich

$$(-1)^n \left(\prod_{j=1}^{n+1} \alpha_j \right)^2 \det \Lambda'_{n+2} = \left(\beta_{n+1} + 2\alpha_{n+1} \sum_{j=1}^{n} \frac{\beta_j}{\alpha_j} \right) \det \Lambda_n - \alpha_{n+1} \det \Lambda'_{n+1}$$

wobei

$$\Lambda'_n := \begin{bmatrix} \lambda_1 & \cdots & \lambda_{n-1} & 0 \\ \vdots & & \vdots & \vdots \\ \lambda_{n-2} & \cdots & \lambda_{2n-4} & 0 \\ \lambda_{n-1} & \cdots & \lambda_{2n-3} & 1 \\ \lambda_n & \cdots & \lambda_{2n-2} & 0 \end{bmatrix}.$$

Mit Umstellen erhalten wir eine Gleichung für die β_j, wie sie in (1.16) festgehalten wird.

Die Bestimmung der Rekursionskoeffizienten für den assoziierten Kettenbruch einer Laurent-Reihe wollen wir zum Schluss dieses Kapitels noch mit einem Satz festhalten.

14 Korollar.
Für $n \in \mathbb{N}$ sind die Rekursionskoeffizienten für die Konvergenten und somit die Koeffizienten $r_n(x) = \alpha_n x + \beta_n$ der Kettenbrucherweiterung rekursiv gegeben durch

$$\alpha_{n+1} = (-1)^n \left(\prod_{j=1}^{n} \alpha_j \right)^{-2} \frac{\det(\Lambda_n)}{\det(\Lambda_{n+1})} = \frac{1}{\alpha_n} \frac{(\det(\Lambda_n))^2}{\det(\Lambda_{n+1}) \cdot \det(\Lambda_{n-1})}, \tag{1.15}$$

$$\beta_{n+1} = \alpha_{n+1} \left(\frac{\det(\Lambda'_{n+2})}{\det(\Lambda_{n+1})} + \frac{\det(\Lambda'_{n+1})}{\det(\Lambda_n)} - 2 \sum_{j=1}^{n} \frac{\beta_j}{\alpha_j} \right), \tag{1.16}$$

beginnend mit $\alpha_1 = \dfrac{1}{\lambda_1}$, $\beta_1 = -\dfrac{\lambda_2}{\lambda_1^2}$ *und* $\Lambda'_n = \begin{bmatrix} \lambda_1 & \cdots & \lambda_{n-1} & 0 \\ \vdots & \ddots & \vdots & \vdots \\ \lambda_{n-2} & \cdots & \lambda_{2n-4} & 0 \\ \lambda_{n-1} & \cdots & \lambda_{2n-3} & 1 \\ \lambda_n & \cdots & \lambda_{2n-2} & 0 \end{bmatrix}$.

1.3 Prony's Problem

In diesem Kapitel wird nun behandelt, wie mithilfe der bisher behandelten Theorie das Superresolution-Problem gelöst werden kann. Explizit geht es darum, Exponentialsummen „zurückzugewinnen", wobei wir wieder Kettenbrüche verwenden werden. Die Exponentialsumme ist von der Form

$$f(x) = \sum_{j=1}^{s} f_j e^{\omega_j x}, \tag{1.17}$$

mit Frequenzen $\omega_j \in \mathbb{R}+i\mathbb{T}$ und Gewichten $f_j \neq 0$. Dabei bezeichnet $\mathbb{T} = \mathbb{R}/2\pi\mathbb{Z} \simeq [0, 2\pi]$ den Torus. Die Summe soll anhand von Stichproben zurückerworben werden. Beispielsweise stehen endlich viele Funktionswerte zur Verfügung, von denen wir annehmen, dass sie gleichmäßig verteilt sind:

$$f(0), ..., f(N) \text{ mit } N \in \mathbb{N}.$$

Natürlich wird N von s abhängen, zumindest falls wir eine Rekonstruktion von f erhalten wollen.

15 Bemerkung.
Wir normalisieren die Frequenzen ω_j auf

$$\mathbb{R} + i\mathbb{T} = \mathbb{R} + i(\mathbb{R}/2\pi\mathbb{Z}) \simeq \mathbb{R} + i\,[-\pi, \pi)\,,$$

um Mehrdeutigkeiten in der Repräsentation (1.17) zu vermeiden, die das Problem unlösbar machen würden, wie z.B. die Funktion

$$\sin(\pi) = \frac{1}{2i}\left(e^{i\pi} - e^{-i\pi}\right)$$

zu erzeugen, welche mit keiner Teilmenge von $\mathbb{Z}$ rekonstruiert werden kann.

Die Bedingung $f_j \neq 0$ für alle j gestaltet die Darstellung sparsamer, da sie so keine unnötigen Frequenzen ohne Gewicht enthält.

Eine Stichprobe auf $0, ..., N$ schränkt nicht ein, da

$$f(ax+b) = \sum_{j=1}^{s} f_j e^{\omega_j(ax+b)} = \sum_{j=1}^{s} \overbrace{\left(e^{\omega_j b} f_j\right)}^{\tilde{f}_j} \cdot e^{\overbrace{a\omega_j}^{\tilde{\omega}_j} x} =: \sum_{j=1}^{s} \tilde{f}_j e^{\tilde{\omega}_j x}$$

zeigt, dass jede affine Transformation nur die Koeffizienten und die Frequenzen ändert, jedoch keinen Einfluss auf die Struktur, oder die Lösbarkeit des Problems hat. Anders gesagt: Eine gleiche Abstände aufweisende Stichprobe, basierend auf den Knoten

$$x_0 + kh, \quad k = 0, ..., N, \quad x_0 \in \mathbb{R}, \quad h > 0$$

kann einfach auf eine Stichprobe an den Werten $0, ..., N$ reduziert werden, ohne das Wesen des Problems zu ändern.

Der interessante Teil von Prony's Problem besteht darin, die Frequenzen zu rekonstruieren. Sobald diese bekannt sind, bleibt lediglich das lineare System

$$f(k) = \sum_{j=1}^{s} f_j e^{\omega_j k}, \quad k = 0, ..., N,$$

zu lösen, welches auch in Form einer Matrix

$$\begin{bmatrix} f(0) \\ \vdots \\ f(N) \end{bmatrix} = \begin{bmatrix} 1 & \cdots & 1 \\ e^{\omega_1} & \cdots & e^{\omega_s} \\ \vdots & \ddots & \vdots \\ e^{N\omega_1} & \cdots & e^{N\omega_s} \end{bmatrix} \cdot \begin{bmatrix} f_1 \\ \vdots \\ f_s \end{bmatrix}$$

dargestellt werden kann, oder noch kompakter:

$$[f(j) : j = 0, ..., N] = V^T [f_j : j = 1, ..., s].$$

Dabei besitzt die Matrix V Rang s, falls die ω_j alle voneinander verschieden sind und $N \geq s - 1$, sodass die Koeffizienten von s Stichproben eindeutig bestimmbar sind, sobald die Frequenzen bekannt sind. Um die Frequenzen ω_j und damit die Matrix V zu bestimmen hatte Prony einen genialen Einfall, der auf der folgenden Idee beruht:

Sei $p(x) = p_0 + p_1 x + ... + p_m x^m$ ein Polynom von Grad $m \geq n$. Wir betrachten für ein fixes $0 \leq \boldsymbol{j} \leq N - m$:

$$\sum_{k=0}^{m} f(j+k)p_k = \sum_{k=0}^{m} \sum_{l=1}^{s} f_l e^{\omega_l (j+k)} p_k = \sum_{l=1}^{s} f_l e^{\omega_l j} \underbrace{\sum_{k=0}^{m} p_k \left(e^{\omega_l}\right)^k}_{=p(e^{\omega_l})} = \sum_{l=1}^{s} f_l e^{\omega_l j} \cdot p(e^{\omega_l}).$$

Dies kann ebenfalls in Matrixschreibweise ausgedrückt werden:

$$\begin{bmatrix} f(0) & \cdots & f(m) \\ f(1) & \cdots & f(m+1) \\ \vdots & \ddots & \vdots \\ f(N-m) & \cdots & f(N) \end{bmatrix} \begin{bmatrix} p_0 \\ \vdots \\ p_m \end{bmatrix} = \begin{bmatrix} 1 & \cdots & 1 \\ e^{\omega_1} & \cdots & e^{\omega_s} \\ \vdots & \ddots & \vdots \\ e^{(N-m)\omega_1} & \cdots & e^{(N-m)\omega_s} \end{bmatrix} \begin{bmatrix} f_1 & & \\ & \ddots & \\ & & f_s \end{bmatrix} \begin{bmatrix} p(e^{\omega_1}) \\ \vdots \\ p(e^{\omega_s}) \end{bmatrix} \tag{1.18}$$

Weiter erhalten wir damit das folgende Resultat, welches den Kern von Prony's Methode beschreibt.

16 Lemma.
Sei f wie in (1.17) und $N \geq 2m - 1$, $m \geq s$, dann gilt:

$$\sum_{k=0}^{m} f(j+k)p_k = 0, \quad j = 0, ..., N - m \quad \Leftrightarrow \quad p(e^{\omega_j}) = 0, \quad j = 1, ..., s,$$

wobei $p = p_0 + p_1 x + ... + p_m x^m$.

Für das Polynom aus dem letzten Lemma gibt es auch einen besonderen Namen.

17 Definition.
Das Polynom p kleinsten Grades, mit $p(e^{\omega_j}) = 0$ für $j = 0, ..., s$ heißt **Prony-Polynom** für die Funktion $f(x) = \sum_{j=1}^{s} f_j e^{\omega_j x}$ von (1.17).

Das Prony-Polynom ist immer ein Polynom vom Grad s (Anzahl der Summanden in (1.17)) und von der Form

$$p = p_s \prod_{j=1}^{s} (\cdot - e^{\omega_j}), \quad p_s \neq 0.$$

Lemma 16 liefert uns bereits eine Möglichkeit, Prony's Problem zu lösen, d.h. (1.17) „zurückzugewinnen", vorausgesetzt die Zahl n der Exponentiale ist bekannt:

Bestimme den Kern der Hankel-Matrix (eine Matrix A mit Einträgen $a_{j,k}$ für die gilt $a_{j,k} = a_{j+k}$; wird im unendlichen Fall Hankel-Operator für die Folge $(a_n)_{n\in\mathbb{N}}$ genannt)

$$F_s := \begin{bmatrix} f(0) & \cdots & f(s) \\ \vdots & & \vdots \\ f(s) & \cdots & f(2s) \end{bmatrix} \in \mathbb{C}^{s+1\times s+1},$$

identifiziere die Lösung $p \in \mathbb{C}^{s+1}$, sodass $F_s p = 0$, $p \neq 0$ mit einem Polynom $p(x)$ und berechne seine Nullstellen, welche genau die gesuchten Zahlen e^{ω_j} sind für $j = 1, ..., s$.

Nun brauchen wir noch eine Möglichkeit, die Lösung p zu bestimmen. Diese kann mithilfe von Satz 12, in Verbindung mit Korollar 14 erarbeitet werden. Das soll im Folgenden etwas genauer formuliert werden.

Doch zunächst definieren wir die Dirac-Distribution, die wir später auch in unserem Grundmodell verwenden sowie die Eigenschaft der Nicht-Entartung, um den für die Lösung des Superresolution nötigen Satz zu formulieren.

18 Definition.
Die **Dirac-Distribution** δ_x für $x \in \mathbb{R}$ ist definiert durch die Bedingung

$$\int_{\mathbb{R}} f(t)\delta_x(t)dt = f(x), \quad f \in \mathbb{C}_{00}(\mathbb{R}),$$

wobei $C_{00}(\mathbb{R})$ alle (reell- und komplexwertigen) Funktionen auf $\mathbb{R}$ bezeichnet, die einen kompakten Träger besitzen.

19 Definition.
Der **Rang des Hankel-Operators** M ist definiert als

$$\operatorname{rang}(M) := \sup_{n\in\mathbb{N}_0} \operatorname{rang}(M_n) = \sup_{n\in\mathbb{N}_0} \operatorname{rang}\left(\begin{bmatrix} \mu_0 & \cdots & \mu_n \\ \vdots & & \vdots \\ \mu_n & \cdots & \mu_{2n} \end{bmatrix}\right).$$

Die Folge μ heißt **nicht entartet**, falls sie einen Hankel-Operator mit endlichem Rang definiert, mit der Eigenschaft, dass mit $n := \operatorname{rang}(M)$ gilt:

$$1 = \operatorname{rang}(M_0) < \operatorname{rang}(M_1) < \cdots < \operatorname{rang}(M_{n-1}) = \operatorname{rang}(M_n) = \cdots = \operatorname{rang}(M).$$

Wir kennen bereits Hankel-Operatoren von endlichem Rank. Betrachten wir

$$\mu_k = \sum_{j=1}^{s} f_j e^{\omega_j k}, \quad k \in \mathbb{N}_0$$

wie in Prony's Problem, dann erhalten wir

$$\begin{aligned} V_k^T F V_k : &= \underbrace{\begin{bmatrix} 1 & \cdots & 1 \\ e^{\omega_1} & \cdots & e^{\omega_s} \\ \vdots & \ddots & \vdots \\ e^{k\omega_1} & \cdots & e^{k\omega_s} \end{bmatrix}}_{k+1\times s} \underbrace{\begin{bmatrix} f_1 & & \\ & \ddots & \\ & & f_s \end{bmatrix}}_{s\times s} \underbrace{\begin{bmatrix} 1 & e^{\omega_1} & \cdots & e^{k\omega_1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & e^{\omega_s} & \cdots & e^{k\omega_s} \end{bmatrix}}_{s\times k+1} \\ &= \begin{bmatrix} f_1 & \cdots & f_s \\ f_1 e^{\omega_1} & \cdots & f_s e^{\omega_s} \\ \vdots & & \vdots \\ f_1 e^{k\omega_1} & \cdots & f_s e^{k\omega_s} \end{bmatrix} \begin{bmatrix} 1 & e^{\omega_1} & \cdots & e^{k\omega_1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & e^{\omega_s} & \cdots & e^{k\omega_s} \end{bmatrix} \\ &= \begin{bmatrix} \sum_{j=1}^s f_j e^0 & \sum_{j=1}^s f_j e^{\omega_j} & \cdots & \sum_{j=1}^s f_j e^{k\omega_j} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{j=1}^s f_j e^{k\omega_j} & \sum_{j=1}^s f_j e^{(k+1)\omega_j} & \cdots & \sum_{j=1}^s f_j e^{2k\omega_j} \end{bmatrix} \\ &= \begin{bmatrix} \mu_0 & \cdots & \mu_k \\ \vdots & & \vdots \\ \mu_k & \cdots & \mu_{2k} \end{bmatrix} = M_k. \end{aligned}$$

Mit $\operatorname{rang}(V_k) = \min(k+1, s)$ und der invertierbaren Matrix F können wir also festhalten, dass

$$\operatorname{rang}(M) = s. \tag{1.19}$$

D.h. wir können nun unsere Erkenntnisse mit Satz 12 kombinieren. Wir verwenden eine geshiftete Folge λ, um die Bedingung $\lambda_0 = 0$ zu erfüllen und mit (1.19) decken wir $\det(\Lambda_n) \neq 0$ ab:

20 Satz.
Falls für ein exponentielles f der Form

$$f(x) = \sum_{j=1}^{s} f_j e^{\omega_j x}, \quad \omega_j \in \mathbb{R} + i\mathbb{T}, \quad f_j \neq 0 \quad \textit{(vgl. (1.17))}$$

die Folge

$$\lambda = (f(j-1))_{j\in\mathbb{N}}, \quad \textit{d.h. } \lambda_k = \sum_{j=1}^{s} f_j e^{\omega_j (k-1)} \textit{ mit } \lambda_0 = 0, \quad \textit{für } k \in \mathbb{N},$$

nicht entartet ist, dann terminiert die Kettenbrucherweiterung von $\lambda(x)$ nach n Schritten und der Nenner dieser Konvergenten ist das Prony-Polynom.

Nach Lemma 16 müssen wir zur Lösung des Superresolution-Problems noch die Nullstellen des Prony-Polynoms bestimmen, um die Frequenzen zu erhalten. Dem wollen wir uns im folgenden Abschnitt widmen.

1.4 Bestimmung von Nullstellen

Die Bestimmung der Nullstellen des Polynoms wird mithilfe einer Methode aus der linearen Algebra erfolgen. Dafür müssen wir uns allerdings ein wenig in die Theorie einarbeiten.

Sei zu diesem Zweck

$$f(x) = x^{n+1} + \sum_{j=0}^{n} f_j x^j = (x - \zeta_0) \cdot ... \cdot (x - \zeta_n)$$

ein normiertes Polynom von Grad $n + 1$. Das Hauptideal

$$\langle f \rangle := f \cdot \Pi = \{f \cdot p : p \in \Pi\}$$

definiert eine Menge von Äquivalenzklassen $\Pi / \langle f \rangle$, wobei

$$p \equiv q \Leftrightarrow p - q \in \langle f \rangle$$

ein Vielfaches von f ist. Der eindeutige Repräsentant für die Äquivalenzklasse von $p \in \Pi$ ist der Rest $r \in \Pi_n$ der Division:

$$p = qf + r, \quad \deg(r) < \deg(f).$$

21 Definition (Modulo Operation)**.**
Der Rest r der Division von p durch f wird bezeichnet als $(p)_{\langle f \rangle}$ und genannt p **modulo** $\langle f \rangle$, oder kurz p **modulo** f.

Trotz, dass $\Pi / \langle f \rangle \simeq \Pi_n$ die gleiche Menge von Polynomen von Grad maximal n ist und die Vektorraumeigenschaften erbt, besitzt $\Pi / \langle f \rangle$ noch mehr Struktur und zwar die einer Algebra, sobald wir eine Multiplikation für $\Pi / \langle f \rangle$ definiert haben.

22 Definition (Multiplikation in $\Pi / \langle f \rangle$)**.**
Die **Multiplikation** von $p, q \in \Pi / \langle f \rangle$ ist definiert als

$$p \cdot q := (pq)_{\langle f \rangle} \in \Pi / \langle f \rangle .$$

Für ein fixes $q \in \Pi / \langle f \rangle$ ist die Operation

$$\Pi / \langle f \rangle \ni p \mapsto pq = (pq)_{\langle f \rangle}$$

eine lineare Operation in p und kann mit Bezug zu jeder Basis $B = \{b_0, ..., b_n\}$ von $\Pi / \langle f \rangle \simeq \Pi_n$ von einer Matrix $M_{q,B} \in \mathbb{C}^{(n+1)\times(n+1)}$ repräsentiert werden. Dies ist definiert durch

$$q \cdot b_j = \sum_{k=0}^{n} (M_{q,B})_{kj} \, b_k \quad j = 0, ..., n.$$

Diese Matrix wird **Begleitmatrix**, oder **Multiplikationstabelle**, für $\Pi / \langle f \rangle$ mit Bezug zur Basis b genannt.

23 Beispiel.
Für die Basis $B = \{1, ..., x^n\}$ und $q(x) = x$ erhalten wir

$$x^k q(x) = x^{k+1}, \quad k = 0, ..., n-1 \quad \text{und}$$
$$x^n q(x) = x^{n+1} = f(x) - \sum_{j=0}^{n} f_j x^j \equiv -\sum_{j=0}^{n} f_j x^j.$$

Das liefert

$$M_{(\cdot),B} = \begin{bmatrix} 0 & 0 & \cdots & 0 & -f_0 \\ 1 & 0 & & & \\ 0 & 1 & & & \vdots \\ \vdots & & \ddots & & \\ 0 & \cdots & 0 & 1 & -f_n \end{bmatrix}.$$

Das ist die bekannte Frobenius Begleitmatrix des normierten Polynoms f. Variiert man die Basis, verändert sich auch die Multiplikationstabelle, was wertvolle Informationen über f offenbart. Für ein weiteres Beispiel mit veränderter Basis siehe [Sau20] Example 4.3.

Nun können wir uns dem Satz widmen, mit dessen Hilfe wir die Nullstellen des Prony-Polynoms bestimmen wollen.

24 Satz.
Sei $f \in \Pi_{n+1}$ ein normiertes Polynom mit einfachen Nullstellen $\zeta_0, ..., \zeta_n$. Dann besitzen alle Multiplikationstabellen für $\Pi/\langle f \rangle$ die Eigenwerte $\zeta_0, ..., \zeta_n$ und die zugehörigen Eigenvektoren sind $\ell_0, ..., \ell_n$ mit $\ell_j = \prod_{k \neq j}(\cdot - \zeta_j)$.

Für das Aufstellen der Multiplikationstabelle für das Prony-Polynom q_n erinnern wir uns an die Relation (1.3) in Satz 13. Wir werden später im Superresolution-Problem Polynome der Form

$$q_k(x) = (\alpha_k x + \beta_k) q_{k-1}(x) + q_{k-2}(x) \tag{1.20}$$

haben. Satz 24 gilt allerdings nur für normierte Polynome, weshalb wir vorerst mit den normierten Polynomen

$$q_k(x) = (x + \beta_n) q_{k-1}(x) + \gamma_n q_{k-2}(x)$$

arbeiten und auf das Problem zurückkommen.

Da $q_k(x) \in \Pi_k$ mit $q_k(x) = x^k + ...$ sind die Polynome linear unabhängig und bilden

eine Basis B von $\Pi_n \cong \Pi/\langle q_{n+1}\rangle$. Mit dieser Basis haben wir

$$\begin{aligned} xq_k(x) &= \underbrace{((x+\beta_{k+1})q_k + \gamma_{k+1}q_{k-1})}_{=q_{k+1}(x)} - \beta_{k+1}q_k - \gamma_{k+1}q_{k-1} \\ &= q_{k+1}(x) - \beta_{k+1}q_k - \gamma_{k+1}q_{k-1}, \end{aligned}$$

was uns die Multiplikationstabelle

$$M_{(\cdot),B} = \begin{bmatrix} -\beta_1 & -\gamma_2 & & & & \\ 1 & -\beta_2 & -\gamma_3 & & & \\ & 1 & -\beta_3 & \ddots & & \\ & & 1 & \ddots & -\gamma_{n-1} & \\ & & & \ddots & -\beta_{n-1} & -\gamma_n \\ & & & & 1 & -\beta_n \end{bmatrix}$$

liefert. Da wir allerdings, wie bereits angesprochen, mit Polynomen der Form (1.20) arbeiten müssen, werden diese normiert, indem wir durch α_k teilen und damit die Multiplikationstabelle

$$M = \begin{bmatrix} -\frac{\beta_1}{\alpha_1} & -\frac{1}{\alpha_2} & & & & \\ \frac{1}{\alpha_1} & -\frac{\beta_2}{\alpha_2} & -\frac{1}{\alpha_3} & & & \\ & \frac{1}{\alpha_2} & -\frac{\beta_3}{\alpha_3} & \ddots & & \\ & & \frac{1}{\alpha_3} & \ddots & -\frac{1}{\alpha_{n-1}} & \\ & & & \ddots & -\frac{\beta_{n-1}}{\alpha_{n-1}} & -\frac{1}{\alpha_n} \\ & & & & \frac{1}{\alpha_{n-1}} & -\frac{\beta_n}{\alpha_n} \end{bmatrix} \tag{1.21}$$

erhalten.

Nun können wir uns endlich der Bildverarbeitung widmen und ein paar letzte Begriffe, die wir für das Superresolution-Problem benötigen, einführen.

1.5 Signale und Filter

Da wir später zunächst ein Signal brauchen, welches wir verarbeiten wollen, müssen wir zuerst festhalten, was man darunter eigentlich versteht.

Ein **zeitlich diskretes Signal** ist eine doppelt unendliche Folge der Art

$$\sigma = (\sigma_j)_{j\in\mathbb{Z}} \in \ell(\mathbb{Z}).$$

Wir bezeichnen hier

$$\ell(\mathbb{Z}) = \{\sigma = (\sigma_j)_{j\in\mathbb{Z}} : \sigma_j \in \mathbb{C}\}$$

als Vektorraum aller Signale.

Realistische Signale haben normalerweise einen Anfang und ein Ende, also einen endlichen Träger und mindestens endliche Energie. Es ist jedoch viel praktischer, mit doppelt unendlichen Signalen zu arbeiten, da wir uns so weniger Gedanken über Grenzwertprobleme machen müssen, welche sehr lästig sein können.

Außerdem brauchen wir einen Filter, da wir später ein Signal mit wenigen Spitzen verarbeiten wollen, bei dem es unmöglich ist, diese über Abtasten zu treffen. Genauer gesagt wollen wir einen Tiefpassfilter, doch dazu später Genaueres. Zunächst sehr allgemein:

25 Definition.
Ein **Filter** $F : \ell(\mathbb{Z}) \to \ell(\mathbb{Z})$ ist ein Operator auf einem diskreten Signalraum.

Zu guter Letzt benötigen wir noch die Fouriertransformierte, welche uns den Anteil der entsprechenden Frequenz an einem Signal liefert sowie den vorhin bereits angesprochenen Tiefpassfilter.

26 Definition.
Für $f \in \ell_1(\mathbb{R})$ definieren wir die **Fouriertransformierte** $\hat{f}(\xi) : \mathbb{R} \to \mathbb{C}$ als

$$\hat{f}(\xi) := \int_{\mathbb{R}} f(t)e^{-i\xi t}dt, \quad \xi \in \mathbb{R}$$

und die Fouriertransformierte einer Folge $c \in \ell_1(\mathbb{Z})$ als diskretes Gegenstück

$$\hat{c}(\xi) := \sum_{k\in\mathbb{Z}} c(k)e^{-ik\xi}, \quad \xi \in \mathbb{R}.$$

27 Definition.
Ein Filter F wird **Tiefpassfilter** genannt, falls

$$\operatorname{supp} \hat{f} = [-a, a]$$

für ein $a \in (0, \pi)$

Darunter kann man sich evtl. noch nicht allzu viel vorstellen. Da wir uns allerdings nicht weiter mit Beispielen aufhalten wollen erfolgt an dieser Stelle erneut der Verweis auf [Sau20], wo einige Beispiele zu Filter-Design und Konstruktion von Tiefpassfilter zu finden sind.

Wir betrachten allerdings eine kleine grafische Veranschaulichung, um zu verstehen, was bei der Tiefpassfilterung passiert. In Abb. 1.1 haben wir ein Signal mit ein paar Peaks, bei dem es unmöglich ist, mit Abtasten (die kleinen Striche in blau) diese Peaks herauszufinden. Deswegen verwendet man eine Tiefpassfilterung die in Abb. 1.2 dargestellt ist. Bei dieser nehmen die Abtastwerte immer weiter zu, je näher man dem Peak kommt.

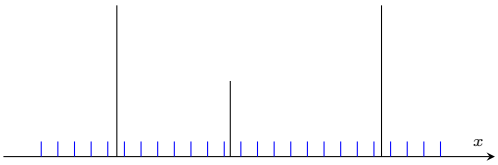

Abbildung 1.1: Signal mit wenigen Peaks

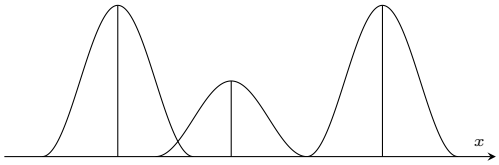

Abbildung 1.2: Signal inkl. Tiefpassfilterung

Damit haben wir alles zusammen, um uns dem eigentlichen Problem der Arbeit zu widmen.

1.6 Superresolution über Kettenbrüche

Grundmodell
Wir haben ein Signal $s : \mathbb{R} \to \mathbb{R}$, bestehend aus m Peaks:

$$s = \sum_{j=1}^{m} f_j \delta_{x_j}, \quad f_j > 0, \quad x_j \in \mathbb{R}.$$

Das Signal wird nicht direkt abgetastet, sondern erst nach einer Tiefpassfilterung, aus zuvor genanntem Grund. Dafür benötigen wir zunächst die Fourier-Transformierte

$$\hat{s}(\xi) = \int_{\mathbb{R}} s(x) e^{-i\xi x} dx = \int_{\mathbb{R}} \sum_{j=1}^{m} f_j \delta_{x_j}(x) e^{-i\xi x} dx = \sum_{j=1}^{m} f_j \int_{\mathbb{R}} e^{-i\xi x} \delta_{x_j}(x) dx$$
$$= \sum_{j=1}^{m} f_j e^{-i\xi x_j}.$$

Für $k \leq |N|$ und $h > 0$ nennen wir den Tiefpassinhalt

$$\sigma_k := \hat{s}(kh) = \sum_{j=1}^{m} f_j e^{-ix_j kh}.$$

Dabei steht N für die Anzahl verfügbarer Tiefpass-Koeffizienten.

Es gilt

$$\sigma_{-k} = \sum_{j=1}^{m} f_j e^{ix_j kh} = \sum_{j=1}^{m} \overline{f_j e^{-ix_j kh}} = \overline{\sum_{j=1}^{m} f_j e^{-ix_j kh}} = \overline{\sigma_k},$$

womit bereits die gesamte Information in $\sigma_0, ..., \sigma_N$ enthalten ist. Wir erhalten also ein Problem nach Prony's Art (siehe Beginn Kapitel 1.3). Allerdings gibt es zwei spezielle Einschränkungen:

- Die Gewichte f_j sind $\neq 0$, was später noch nützlich sein wird.
- Die Frequenzen $\omega_j = -ihx_j$ sind rein imaginär. Das ist nicht nur aus Sicht der Stabilität von Vorteil, sondern schränkt das Problem zusätzlich auf den Torus $\mathbb{T}$ ein. D.h. die Werte hx_j sind nur $\bmod 2\pi$ relevant und müssen sich in $\mathbb{T}$ voneinander unterscheiden.

Wir werden, um das Superresolution-Problem zu lösen, die Frequenzen rekonstruieren, indem wir die Konvergenten der Laurent-Reihe

$$\sigma(x) = \sum_{j=0}^{\infty} \sigma_j x^{-j}$$

ermittel, unter der Annahme, dass wir genügend Stichproben haben, damit

$$\operatorname{rang} \left(\begin{bmatrix} \sigma_o & \cdots & \sigma_{N/2} \\ \vdots & \ddots & \vdots \\ \sigma_{N/2} & \cdots & \sigma_N \end{bmatrix} \right) = m.$$

Um die Konvergenten und sogar die Kettenbrucherweiterung der Laurent-Reihe $\sigma(x)$ zu bestimmen, nutzen wir die Rekursionsrelation aus Korollar 14. Numerisch gesehen ist der Quotient von Determinanten in der Formel zu α_{n+1} am aufwendigsten. Explizit ist nach (1.15) für $n \in \mathbb{N}$

$$\alpha_{n+1} = (-1)^n \left(\prod_{j=1}^{n} \alpha_j \right)^{-2} \frac{\det(\Lambda_n)}{\det(\Lambda_{n+1})}, \quad \text{mit} \quad \Lambda_n = \begin{bmatrix} \sigma_o & \cdots & \sigma_{n-1} \\ \vdots & \ddots & \vdots \\ \sigma_{n-1} & \cdots & \sigma_{2n-2} \end{bmatrix}$$

zu bestimmen.

Trotz dass die Determinanten mit (1.19) nicht 0 sind, da $f_j > 0$, ist ihre explizite Berechnung numerisch instabil und somit nicht empfehlenswert. Um dies also zu vermeiden, stellen wir zunächst fest, dass nach dem Laplaceschen Entwicklungssatz gilt

$$\det(\Lambda_n) = \det\left(\begin{bmatrix} \sigma_o & \cdots & \sigma_{n-1} & 0 \\ \vdots & \ddots & \vdots & \vdots \\ \sigma_{n-1} & \cdots & \sigma_{2n-2} & 0 \\ \sigma_n & \cdots & \sigma_{2n-1} & 1 \end{bmatrix} \right).$$

Außerdem liefert uns die Cramersche Regel für das LGS

$$\underbrace{\begin{bmatrix} \sigma_o & \cdots & \sigma_n \\ \vdots & \ddots & \vdots \\ \sigma_n & \cdots & \sigma_{2n} \end{bmatrix}}_{=\Lambda_{n+1}} \underbrace{\begin{bmatrix} x_{n+1,1} \\ \vdots \\ x_{n+1,n+1} \end{bmatrix}}_{=:x_{n+1}} = \underbrace{\begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}}_{=e_{n+1}},$$

dass

$$x_{n+1,n+1} = \frac{\det(\Lambda_n)}{\det(\Lambda_{n+1})}.$$

Zudem gilt

$$x_{n+1,n+1} = e_{n+1}^T x_{n+1} \quad \text{sowie} \quad \Lambda_{n+1} x_{n+1} = e_{n+1} \quad \text{und damit} \quad x_{n+1} = \Lambda_{n+1}^{-1} e_{n+1}.$$

Damit erhalten wir

$$\frac{\det(\Lambda_n)}{\det(\Lambda_{n+1})} = x_{n+1,n+1} = e_{n+1}^T x_{n+1} = e_{n+1}^T \Lambda_{n+1}^{-1} e_{n+1}. \tag{1.22}$$

Die Schur-Komplement-Regel, angewandt auf (1.22), liefert schließlich:

$$\frac{\det(\Lambda_n)}{\det(\Lambda_{n+1})} = \left(\sigma_{2n} - v_n^T \underbrace{\Lambda_n^{-1} v_n}_{=:y_n} \right)^{-1}, \quad \text{wobei} \quad v_n = \begin{bmatrix} \sigma_n \\ \vdots \\ \sigma_{2n-1} \end{bmatrix}.$$

Wir lösen also das quadratische, symmetrische und positiv definite System $\Lambda_n y_n = v_n$ und bestimmen dann

$$t_{n+1} := \sigma_{2n} - v_n^T y_n \tag{1.23}$$

mit dem Sonderfall

$$t_1 = \left(e_0 \Lambda_1^{-1} e_0\right)^{-1} = \sigma_0.$$

Falls für ein $n^* \in \mathbb{N}$ gilt $t_{n^*+1} = 0$, dann ist q_{n^*+1} das Prony-Polynom. Andernfalls aktualisieren wir

$$\begin{aligned}\alpha_{n+1} &= (-1)^n \prod_{j=1}^{n} \alpha_j^{-2} \frac{\det(\Lambda_n)}{\det(\Lambda_{n+1})} = (-1)^n \prod_{j=1}^{n} \alpha_j^{-2} \left(\sigma_{2n} - v_n^T \Lambda_n^{-1} v_n\right)^{-1} \\ &= (-1)^n \prod_{j=1}^{n} \alpha_j^{-2} \left(\sigma_{2n} - v_n^T y_n\right)^{-1} = \frac{(-1)^n}{t_{n+1}} \prod_{j=1}^{n} \alpha_j^{-2}. \end{aligned} \tag{1.24}$$

Die letzte Umformung (1.24) erlaubt es uns, t_{n+1} in Abhängigkeit der α_n darzustellen:

$$t_{n+1} = (-1)^n \alpha_{n+1} \prod_{j=1}^{n+1} \alpha_j^{-2} \tag{1.25}$$

und wir erhalten damit

$$\alpha_{n+1} = \frac{(-1)^n}{t_{n+1}} \prod_{j=1}^{n} \alpha_j^{-2} = \frac{-1}{\alpha_n t_{n+1}} \left((-1)^{n-1} \alpha_n \prod_{j=1}^{n} \alpha_j^{-2} \right) = \frac{-t_n}{\alpha_n t_{n+1}}, \tag{1.26}$$

wegen (1.23) und (1.25), initialisiert mit $\alpha_1 = \dfrac{1}{\sigma_0}$. Damit haben wir die α_n ermittelt und widmen uns den β_n. Nach (1.16) erhalten wir

$$\beta_{n+1} = \alpha_{n+1} \left(\frac{\det(\Lambda'_{n+2})}{\det(\Lambda_{n+1})} + \frac{\det(\Lambda'_{n+1})}{\det(\Lambda_n)} - 2 \sum_{j=1}^{n} \frac{\beta_j}{\alpha_j} \right),$$

mit $\beta_1 = -\dfrac{\sigma_1}{\sigma_0^2}$ und $\Lambda'_n = \begin{bmatrix} \sigma_1 & \cdots & \sigma_{n-1} & 0 \\ \vdots & \ddots & \vdots & \vdots \\ \sigma_{n-2} & \cdots & \sigma_{2n-4} & 0 \\ \sigma_{n-1} & \cdots & \sigma_{2n-3} & 1 \\ \sigma_n & \cdots & \sigma_{2n-2} & 0 \end{bmatrix}$.

Dieser Ausdruck soll zunächst etwas vereinfacht werden. Im ersten Schritt wenden wir den Laplaceschen Entwicklungssatz an, indem wir $\det(\Lambda'_{n+1})$ nach der $(n+1)$-ten Spalte entwickeln. Anschließend wird die Matrix transponiert, wodurch wir in der Determinante die Matrix Λ_n erhalten, bei der die letzte Spalte durch den Vektor v_n

ersetzt wurde.

$$\frac{\det \Lambda'_{n+1}}{\det \Lambda_n} = \frac{\det \begin{bmatrix} \sigma_0 & \cdots & \sigma_{n-1} & 0 \\ \vdots & & \vdots & \vdots \\ \sigma_{n-2} & \cdots & \sigma_{2n-3} & 0 \\ \sigma_{n-1} & \cdots & \sigma_{2n-2} & 1 \\ \sigma_n & \cdots & \sigma_{2n-1} & 0 \end{bmatrix}^T}{\det \Lambda_n}$$

$$= \frac{\overbrace{(-1)^{n+n+1}}^{=-1} \cdot 1 \cdot \det \begin{bmatrix} \sigma_0 & \cdots & \sigma_{n-2} & \overbrace{\sigma_n}^{v_n} \\ \vdots & & \vdots & \vdots \\ \sigma_{n-1} & \cdots & \sigma_{2n-3} & \sigma_{2n-1} \end{bmatrix}}{\det \Lambda_n}$$

Erinnern wir uns an das LGS $\Lambda_n y_n = v_n$ welches wir betrachten, so liefert die Cramersche Regel

$$\frac{\det \Lambda'_{n+1}}{\det \Lambda_n} = -\underbrace{y_{n,n}}_{n\text{-ter Eintrag von } y_n} . \tag{1.27}$$

Mit (1.27) können wir die β_n also wie folgt ermitteln:

$$\beta_n = \alpha_{n+1} \left(-y_{n+1,n+1} - y_{n,n} - 2 \sum_{j=1}^{n} \frac{\beta_j}{\alpha_j} \right),$$

initialisiert mit $\beta_1 = -\frac{\sigma_1}{\sigma_0^2}$. Damit haben wir alle nötigen Parameter bestimmt, um das Prony-Polynom aufzustellen, dessen Nullstellen uns die gesuchten Frequenzen liefert.

Fassen wir an dieser Stelle die Erkenntnisse zusammen und stellen einen effizienten Algorithmus zur Lösung des Superresolution-Problems auf.

1. Beginne mit den initialen Parametern

   $$\alpha_1 = \frac{1}{\sigma_0}, \quad \beta_1 = -\alpha_1 \cdot x_{1,1}, \quad t_1 = \sigma_0$$

   und führe die folgenden Schritte für $n = 1, 2, ...$ aus.

2. Löse das LGS

   $$\underbrace{\begin{bmatrix} \sigma_o & \cdots & \sigma_{n-1} \\ \vdots & \ddots & \vdots \\ \sigma_{n-1} & \cdots & \sigma_{2n-2} \end{bmatrix}}_{=\Lambda_n} y_n = \underbrace{\begin{bmatrix} \sigma_n \\ \vdots \\ \sigma_{2n-1} \end{bmatrix}}_{=v_n}$$

   und setzte

   $$t_{n+1} = \sigma_{2n} - v_n^T y_n.$$

3. Falls $t_{n+1} \neq 0$, bestimme

$$\alpha_{n+1} = \frac{t_n}{-\alpha_n t_{n+1}}$$

sowie

$$\beta_{n+1} = \alpha_{n+1}\left(-x_{n+1,n+1} - x_{n,n} - 2\sum_{j=1}^{n}\frac{\beta_j}{\alpha_j}\right)$$

und fahre mit dem nächsten Wert für n fort.

4. Falls $t_{n+1} = 0$, setze

$$\beta_n = \alpha_n\left(-x_{n,n} - x_{n-1,n-1} - 2\sum_{j=1}^{n-1}\frac{\beta_j}{\alpha_j}\right)$$

und beende den Algorithmus.

Nun kann die Kettenbrucherweiterung

$$\sigma(x) = \frac{1|}{|\alpha_1 x + \beta_1} + \cdots + \frac{1|}{|\alpha_m x + \beta_m}$$

der Potenzreihe des zurückerhaltenen Signals aufgestellt werden.

28 Bemerkung.
Die Bestimmung der Kettenbrucherweiterung kann letztendlich dazu verwendet werden, Prony's Problem zu lösen:

1. Die letzte Konvergente $\frac{p_m}{q_m}$ enthält nach Satz 20 das Prony-Polynom $q_m \in \Pi_{m+1}$ im Nenner.
2. Mit (1.21) können die Rekursionskoeffizienten genutzt werden, um die tridiagonale Matrix

$$M = \begin{bmatrix} -\frac{\beta_1}{\alpha_1} & -\frac{1}{\alpha_2} & & & \\ \frac{1}{\alpha_1} & -\frac{\beta_2}{\alpha_2} & -\frac{1}{\alpha_3} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{1}{\alpha_{m-2}} & -\frac{\beta_{m-1}}{\alpha_{m-1}} & -\frac{1}{\alpha_m} \\ & & & \frac{1}{\alpha_{m-1}} & -\frac{\beta_m}{\alpha_m} \end{bmatrix} \in \mathbb{C}^{m\times m} \tag{1.28}$$

aufzustellen, deren Eigenwerte die Nullstellen des Prony-Polynoms sind und damit die zurückerworbenen Frequenzen.

Im Folgenden ist der Code zu finden, mit dem das Superresolution-Problem mithilfe von Kettenbrüchen, wie oben ausgeführt, gelöst wird. Das Programm wurde in *Octave*, Version 5.2.0 geschrieben.

Kapitel 2

Algorithmus

```
clear;

global weights; %Gewichte
global frequencies; %Frequenzen
global frequencies_mod; %Frequenzen, modulo 2*pi, mit denen
   der Algorithmus arbeiten muss
global toleranceLimit; %Da t_n niemals wirklich den Wert 0
   annimmt muessen wir eine Toleranzgrenze angeben, die t_n
   nicht ueberschreiten darf, z.B. 1e-10
global numberOfValidParameters; %Anzahl der Parameter, die
   berechnet werden koennen, solange t_n den Wert 0 nicht
   annimmt, bzw. die Toleranzgrenze nicht ueberschreitet

%setParemteres ist eine Methode fuer das zentrale Setzen der
    Parameter.
function setParameters(newWeights, newFrequencies,
   newToleranceLimit)
  if length(newWeights) != length(newFrequencies)
    error("Anz. der Gewichte und Frequenzen muss gleich sein
       !");
  elseif newToleranceLimit <= 0
    error("Valide Toleranz > 0 angeben!")
  endif
  global weights;
  weights = newWeights;
  global frequencies;
  frequencies = -i*newFrequencies;
  global frequencies_mod;
  frequencies_mod = frequencies - (2*pi) .* floor (
     frequencies ./ (2*pi)); %wir betrachten die Frequenzen
     modulo 2pi
  global toleranceLimit;
```

```
  toleranceLimit = newToleranceLimit;
  global numberOfValidParameters;
  numberOfValidParameters = 0; %zunaechst mit 0
     initialisieren
endfunction

%sigma_k erstellt in Abhaengigkeit von k den Tiefpassanteil.
    Der in der Arbeit verwendete Parameter h>0 wird hier mit
    1 belegt.
function result = sigma_k(k)
  global frequencies_mod;
  global weights;
  if k < 0
    error("Das Argument fuer sigma_k darf nicht kleiner 0
       sein!");
  else
    f = @(n) weights(n)*exp(frequencies_mod(n)*k);
    summation = 0;
    for k = 1:length(weights)
      summation = summation + f(k);
    endfor
    result = summation;
  endif
endfunction

%Lambda_n erstellt in Abhaengigkeit von n die nxn-Matrix die
    fuer die Bestimmung der Parameter alpha_n und beta_n
   benoetitgt wird.
function result = Lambda_n(n)
  if n > 0
    res = [sigma_k(0)];
    for k = 1:(n-1)
      res = [res, sigma_k(k)];
    endfor
    for k = 1:(n-1)
      appendsigma = [sigma_k(k)];
      for l = 1:(n-1)
        appendsigma = [appendsigma, sigma_k(k+l)];
      endfor
      res = [res;appendsigma];
    endfor
    result = res;
  else
    error ("Lambda_n braucht ein Argument, dass groesser als
       0 ist!");
  endif
```

```
endfunction

%v_n erstellt in Abhaengigkeit von n den n-dimensionalen
   Vektor. Mit diesem und Lambda_n wird ein LGS aufgestellt
   und geloest, um das numerisch instabile Berechnen von
   Determinanten (fuer die Bestimmung der alpha und beta) zu
    umgehen.
function result = v_n(n)
  if n > 0
    res = [sigma_k(n)];
    for k = (n+1):(2*n-1)
      res = [res; sigma_k(k)];
    endfor
    result = res;
  else
    error("v_n braucht ein Argument, dass groesser als 0 ist
       !");
  endif
endfunction

%x_n liefert in Abhaengigkeit von n die n-dimensionale
   Loesung des LGS aus Lambda_n und v_n
function result = x_n(n)
  if n > 0
    result = Lambda_n(n)\v_n(n);
  else
    error("x_n braucht ein Argument, dass groesser als 0 ist
       !");
  endif
endfunction

%t_n bestimmt in Abaengigkeit von n den Parameter t aus dem
   Algorithmus. Durch ihn koennen die Parameter alpha und
   beta wesentlich eleganter berechnet werden (wie in der
   Erklaerung zum Algorithmus ausgefuehrt).
function result = t_n(n)
  if n <= 0
    error("t_n braucht ein Argument, dass groesser als 0 ist
       !")
  elseif n == 1
    result = sigma_k(0);
  else
    result = sigma_k(2*(n-1)) - v_n(n-1).' * x_n(n-1);
  endif
endfunction
```

```
%alpha_n bestimmt in Abhaengigkeit von n die Parameter alpha
function result = alpha_n(n)
  if n <= 0
    error("alpha_n braucht ein Argument, dass groesser als 0
       ist!");
  elseif n == 1
    result = 1/sigma_k(0);
  else
    result = t_n(n-1)/(-alpha_n(n-1)*t_n(n));
  endif
endfunction

%beta_n bestimmt in Abhaengigkeit von n die Parameter beta
function result = beta_n(n)
  if n <= 0
    error("beta_n braucht ein Argument, dass groesser als 0
       ist!");
  elseif n == 1
    result = alpha_n(n)*(0 - x_n(n)(n));
  else
    sumtoadd = 0;
    for k = 1:(n-1)
      sumtoadd = sumtoadd + beta_n(k)/alpha_n(k);
    endfor
    result = alpha_n(n)*(-x_n(n)(n) - x_n(n-1)(n-1) - 2*
       sumtoadd);
  endif
endfunction

%tridiagonalM loest letzendlich das Superresolution-Problem,
    indem die Methode die tridiagonale Matrix aufstellt und
   ihre Eigenwerte bestimmt, wie in der Bemerkung nach dem
   Algorithmus ausgefuehrt.
function result = tridiagonalM()
  global weights;
  global numberOfValidParameters;
  if numberOfValidParameters < 1
    error("keine gueltigen Parameter vorhanden");
  else
    matrix = zeros(numberOfValidParameters,
       numberOfValidParameters);
    for k = 1:numberOfValidParameters %mittlere Diagonale
      matrix(k,k) = -beta_n(k)/alpha_n(k);
    endfor
    for k = 2:numberOfValidParameters %obere Diagonale
      matrix(k-1,k) = -1/alpha_n(k);
```

```
    endfor
    for k = 1:numberOfValidParameters-1 %untere Diagonale
      matrix(k+1,k) = 1/alpha_n(k);
    endfor
  endif
  result = matrix;
endfunction

%superresolution_prony ist fuer die Nutzerinteraktion
   zustaendig. Die Methode benoetigt die zwei Vektoren
   newWeights und newFrequencies gleicher Dimension, in
   denen die Gewichte und Frequenzen stehen, wie sie nach
   der Tiefpassfilterung vorhanden sind sowie eine
   Toleranzgrenze die t_n nicht ueberschreiten darf. Sie
   ruft anschliessend die Methode tridiagonalM auf, welche
   fuer die Loesung des Superresolution-Problem zustaendig
   ist und liefert als Ergebnis eine Matrix in der in der
   ersten Spalte die zurueckgewonnenen Gewichte und in der
   zweiten Spalte die zurueckgewonnenen Frequenzen stehen.
function result = superresolution_prony (newWeights,
   newFrequencies,newToleranceLimit)
  setParameters(newWeights,newFrequencies,newToleranceLimit)
     ;
  n = 1;
  global toleranceLimit;
  global numberOfValidParameters;
  numberOfValidParameters = 0;
  %Folgende Schleife bestimmt die Anzahl der gueltigen
     Parameter, d.h. die Zahl n fuer die t_n ungleich 0,
     aber t_{n+1} gleich 0 ist bzw. t_n die Toleranzgrenze
     nicht ueberschreitet.
  while abs(t_n(n)) > toleranceLimit
    numberOfValidParameters = numberOfValidParameters + 1;
    n = n+1;
  endwhile
  global weights;
  frequencyParameters = eig(tridiagonalM());
  resFrequencies = i*log(frequencyParameters);
  matrix = [];
  vector = [];
  %Folgende Schleife setzt Matrix und Vektor fuer das LGS
     zur Bestimmung der Gewichte auf.
  for k = 0:numberOfValidParameters
    matrix = [matrix; frequencyParameters.'.^k];
    vector = [vector; sigma_k(k)];
  endfor
```

```
  resWeights = matrix \ vector;
  result = [resWeights, resFrequencies];
endfunction
```

Kapitel 3

Tests auf Stabilität

In diesem abschließenden Kapitel wird der Algorithmus auf Stabilität untersucht. Dabei gehen wir auf verschiedene Kriterien ein:

- Variationen an den Gewichten
- Variationen an den Frequenzen
- Variation der Anzahl der Eingabeparameter.

Im Folgenden werden jeweils einige Versuchsreihen aufgezählt und anschließend mit ein paar Worten zusammengefasst. Angegeben sind dabei immer zuerst die Gewichte dann die Frequenzen und eventuelle Abweichungen des Ergebnisses von der Eingabe. Es wird an vielen Stellen die Schreibweise

$$xe + y = x \cdot 10^y \quad \text{sowie} \quad xe + yi = x \cdot 10^y i$$

verwendet, da diese so auch in *Octave* verwendet wird (z.B. bei der Ausgabe des Algorithmus). Bei der Betrachtung der Ergebnisse muss im Hinterkopf behalten werden, dass der Algorithmus die Ergebnisse nur $\bmod 2\pi$ genau bestimmen kann.

Die Tests wurden mit einem Algorithmus ausgeführt, der nicht bei der Zahl n abbricht, für die $t_n = 0$ gilt, sondern genau der Anzahl der Eingabeparameter entsprechend viele Parameter bestimmt. Damit sollen in den Grafiken eventuelle Abweichungen besser sichtbar werden.

Variationen an den Gewichten

- $(\underbrace{[1,1,1,1]}_{\text{Gewichte}}, \underbrace{[1,2,3,4]}_{\text{Frequenzen}})$ liefert keinerlei Abweichungen.
- Auch $([1,2,3,4],[1,2,3,4])$, $([1,10,100,1000],[1,2,3,4])$ und $([1,100,10000,1000000],[1,2,3,4])$ liefern keinerlei Abweichungen.
- $([1,1,1,1e-10],[1,2,3,4])$ liefert maximale Abweichungen von $3.6e-6$ bei den Frequenzen, welche auch bei Frequenz 4 liegt. Die anderen werden ohne Abweichungen erkannt.

So viel zu ein paar Standardtests. Viel interessanter ist jetzt allerdings, was passiert, wenn wir Gewichte gegen 0, oder ∞ laufen lassen. Das soll in den folgenden Versuchen untersucht werden.

- Lassen wir ein Gewicht gegen 0 laufen und die anderen konstant bei 1. Explizit betrachten wir $([1, 1, 2^{-n}, 1], [1, 2, 3, 4])$ mit $n \in \{1, ..., 100\}$ (wir betrachten allerdings erst Werte für Gewichte $\leq 2^{-45}$, da es erst dort interessant wird). Plotten wir die zurückerhaltenen Frequenzvektoren gegen das Gewicht, welches gegen 0 geht erhalten wir folgende Grafik:

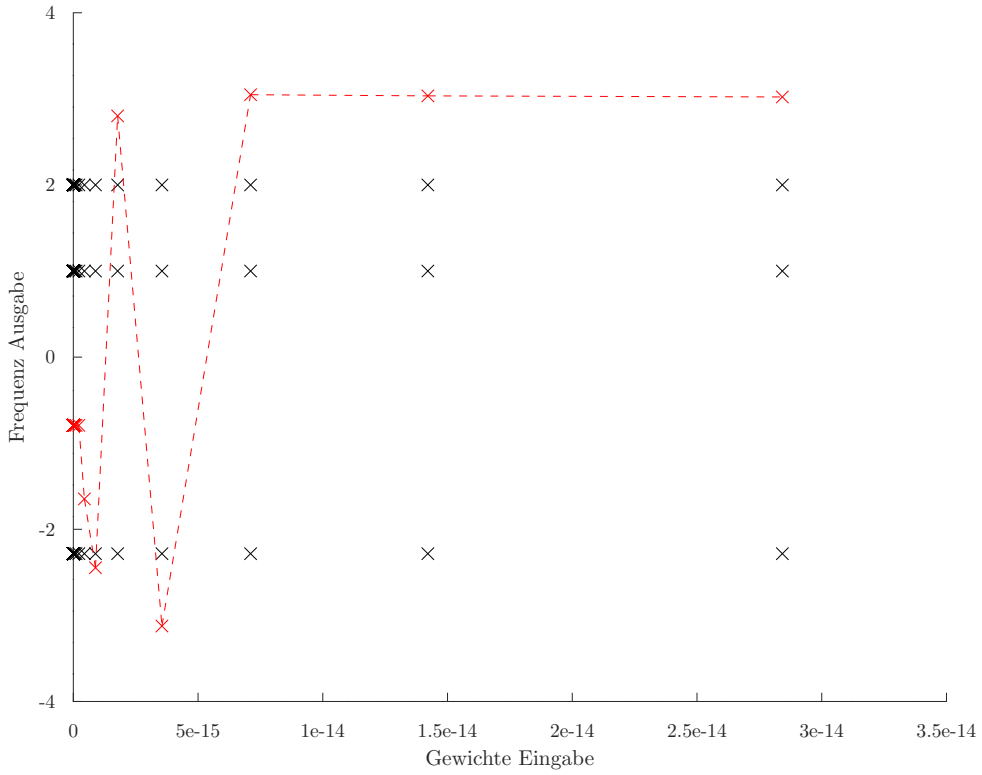

Abbildung 3.1: ein Gewicht läuft gegen 0

In dieser Abbildung wurden die Werte für Frequenz 3 nur für die bessere Sichtbarkeit gestrichelt verbunden (was also nicht den Eindruck einer stetigen Auswertung erwecken soll, da wir nur punktweise an 2^{-n} auswerten). Zu erkennen ist, dass der Algorithmus erst ab Gewichtgrößen von unter ca. 2^{-47} Probleme hat, die entsprechende Frequenz zu bestimmen. Alle anderen Frequenzen, welche mit 1 gewichtet sind bleiben davon unbeeinflusst.

- $([1e-10, 1e-10, 1e-10, 1e-10], [1, 2, 3, 4])$ liefert keinerlei Abweichungen bei den Frequenzen.
- Lassen wir nun alle Gewichte gegen 0 laufen. Explizit betrachten wir $([2^{-n}, 2^{-n}, 2^{-n}, 2^{-n}], [1, 2, 3, 4])$ mit $n \in \{1, ..., 100\}$. Beim Plotten der Fre-

quenzvektoren gegen das Gewicht lassen sich ebenfalls keine Abweichungen beobachten. In der folgenden Grafik wurden erst Gewichte unter 2^{-80} betrachtet. Größere Gewichte liefern die gleichen Ergebnisse.

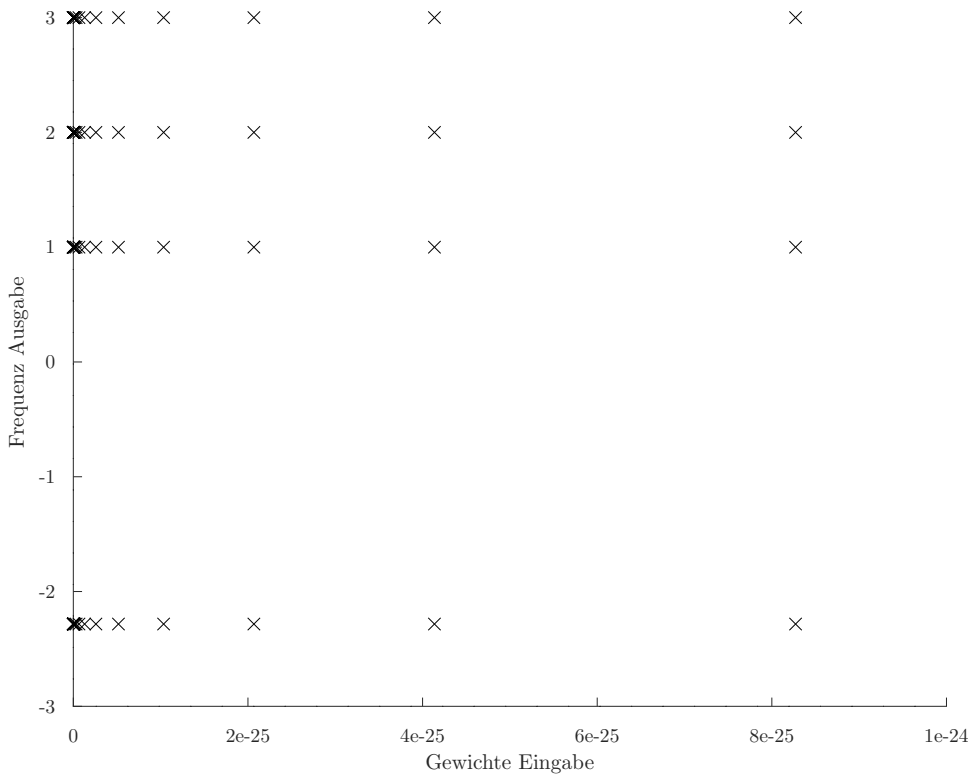

Abbildung 3.2: alle Gewichte laufen gegen 0

- $([1e10, 1e10, 1e10, 1e10], [1, 2, 3, 4])$ liefert keinerlei Abweichungen.
- Lassen wir jetzt alle Gewichte gegen ∞ laufen. Explizit betrachten wir $([2^n, 2^n, 2^n, 2^n], [1, 2, 3, 4])$ mit $n \in \{1, ..., 100\}$. Beim Plotten der Frequenzvektoren gegen das Gewicht lassen sich ebenfalls keine Abweichungen beobachten. Die Grafik dazu sieht sehr ähnlich aus wie Abbildung 3.2, wird deswegen an dieser Stelle ausgelassen.
- Lassen wir nur ein Gewicht gegen ∞ laufen und die anderen bei 1, d.h. betrachten wir explizit $([1, 1, 2^n, 1], [1, 2, 3, 4])$ mit $n \in \{1, ..., 55\}$ (für größere Werte lassen sich im Algorithmus die Eigenwerte der Matrix (1.28) nicht mehr bestimmen, da sie laut *Octave* „Inf"-, oder „NaN"-Werte enthält). In der folgenden Grafik ist der Abschnitt von 2^{41} bis 2^{52} dargestellt.

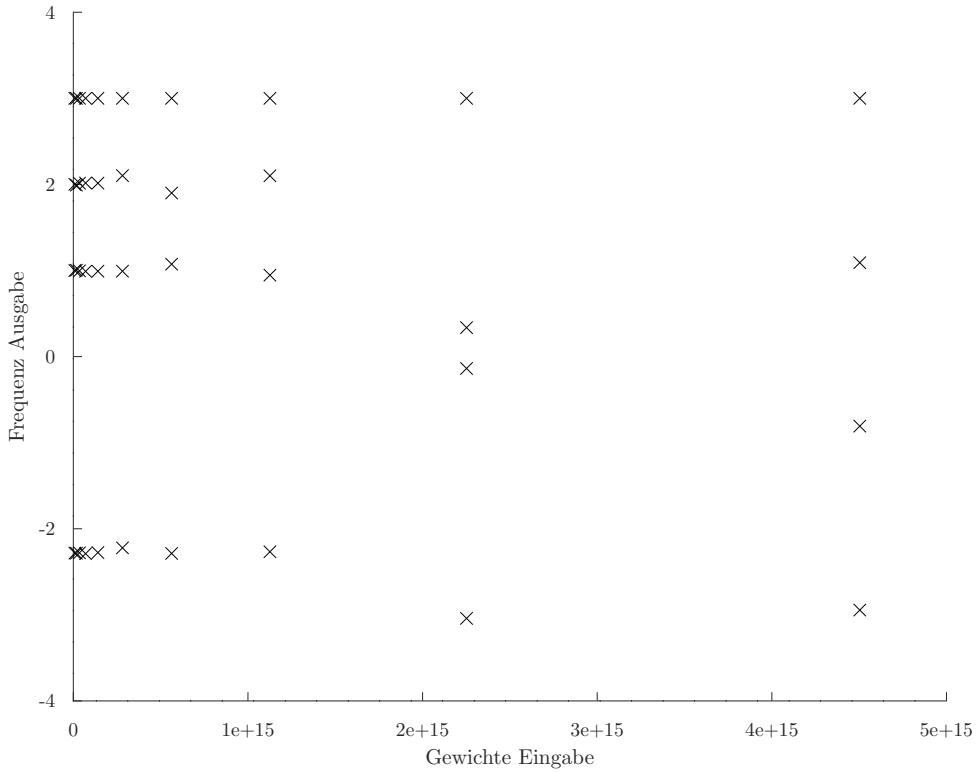

Abbildung 3.3: ein Gewicht läuft gegen ∞

Zu sehen ist, dass der Algorithmus nur die Frequenz 3 durchweg stabil bestimmen kann. Die anderen Frequenzen besitzen im Verhältnis dazu zu kleine Gewichte (vgl. Versuch bei dem wir ein Gewicht gegen 0 haben laufen lassen). Ab Gewichten ab 2^{48} sind deutlich sichtbare Abweichungen zu beobachten. Vor allem ab Gewichten über 2^{50} stimmen die Ergebnisse dann überhaupt nicht mehr mit der Eingabe überein. Ähnliche Ergebnisse erhalten wir auch, wenn wir mehrere Gewichte (z.B. drei) gegen 0 laufen lassen. Entsprechend wird nur noch die Frequenz stabil erkannt, dessen Gewicht nicht gegen 0 läuft.

Zusammenfassend lässt sich über diese Versuchsreihe sagen, dass der Algorithmus vor allem dann Schwierigkeiten hat, wenn die Gewichte sehr weit auseinander liegen. Sehr große, oder betragsmäßig sehr kleine Gewichte bereiten keine Probleme, solange sie alle ähnlich groß, oder klein sind.

Variationen an den Frequenzen

Interessant ist hierbei vor allem, was passiert, wenn Frequenzen gegeneinander laufen.

- Lassen wir zunächst zwei Frequenzen gegeneinander laufen. Explizit betrachten wir $([1, 1], [2, 2 + 2^{-n}])$ mit $n \in \{1, ..., 26\}$ (für größere Werte lassen sich wieder, wie vorhin beschrieben, die Eigenwerte der Matrix nicht mehr bestim-

men). In der folgenden Grafik wurden die erhaltenen Frequenzvektoren gegen die Differenzen zwischen den eingegebenen Frequenzen geplottet. Zu sehen sind erst die Ergebnisse für Differenzen $\leq 2^{-19}$, da es dort erst interessant wird.

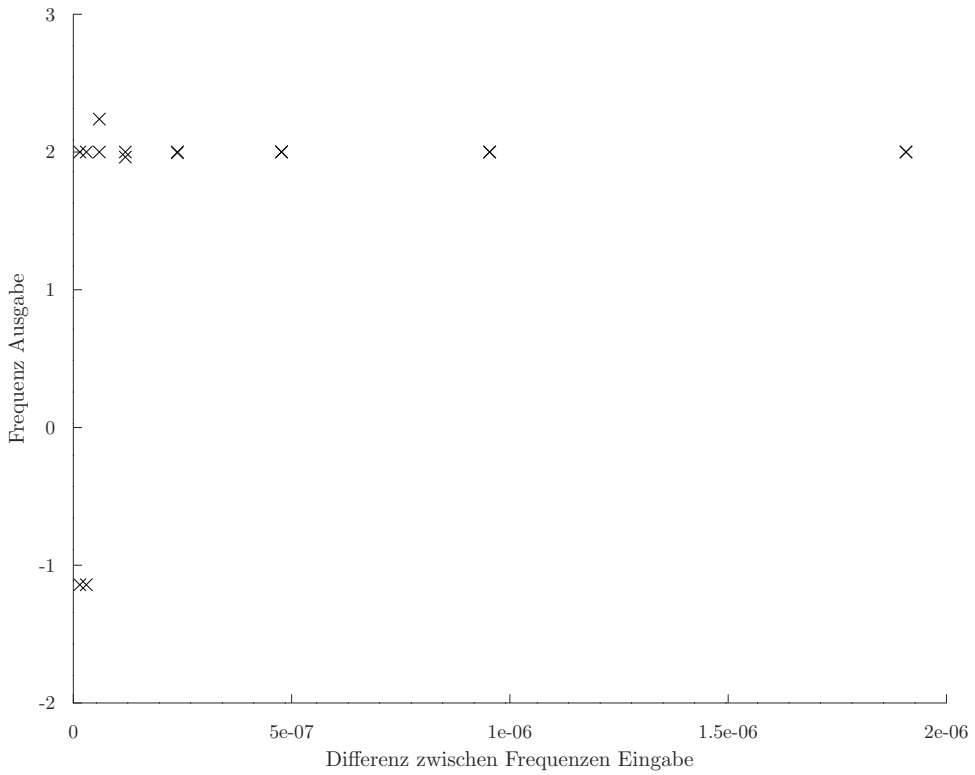

Abbildung 3.4: zwei Frequenzen laufen gegeneinander

Wir können beobachten, dass ab einer Differenz von 2^{-23} bereits sichtbare Abweichungen und ab einer Differenz von 2^{-25} sehr deutliche Abweichungen von der Eingabe auftreten.

- Lassen wir jetzt drei Frequenzen gegeneinander laufen. Betrachten wir $([1,1,1],[2,2+2^{-n},2-2^{-n}])$ mit $n \in \{1,...,26\}$ (auch hier lassen sich für größere Werte die Eigenwerte der Matrix nicht mehr bestimmen) sowie die Grafik der geplotteten Frequenzen in der die Ergebnisse ab einer Differenz $\leq 2^{-10}$ zu sehen sind.

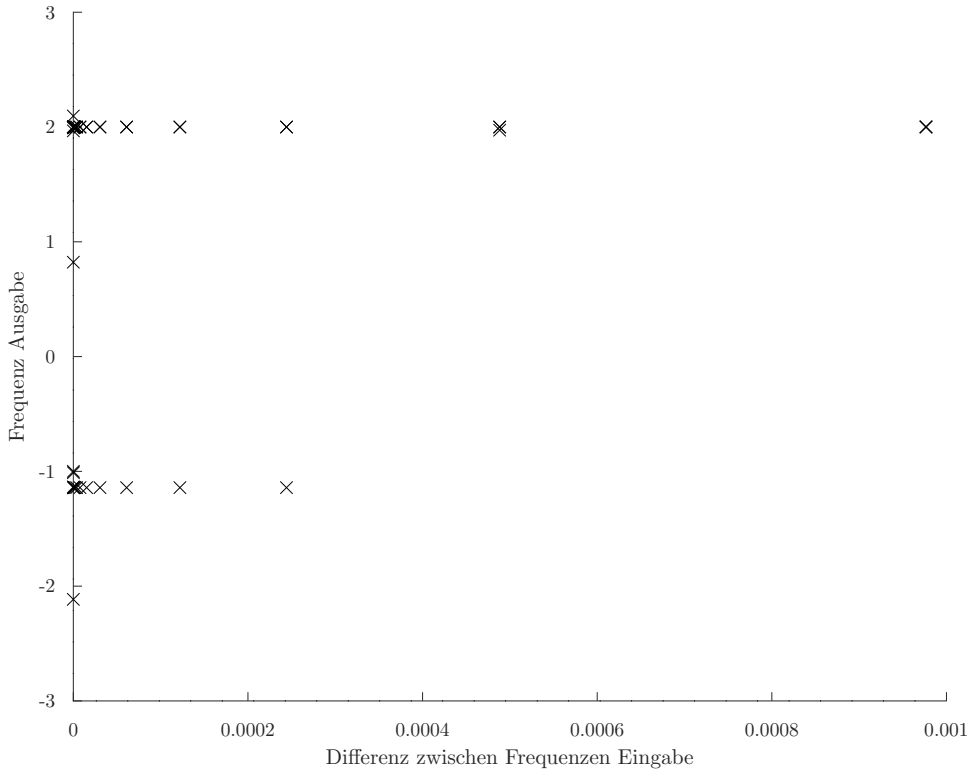

Abbildung 3.5: drei Frequenzen laufen gegeneinander

Bereits bei einer Differenz von 2^{-12} liefert der Algorithmus nur noch zwei Frequenzen stabil, eine weicht bereits sehr stark von der Eingabe ab. Ab einer Differenz von höchstens 2^{-26} weicht dann auch eine zweite Frequenz sehr stark von der Eingabe ab, sodass nur noch eine Frequenz stabil ermittelt werden kann.

- Lassen wir noch mehr Frequenzen zusammenlaufen, betrachten wir zum Beispiel $([1,1,1,1,1,1,1], [1,2,2+2\cdot 2^{-n}, 2+2^{-n}, 2-2^{-n}, 2-2\cdot 2^{-n}, 3])$ im Bereich $n \in \{5,...,15\}$, ergibt sich ein noch schlechteres Bild. Bereits ab einer Differenz von höchstens 2^{-6} erkennt der Algorithmus eine Frequenz nicht mehr. Ab einer Differenz von höchstens 2^{-8} weist bereits eine zweite Frequenz sichtbare Abweichungen von der ursprünglichen Eingabe auf. Davon unberührt bleiben hingegen die beiden Frequenzen 1 und 3, welche durchgehend stabil erkannt werden. Die folgende Grafik visualisiert den oben genannten Abschnitt $n \in \{5,...,15\}$.

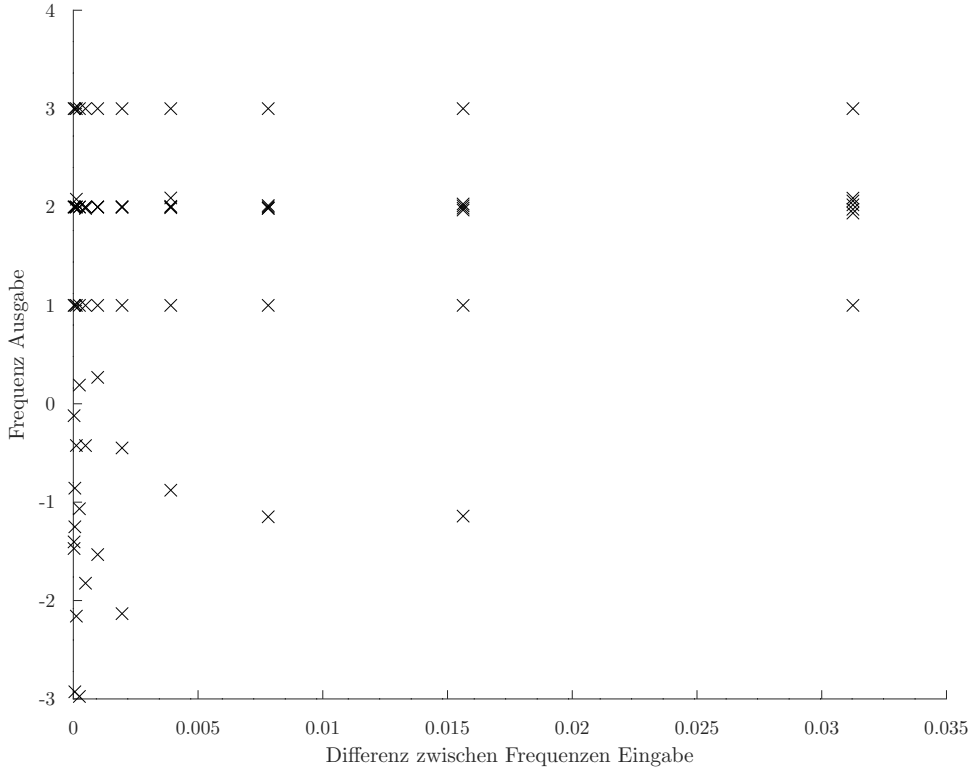

Abbildung 3.6: fünf Frequenzen laufen gegeneinander

Diese Versuchsreihe zeigt auf, dass der Algorithmus desto mehr Probleme bekommt, je mehr Frequenzen eng zusammen liegen. Solange z.B. nur zwei Frequenzen davon betroffen sind, erkennt der Algorithmus beide noch mit betragsmäßig sehr niedriger Differenz. Je mehr Frequenzen allerdings betroffen sind, desto größer müssen die Differenzen zwischen ihnen sein, damit der Algorithmus auch alle stabil ermitteln kann.

Variation der Anzahl der Eingabeparameter

Hierbei ist lediglich festzustellen, dass der Algorithmus mit zunehmender Anzahl an Gewichten und Frequenzen auch mehr Rechenzeit benötigt, was mit der Anzahl der Parameter, die bestimmt werden müssen, zusammenhängt. Außerdem sollte natürlich darauf geachtet werden, dass bei erhöhter Anzahl von Eingabeparametern die Frequenzen modulo 2π unterschiedlich sind und nicht zu nahe beieinander liegen.

Fazit

Fassen wir die Erkenntnisse aus den drei Versuchsreihen zusammen: Solange die Gewichte annähernd gleich groß sind und die Frequenzen nicht zu nahe beieinander liegen, bzw. nicht zu viele Frequenzen zu ähnlich sind liefert der Algorithmus stabile Ergebnisse und bietet damit eine erwähnenswerte Alternative zu den klassischen Methoden.

Literaturverzeichnis

[Sau20] Sauer. *Continued Fractions and Signal Processing*. Springer Nature, 1 edition, 2020.