Fakultät für Informatik und Mathematik
Universität Passau

Prony Methode zur Klanganalyse

Bachelorarbeit

Johannes Müller
Matrikel-Nummer 67232

Betreuer Prof. Dr. Tomas Sauer

Inhaltsverzeichnis

1 Einleitung 1

2 Grundlagen 3
- 2.1 Matrix-Vektor-Form . . . 3
- 2.2 Hankel Matrix . . . 4
- 2.3 Companion Matrix . . . 5
- 2.4 Vandermonde Matrix . . . 5

3 Klassische Prony-Methode 7
- 3.1 Bestimmung von f_j . . . 8
- 3.2 Bestimmung der c_j . . . 10

4 Erweiterte Prony-Methode 12

5 Implementierung 15
- 5.1 Klassische Prony-Methode . . . 15
- 5.2 Erweiterte Prony-Methode . . . 16
- 5.3 Anlegen der Vergleichsdaten . . . 17
- 5.4 Identifikation eines Klaviertons . . . 17
- 5.5 Auswertung der Ergebnisse . . . 17

6 Zusammenfassung 22
- 6.1 Klassische Prony Methode . . . 22
- 6.2 Erweiterte Prony Methode . . . 22
- 6.3 Implementierung . . . 23

A Anhang 24

Literaturverzeichnis 33

1 Einleitung

Vorliegende Arbeit beschäftigt sich mit der Rekonstruktion der Teiltöne eines von einem Klavier erzeugten Klangs. Unter Klang werden in der Akustik und Physik verschiedene Phänomene verstanden. Das, was in der Physik ein Klang ist, wird in der Akustik als Ton bezeichnet. Wird auf einem Klavier beispielsweise ein c gespielt, würde man dieses c in der Akustik als Klavierton bezeichnen, während es in der Physik ein Klang ist. In dieser Arbeit wird der physikalische Begriff Klang verwendet. In der Physik ist ein Ton eine durch eine Sinus- oder Kosinusfunktion beschreibbare Schallwelle. In der Natur kommen einzelne Töne nicht vor, sondern nur Überlagerungen von Tönen. Ist diese Überlagerung harmonisch, das heißt, dass die Frequenzen der einzelnen Töne ein ganzes Vielfaches ihres Grundtons sind, ist von einem Klang die Rede. Ist diese Überlagerung nicht harmonisch spricht man von einem Geräusch. Ein Klang setzt sich also aus mehreren Tönen zusammen. Der Ton mit der tiefsten Frequenz ist der Grundton und alle anderen Töne bezeichnet man als Obertöne[1].
Ziel dieser Arbeit ist es, mithilfe von Messproben eines Klanges dessen Teiltöne zu rekonstruieren, um damit Aussagen über diesen treffen zu können. Klänge unterscheiden sich nicht nur in der Tonhöhe, welche durch den Grundton bestimmt wird, sondern auch in ihrer Klangfarbe. Die Klangfarbe variiert je nach Anzahl der Obertöne eines Klangs. Besitzt ein Klang viele Obertöne, wird er als farbiger, bzw. voller wahrgenommen, während hingegen ein Klang mit nur wenigen Obertönen ein sehr klares oder reines Klangbild aufweist. Die Klangfarbe ist charakteristisch für Instrumente. Sie unterscheidet sich nicht nur bei verschiedenen Instrumentenarten, wie Violine oder Klavier, sondern auch, wenn diese gleicher Gattung sind. Deshalb ist es möglich, nicht nur die Tonhöhe eines Klangs zu bestimmen, sondern auch herauszufinden von welchem Klavier dieser gespielt wurde, falls entsprechende Vergleichsdaten vorliegen. Somit ist es möglich mithilfe der Teiltöne einen Klang eindeutig zu identifizieren. Man kann also behaupten, dass die Teiltöne eines Klangs für diesen so einzigartig sind, wie für uns Menschen der Fingerabdruck[2].

Ein Klang, also die Summe mehrerer Teiltöne, kann durch folgende Exponentialsumme beschrieben werden:

$$h(k) = \sum_{j=1}^{M} c_j e^{f_j k}$$

mit unbekannten Parametern $M \in \mathbb{N}$, $f_j \in [-\alpha, 0] + i[-\pi, \pi)$ mit $\alpha > 0$ und $c_j \in \mathbb{C}$ mit $c_j \neq 0$ dabei ist $k = 1, ..., 2M - 1$. Die c_j beschreiben die Amplitude eines Teiltones und die f_j die Abklingrate so wie Frequenz. Eine Methode zur Rekonstruktion dieser Unbekannten, genannt "Prony Methode", wurde von Gaspard Riche de Prony 1795 entwickelt[3].
Im Verlauf dieser Arbeit wird die klassische Prony Methode beschrieben, diese erweitert und die Ergebnisse, welche mit einer Implementierung in Matlab erhalten wurden, werden ausgewertet.

2 Grundlagen

Bei der Durchführung der Prony Methode werden hauptsächlich lineare Gleichungssysteme gelöst. In dem folgenden Kapitel werden die zum Verständnis nötigen mathematischen Begriffe erklärt.

2.1 Matrix-Vektor-Form

Mit der Matrix-Vektor-Form werden lineare Gleichungssysteme nur durch Verwendung von Matrizen und Vektoren beschrieben.

Für gewöhnlich werden Gleichungssysteme folgendermaßen dargestellt:

$$
\begin{array}{llll}
3x_1 & +x_2 & -3x_3 & = 3 \\
x_1 & -2x_2 & +4x_3 & = 5 \\
x_1 & -2x_2 & +4x_3 & = 5
\end{array}
$$

In Matrix-Vektor-Form wird das System durch das Produkt einer Matrix mit allen Koeffizienten und einem Vektor mit allen Unbekannten beschrieben:

$$
\begin{pmatrix} 3 & 1 & 3 \\ 1 & -2 & 4 \\ 1 & -2 & 4 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 3 \\ 5 \\ 5 \end{pmatrix}
$$

Führt man hier die Matrixmultiplikation durch, so erhält man das Gleichungssystem in gewohnter Darstellung:

$$\begin{pmatrix} 3x_1 & 1x_2 & 3x_3 \\ 1x_1 & -2x_2 & 4x_3 \\ 1x_1 & -2x_2 & 4x_3 \end{pmatrix} = \begin{pmatrix} 3 \\ 5 \\ 5 \end{pmatrix}$$

Im weiteren Verlauf der Arbeit werden lineare Gleichungssysteme in Matrix-Vektor-Form dargestellt[4].

2.2 Hankel Matrix

Unter einer Hankel Matrix (benannt nach Hermann Hankel) versteht man eine Matrix, in der alle Einträge, auf einer Diagonalen von rechts oben nach links unten verlaufend, innerhalb der Matrix den selben Wert besitzen.

Ein einfaches Beispiel:

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 \\ 4 & 5 & 6 & 7 & 8 \\ 5 & 6 & 7 & 8 & 9 \end{pmatrix}$$

Es reichen die oberste Zeile und die Spalte rechts außen aus, um eine solche Matrix zu beschreiben[5].

2.3 Companion Matrix

Sei ein Polynom der Form

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} + x^n$$

gegeben. Wichtig hierbei ist, dass der letzte Koeffizient $a_n = 1$ ist. Die rechte Spalte der zugehörigen Companion Matrix beinhaltet die Gegenzahlen der Koeffizienten $a_0, \cdots, a_{n-1}$ des Polynoms $p(x)$. Die Subdiagonale der Matrix hat konstant den Wert 1. Alle anderen Einträge der Matrix sind 0.

Die zum Polynom $p(x)$ gehörende Companion Matrix $C(p)$ hat also die Form:

$$C(p) = \begin{pmatrix} 0 & 0 & \cdots & 0 & -a_0 \\ 1 & 0 & \cdots & 0 & -a_1 \\ 0 & 1 & \cdots & 0 & -a_3 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & -a_{n-1} \end{pmatrix}$$

Eine für die Prony Methode wichtige Eigenschaft der Companion Matrix ist, dass die Eigenwerte der Matrix den Nullstellen des zugehörigen Polynoms $p(x)$ entsprechen[6][3].

2.4 Vandermonde Matrix

Eine Vandermonde Matrix, benannt nach Alexandre-Théophile Vandermonde, wird durch ein n-Tupel reeller Zahlen $(x_1, x_2, \cdots, x_n)$, auch Stützstellen genannt, definiert. Die Matrix besitzt die Werte $V_{i,j} = x_i^{j-1}$, wobei i Index der Zeile und j Index der Spalte ist.

Eine Vandermonde Matrix mit Stützstellen $(x_1, \cdots, x_n)$ hat also die Form:

$$V(x_1, x_2, \cdots, x_n) = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ 1 & x_3 & x_3^2 & \cdots & x_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{pmatrix}$$

Für die Prony Methode interessant ist die Anwendung der Vandermonde Matrix bei Interpolationsproblemen. Wenn man nur wenige exakte Werte einer Funktion besitzt, wird versucht sich den fehlenden Werten anzunähern. Hat man beispielsweise durch eine physikalische Messung Werte $f_1, ..., f_n$ erhalten und möchte Auskunft über die fehlenden Werte, so können diese über ein Polynom

$$p(x) = a_0 + a_1 x^1 + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$$

approximiert werden. Die Koeffizienten des Polynoms können durch Lösen des Gleichungssystems

$$\begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ 1 & x_3 & x_3^2 & \cdots & x_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ \vdots \\ \vdots \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} f_1 \\ \vdots \\ \vdots \\ \vdots \\ f_n \end{pmatrix}$$

bestimmt werden[7][3].

3 Klassische Prony-Methode

Seien Werte $h(k)$ für $k = 0, ..., 2M-1$ einer Exponentialsumme mit $M \in \mathbb{N}$

$$h(k) = \sum_{j=1}^{M} c_j e^{f_j k}$$

gegeben. Diese Werte können zum Beispiel aus physikalischen Messungen gewonnen worden sein. Ziel der Prony Methode ist es, alle fehlenden Parameter der Exponentialsumme zu bestimmen[3].

Diese sind:

- positiver Integer M, der die Anzahl der Summanden beschreibt
- $f_j \in [-\alpha, 0] + i[-\pi, \pi)$ mit $\alpha > 0$ und $j = 1, ..., M$
- $c_j \in \mathbb{C}$ mit $c_j \neq 0$ und $j = 1, ..., M$

Im Rahmen dieser Arbeit wird die Prony Methode auf Klaviertöne angewandt. Ein von einem Klavier erzeugter Klang besteht aus mehreren Teiltönen. Diese Teiltöne werden jeweils durch die Parameter c_j und f_j beschrieben. Durch die Anwendung der Prony Methode auf einen Klang sollen dessen Teiltöne rekonstruiert werden, um so eine Aussage über die Tonhöhe des Klangs und das zugehörige Klavier treffen zu können.

Für die Anwendung der klassischen Prony Methode, die in diesem Kapitel beschrieben wird, wird für M eine obere Schranke benutzt, die in der Regel bekannt ist. In der Praxis kann es durchaus der Fall sein, dass beispielsweise ein Klang aus weniger als M Teiltönen besteht, aber nicht aus mehr als M.

Prony's Idee baut hauptsächlich auf der Trennung der unbekannten Exponenten f_j von den unbekannten Koeffizienten c_j auf[3].

3.1 Bestimmung von f_j

Wir definieren:

- $z_j := e^{f_j}$
- $p(z) := \prod\limits_{j=1}^{M} (z - z_j)$ (Prony Polynom) mit $z \in \mathbb{C}$

Somit sind z_j die Nullstellen des Prony Polynoms. Wendet man den natürlichen Logarithmus auf die Nullstellen an, erhält man das gesuchte f_j [3].

$$f_j = \log(z_j) \tag{3.1}$$

Teilziel ist es also zunächst, die z_j zu bestimmen. Dafür wird das Prony Polynom in folgende Form gebracht:

$$p(z) = \prod_{j=1}^{M}(z - z_j) = \sum_{k=0}^{M-1} p_k z^k + z^M$$

Die zur Berechnung der Nullstellen benötigten Koeffizienten p_k des Polynoms sind allerdings noch unbekannt. Es wird also nach einem Gleichungssystem gesucht, mit Hilfe dessen diese bestimmt werden können.
Es ist bekannt, dass der letzte Koeffizient des Prony Polynoms gleich 1 ist. Also wird $p_M := 1$ gesetzt. Für alle $m \in \mathbb{N}_0$ ist

$$\sum_{k=0}^{M} p_k h(k+m) = \sum_{k=0}^{M} pk \left(\sum_{j=1}^{M} c_j z_j^{k+m} \right) = \sum_{j=1}^{M} c_j z_j^m \left(\sum_{k=0}^{M} p_k z_j^k \right) = \sum_{j=1}^{M} c_j z_j^m p(z_j) = 0 \tag{3.2}$$

erfüllt, da $\sum_{k=0}^{M} p_k z_j^k$ immer 0 sein muss. Es handelt sich hierbei um das Prony Polynom, in das ausschließlich seine Nullstellen eingesetzt werden. Da es ein Faktor des Produkts

$c_j z_j^M p(z_j)$ ist, ist jeder Summand der Summe $\sum_{j=1}^{M} c_j z_j^M p(z_j)$ gleich 0 und somit die gesamte Summe. Mit dem in 3.2 erlangten Wissen, dass

$$\sum_{k=0}^{M} p_k h(k+m) = 0 \tag{3.3}$$

gilt, kann jetzt unter Verwendung der Messwerte für $h(k)$ mit $k = 0, 1, ..., 2M-1$ und unter Berücksichtigung, dass $p_M = 1$ ist

$$\sum_{k=0}^{M-1} p_k h(k+m) = -h(M+m) \tag{3.4}$$

für alle $m = 0, 1, ..., M-1$ gefolgert werden. In Matrix-Vektor Form ausgeschrieben erhält man somit folgendes lineares Gleichungssystem mit einer Hankel Matrix:

$$\begin{pmatrix} h(0) & h(1) & \cdots & h(M-1) \\ h(1) & h(2) & \cdots & h(M) \\ \vdots & \vdots & \vdots & \vdots \\ h(M-1) & h(M) & \cdots & h(2M-2) \end{pmatrix} \cdot \begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_M - 1 \end{pmatrix} = - \begin{pmatrix} h(M) \\ h(M+1) \\ \vdots \\ h(M+M-1) \end{pmatrix} \tag{3.5}$$

m wird hierbei zeilenweise erhöht, in Zeile 1 ist $m = 0$ und in der letzten Zeile ist $m = M-1$. Da alle Werte $h(k)$ mit $k = 0, ..., 2M-1$ bekannt sind, kann man das Gleichungssystem nach den Koeffizienten $p_0, ..., p_M - 1$ auflösen[3].

Nachdem die Koeffizienten berechnet wurden, ist der nächste Schritt die Berechnung der Nullstellen des Prony Polynoms. Hierfür wird zunächst die zum Polynom gehörige Companion Matrix $C_M(p)$ gebildet.

$$C_M(p) = \begin{pmatrix} 0 & 0 & \cdots & 0 & -p_0 \\ 1 & 0 & \cdots & 0 & -p_1 \\ 0 & 1 & \cdots & 0 & -p_3 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & -p_{M-1} \end{pmatrix}$$

Um die Nullstellen z_j des Prony Polynoms zu erhalten, wird die Gegebenheit genutzt, dass sie gleich den Eigenwerten der Companion Matrix sind. Also müssen lediglich diese berechnet werden. Nach dem Erhalt der Eigenwerte und somit der gesuchten Nullstellen z_j

können, wie in 3.1 gezeigt, die Werte der f_j durch Anwendung des natürlichen Logarithmus auf den z_j berechnet werden[3].

3.2 Bestimmung der c_j

Die Bestimmung der Werte der c_j ist vergleichsweise einfach. Sie wird durch Interpolation mithilfe einer an der Hauptdiagonalen gespiegelten Vandermonde Matrix bewerkstelligt. Da die Werte $z_j = e^{f_j}$ bereits durch die Bestimmung von f_j bekannt sind, werden sie mit den passenden Werten von $h(k)$, welche aus Messungen gegeben sind, als Stützstellen benutzt.
Um alle c_j zu erhalten, bilden wir also folgendes Gleichungssystem mithilfe der Exponentialfunktion $h(k) = \sum_{j=1}^{M} c_j z_j^k$:

$$\begin{array}{ccc} h(0) & = & c_1 + c_2 + \cdots + c_M \\ h(1) & = & c_1 z_1 + c_2 z_2 + \cdots + c_M z_M \\ h(2) & = & c_1 z_1^2 + c_2 z_2^2 + \cdots + c_M z_M^2 \\ \vdots & \vdots & \vdots \\ h(M-1) & = & c_1 z_1^{M-1} + c_2 z_2^{M-1} + \cdots + c_M z_M^{M-1} \end{array} \tag{3.6}$$

Setzt man die bekannten Werte für $h(k)$ und z_j ein, so lässt sich dieses Gleichungssystem nach den fehlenden c_j auflösen. In Matrix Vektor Form hat das Gleichungssystem folgende Form:

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ z_1 & z_2 & z_3 & \cdots & z_M \\ z_1^2 & z_2^2 & z_3^2 & \cdots & z_M^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ z_1^{M-1} & z_2^{M-1} & z_3^{M-1} & \cdots & z_M^{M-1} \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_M \end{pmatrix} = \begin{pmatrix} h(0) \\ h(1) \\ h(3) \\ \vdots \\ h(M-1) \end{pmatrix} \tag{3.7}$$

Es ist also zu erkennen, dass es sich bei dem ersten Faktor der Matrixmultiplikation um eine an der Hauptdiagonale gespiegelte Vandermonde Matrix handelt. Die Vorgehensweise zum Bestimmen der z_j ist folglich der Polynominterpolation sehr ähnlich. Nachdem man obiges Gleichungssystem gelöst hat, hat man mit c_j die letzten gesuchten Parameter der Exponentialfunktion $h(k) = \sum_{j=1}^{M} c_j e^{f_j k}$ erhalten[3].

4 Erweiterte Prony-Methode

Anders als bei der klassischen Prony-Methode ist bei der Erweiterung M nicht bekannt. Da M bisher sowieso nur eine obere Schranke war, können mit der in diesem Kapitel beschriebenen Variante der Prony Methode genauere Werte für M ermittelt werden. Der eigentliche Algorithmus ändert sich allerdings erst beim Errechnen der Koeffizienten des Prony Polynoms. Diese wurden vorher durch Lösen von 3.5 ermittelt. Da wir für M jetzt keinen fixen Wert mehr besitzen, wird nicht von vornherein eine $M \times M$ Matrix verwendet und werden nicht zwangsweise M Koeffizienten errechnet. Besteht ein Klang beispielsweise aus weniger als M Subtönen, so ist die Ermittlung von M Koeffizienten nicht nur unnötiger Rechenaufwand, sondern verfälscht auch das Ergebnis. Die Hauptidee hierbei ist, dass die in 3.5 verwendete Hankel-Matrix spaltenweise aufgebaut wird, bis eine Lösung des Gleichungssystems gefunden werden kann. Man erhält also keine $M \times M$ Matrix, sondern eine $M \times n$ Matrix und nur noch n Koeffizienten p_k mit $n \leq M$. Da M allerdings unbekannt ist und wir es für den Algorithmus dennoch benötigen, wird es folgendermaßen abgeschätzt:

Sei $H \in \mathbb{N}$ die Anzahl aller gegebenen $h(k)$, in unserem Fall die Anzahl aller Messproben aus einem Klang, dann gilt

$$M = \lfloor (H+1)/2 \rfloor . \tag{4.1}$$

Das Prony-Polynom hat jetzt die Form:

$$p(z) = \prod_{j=1}^{n} (z - z_j) = \sum_{k=0}^{n-1} p_k z^k + z^n \tag{4.2}$$

Die zum Berechnen der Koeffizienten nötige Hankel-Matrix wird folgendermaßen aufgebaut: Zu Beginn besitzt die Hankel-Matrix eine Spalte und es wird versucht, eine Lösung für

$$\begin{pmatrix} h(0) \\ h(1) \\ \vdots \\ h(M-1) \end{pmatrix} \cdot \begin{pmatrix} p_0 \end{pmatrix} = - \begin{pmatrix} h(1) \\ h(2) \\ \vdots \\ h(M) \end{pmatrix} \tag{4.3}$$

zu finden. Wurde keine passende Lösung gefunden, wird die nächste Spalte angefügt. Dieses Prozedere wird so lange wiederholt, bis sich eine Lösung ergibt. Der Lösungsvektor ist jeweils der als nächstes anzufügende Vektor. Der zweite Schritt würde also so aussehen:

$$\begin{pmatrix} h(0) & h(1) \\ h(1) & h(2) \\ \vdots & \vdots \\ h(M-1) & h(M) \end{pmatrix} \cdot \begin{pmatrix} p_0 \\ p_1 \end{pmatrix} = - \begin{pmatrix} h(2) \\ h(3) \\ \vdots \\ h(M+1) \end{pmatrix} \tag{4.4}$$

Der n-te Schritt hat demnach die Form:

$$\begin{pmatrix} h(0) & \cdots & h(n-1) \\ h(1) & \cdots & h(n) \\ \vdots & \ddots & \vdots \\ h(M-1) & \cdots & h(M+n-2) \end{pmatrix} \cdot \begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_{n-1} \end{pmatrix} = - \begin{pmatrix} h(n) \\ h(n+1) \\ \vdots \\ h(M+n-1) \end{pmatrix} \tag{4.5}$$

Wird eine Lösung gefunden, so sind die Koeffizienten des Prony-Polynoms $(p_0, p_1, ..., p_{n-1}, 1)$. Außerdem ist jetzt bekannt, dass $M = n$ sein muss, da hier das Gleichungssystem erstmals lösbar wurde und damit bekannt ist, dass der Klang aus n Teiltönen besteht.

Wurden die Koeffizienten gefunden, so ist die gesamte weitere Vorgehensweise fast analog zur klassischen Prony-Methode. Der einzige Unterschied ist, dass die in 3.7 verwendete $M \times M$ Vandermonde Matrix jetzt eine $n \times n$ Matrix ist und das neue Gleichungssystem folgendermaßen aussieht:

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ z_1 & z_2 & z_3 & \cdots & z_n \\ z_1^2 & z_2^2 & z_3^2 & \cdots & z_n^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ z_1^{n-1} & z_2^{n-1} & z_3^{n-1} & \cdots & z_n^{n-1} \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} h(0) \\ h(1) \\ h(3) \\ \vdots \\ h(n-1) \end{pmatrix} \tag{4.6}$$

5 Implementierung

Beide Varianten der Prony-Methode wurden in Matlab implementiert. Ziel der Implementierung ist es, Klängen die richtige Tonhöhe und das richtige Klavier zuzuweisen. Die in 3 beschriebenen Parameter c_j und f_j der Exponentialsumme sind sozusagen einzigartig wie ein Fingerabdruck. Mit ihnen kann also nicht nur der Ton, sondern auch das Klavier bestimmt werden. Hierfür wurde eine Datei mit allen fj und cj der Beispieltöne angelegt. Anschließend wurde die Prony-Methode auf die Töne angewendet und die Ergebnisse wurden mit den vorhandenen Daten verglichen, um die passende Tonhöhe und das Klavier, von dem der Ton kommt, zu ermitteln.

5.1 Klassische Prony-Methode

Da die obere Schranke M für die Anzahl der Teiltöne nicht bekannt war, wird sie folgendermaßen für die Anwendung des Algorithmus auf Klänge von Klavieren abgeschätzt:

$$M = floor((size(Hk, 2) + 1)/2) \tag{5.1}$$

Hk sind alle erhaltenen gemessenen Proben des Klangs. Da für die Anwendung des Algorithmus $2M - 1$ Messwerte benötigt werden, lassen sich anhand der Anzahl an Messwerten auf M Rückschlüsse ziehen. Die Implementierung dieser Variante der Prony Methode stellte keine größeren Probleme dar. Der genaue Code befindet sich bei den Anhängen.

Die klassische Prony Methode eignet sich für die Anwendung auf Klaviertöne wesentlich schlechter als ihre erweiterte Version, da man keinerlei Aussage über die Anzahl der Teiltöne treffen kann. Außerdem hat es sich als sehr schwierig erwiesen, eine passende Anzahl an Messproben zu finden, um im Vergleich mit der erweiterten Prony Methode gute Ergebnisse zu erzielen. Deshalb werden im weiteren Verlauf der Arbeit überwiegend Ergebnisse behandelt, die mit der erweiterten Prony Methode erhalten wurden.

5.2 Erweiterte Prony-Methode

In der Implementierung der erweiterten Prony-Methode wird M identisch wie bei der klassischen Prony Methode berechnet:

$$M = floor((size(Hk, 2) + 1)/2); \tag{5.2}$$

Anschließend wird eine Hankel Matrix mit bereits zwei Spalten erstellt. Es ist davon auszugehen, dass sich mit einer Spalte keine Lösung ergibt. In folgender While-Schleife wird die Hankel Matrix weiter aufgebaut und in jeder Iteration versucht, eine Lösung zu finden:

```
while (norm(hMatrix*pk) > 10^14* 2* eps(max(svd(hMatrix))))
 n = n+1;
  if (n <= M+1)
    fn = Hk(n:M+n-1)';
    pk = [hMatrix \ -fn;1];
    hMatrix(:,n) = fn;
  end
```

Dabei haben die benutzten Variablen folgende Bedeutung:

- *hMatrix* ist die Hankel-Matrix
- *pk* ist der Vektor mit den Ergebnissen des Gleichungssystems
- fn ist die als nächste anzufügende Spalte
- n ist der aktuelle Iterationsschritt

Zu Beginn wird der Iterationsschritt erhöht. Anschließend wird überprüft, ob noch genug Messdaten vorhanden sind, um eine weitere Spalte anfügen zu können, beziehungsweise, ob die Matrix nicht größer als $M \times M$ werden würde. Danach wird fn und somit auch der neue Lösungsvektor neu besetzt. Mit diesem wird das nächste *pk* errechnet und im Anschluss wird fn an die Hankel Matrix angefügt. Wenn eine Lösung gefunden wurde, muss $norm(hMatrix * pk)$ gleich 0 sein, bzw. durch Rundungsfehler sehr nahe der 0 sein. Als Abbruchkriterium wird hier

$$norm(hMatrix * pk) > 10^{14} * 2 * eps(max(svd(hMatrix))) \tag{5.3}$$

verwendet. Von $eps(max(svd(hMatrix)))$ erhält man ein Skalar, das nur minimal größer als 0 ist, in Abhängigkeit der aktuellen Hankel Matrix. Dieses Skalar wird mit $10^{14} \cdot 2$ multipliziert. Tests mit verschiedenen Faktoren haben ergeben, dass damit die besten Ergebnisse erzielt werden. Der genaue Code befindet sich im Anhang.

5.3 Anlegen der Vergleichsdaten

Für die Anwendung des Programms wurden zwölf Mono-Klänge eingelesen, von vier verschiedenen Klavieren jeweils drei Klänge. Sie wurden mit Faktor zwanzig heruntergetaktet und anschließen wurde der Reihe nach die erweiterte Prony Methode auf sie angewendet. Die Lösungen für die jeweiligen c_j und f_j wurden in einer Matlab Datei gespeichert, damit sie für spätere Vergleichszwecke geladen werden können.

5.4 Identifikation eines Klaviertons

Wird jetzt einer der zwölf Klänge eingelesen und die erweiterte Prony Methode auf diesen angewandt, so stimmen die erhaltenen Werte der c_j und f_j mit den entsprechenden Werten von exakt diesem Klang überein. In Matlab werden diese Werte also der Reihe nach mithilfe von if-Schleifen mit den vorher angelegten Vergleichsdaten abgeglichen. Sobald eine Übereinstimmung gefunden wurde, wissen wir sowohl welcher Klang eingelesen wurde, als auch von welchem der vier Klaviere dieser gespielt wurde.

5.5 Auswertung der Ergebnisse

Interessant für die Auswertung sind hauptsächlich die Werte der f_j, da diese die wichtigen Eigenschaften des Tones beschreiben. Dabei ist der Realteil von f_j dessen Abklingrate. Er

beschreibt, wie schnell die Amplitude des Tones abnimmt, also wie schnell er leiser wird. Der Imaginärteil von f_j beschreibt wiederum die Kreisfrequenz des Tones. Je höher die Frequenz ist, desto höher wird ein Klang wahrgenommen[3][8].
Der Dämpfungsfaktor, also der Realteil der f_j ist stets negativ. Das heißt, die Schwingung nimmt mit fortlaufender Zeit ab. Wäre einer der Dämpfungsfaktoren positiv, so müsste man von einer fehlerhaften Berechnung seitens des Algorithmus ausgehen, da kein Klang im Laufe der Zeit zunehmen, bzw. lauter werden sollte.
In folgender Tabelle sind die Ergebnisse für die f_j der Klänge gleicher Tonhöhe von vier verschiedenen Klavieren dargestellt:

oneOnefj	twoOnefj	threeOnefj	fourOnefj
-0.0027444+2.5045i	-0.0068541+2.505i	-0.0020894+2.5053i	-0.0097049+0i
-0.0027444-2.5045i	-0.0068541-2.505i	-0.0020894-2.5053i	-0.0034562+2.5115i
-0.37736-3.1416i	-0.00039886+1.2554i	-0.39822-3.1416i	-0.0034562-2.5115i
-0.00087038+1.2581i	-0.00039886-1.2554i	-0.17274+2.6791i	-0.26258+2.6074i
-0.00087038-1.2581i	-0.52143+0.32897i	-0.17274-2.6791i	-0.26258-2.6074i
-0.078531+0i	-0.52143-0.32897i	-0.21346+2.4673i	-0.14019+0.60554i
-0.18408+0.84001i	0+0i	-0.21346-2.4673i	-0.14019-0.60554i
-0.18408-0.84001i	0+0i	-0.071879+1.987i	-0.0014072+1.2546i
-0.58142+1.4645i	0+0i	-0.071879-1.987i	-0.0014072-1.2546i
-0.58142-1.4645i	0+0i	-0.0079019+0i	-0.15729+1.1881i
0+0i	0+0i	-0.22033+1.6712i	-0.15729-1.1881i
0+0i	0+0i	-0.22033-1.6712i	-0.22437+1.7718i
0+0i	0+0i	-0.0024617+1.2521i	-0.22437-1.7718i
0+0i	0+0i	-0.0024617-1.2521i	0+0i
0+0i	0+0i	-0.055767+1.2789i	0+0i
0+0i	0+0i	-0.055767-1.2789i	0+0i
0+0i	0+0i	-0.13076+0.34261i	0+0i
0+0i	0+0i	-0.13076-0.34261i	0+0i
0+0i	0+0i	-0.11812+0.78403i	0+0i
0+0i	0+0i	-0.11812-0.78403i	0+0i
0+0i	0+0i	-0.23461+0.74067i	0+0i
0+0i	0+0i	-0.23461-0.74067i	0+0i
0+0i	0+0i	-1.6794+0i	0+0i

Das erste Teilwort der Spaltenbezeichnung ist jeweils die Nummer des Klaviers, von dem der Klang kommt und das zweite Teilwort ist die Nummer des Klangs. Nach diesem Schema wurden in der Matlab Implementierung alle gesuchten Variablen bezeichnet. Auffällig ist, dass meist jeweils zwei, bis auf das Vorzeichen des Imaginärteils identische Werte für f_j vertreten sind. Dies ist darauf zurückzuführen, dass die Berechnung mit komplexen Zahlen stattgefunden hat[9].
In folgender Tabelle sind alle Werte der c_j von Klang 1 von Klavier 1 aufgeführt:

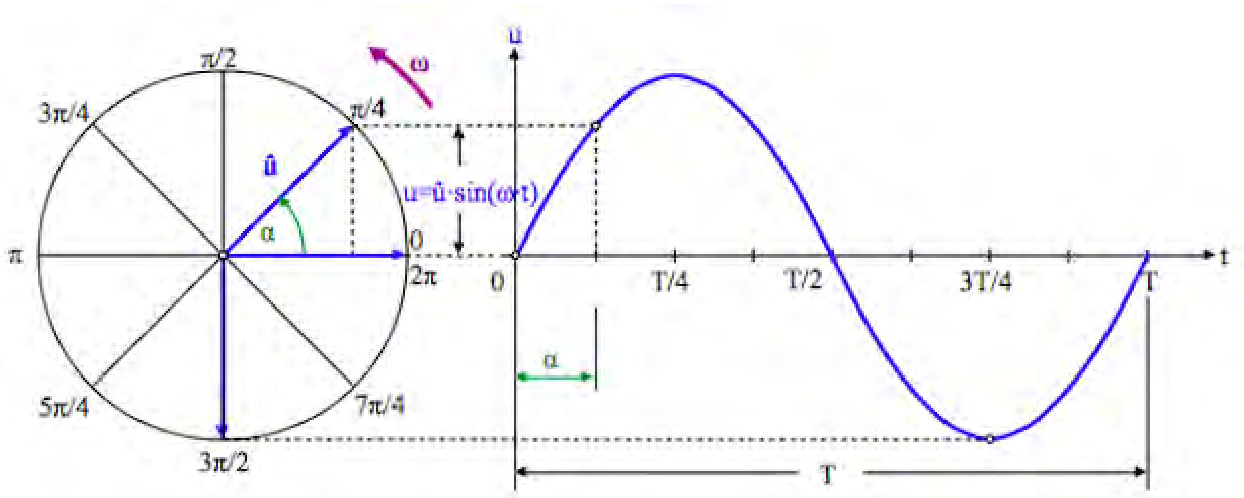

Abbildung 5.1: Quelle:[10]

```
oneOnecj
________________________

 0.00026048-0.00015772i
 0.00026048+0.00015772i
-0.00041325-9.6654e-19i
 0.00095911+0.00066024i
 0.00095911-0.00066024i
  0.0018009+6.6302e-20i
-0.00016058-0.0011985i
-0.00016058+0.0011985i
 -0.0016918+0.00040361i
 -0.0016918-0.00040361i
```

Man erkennt, dass auch die Werte der entsprechenden c_j bis auf das Vorzeichen des Imaginärteils ebenfalls gleich sind. Das lässt sich nicht nur für diesen Klang beobachten, sondern für alle getesteten Klänge. Eine harmonische Schwingung kann durch die kreisförmige Bewegung eines Zeigers dargestellt werden, wie Abbildung 5.1 zeigt.

Die Länge des Zeigers entspricht dabei der Amplitude der Schwingung. Da eine Periode der Schwingung einer Umdrehung des Zeigers entspricht und der Vollwinkel 2π ist, ist die

Kreisfrequenz das 2π fache der Frequenz. Also: $\omega = 2\pi f$. Das Vorzeichen der Kreisfrequenz bestimmt lediglich die Bewegungsrichtung des Zeigers. Bei positivem Vorzeichen bewegt sich der Zeiger wie in Abbildung 5.1 entgegen dem Uhrzeigersinn, wobei er sich mit negativem Vorzeichen mit dem Uhrzeigersinn bewegt[10][8].

Physikalisch gesehen spielt das Vorzeichen keine Rolle, daher können hier die f_j mit negativem Vorzeichen vernachlässigt werden. Die Einträge mit Wert $0 + 0i$ mussten aus Formatierungsgründen hinzugefügt werden und können ignoriert werden. Für Klang Nummer eins von Klavier eins (1. Spalte; oneOnefj) wurden also sechs Teiltöne mithilfe der in Kapitel 4 beschriebenen erweiterten Prony Methode gefunden[9].
Wie bereits erwähnt setzt sich ein Klang aus der harmonischen Überlagerung mehrerer Teiltöne zusammen. Dabei muss zwischen Grundton und Obertönen unterschieden werden. Der Grundton eines Klangs ist der Ton mit der niedrigsten Frequenz. Für eine harmonische Überlagerung muss die Frequenz aller Obertöne ein ganzes Vielfaches der Frequenz des Grundtons betragen. Betrachtet man unter diesem Gesichtspunkt die für Klang eins erhaltenen Teiltöne, fällt auf, dass die Ergebnisse für Klavier eins sehr unsauber sind[11]. Kreisfrequenz des Grundtons wäre hier $0,84001 s^{-1}$. Folgende Tabelle zeigt das Verhältnis der Kreisfrequenzen der Obertöne zu dem des Grundtons:

Kreisfrequenz in s^{-1}	Verhältnis
2,5045	2,98
3,1416	3,74
1,2581	1,50
1,4645	1,74

Während 2,98 ein sehr gutes Ergebnis ist und 3,74 und 1,74 noch im Rahmen sind, weicht 1,5 schon zu sehr ab. Ursache hierfür ist wohl, dass der Algorithmus zu früh abbricht und somit das Ergebnis nicht exakt genug ist oder der Klang nicht aus ausschließlich harmonisch zueinander schwingenden Teiltönen besteht. In der Realität ist es selten der Fall, dass die Obertöne mit exakt ganzzahligem Verhältnis zu dem Grundton stehen, da durch Umweltbedingungen Inharmonizitäten verursacht werden. Eine solche Inharmonizität kann bei einem Klavier zum Beispiel durch das Biegemoment der Saite verursacht werden.

Folgende Tabelle beschreibt die Anzahl aller von der erweiterten Prony Methode berechneten Teiltöne für die jeweiligen Klänge und Klaviere:

	Klavier 1	Klavier 2	Klavier 3	Klavier 4
Klang 1	10	6	23	13
Klang 2	155	100	59	59
Klang 3	147	75	167	59
Summe	312	181	249	131

Auffällig ist hier, dass bei jedem Klavier für Klang 1 wesentlich weniger Teiltöne gefunden wurden als für die anderen Klänge. Außerdem sticht ins Auge, dass sich die Summen der von den Klavieren erzeugten Teiltöne sehr stark unterscheiden. Für Klavier 1 wurden etwa 2,4 mal so viele Töne wie für Klavier 4 gefunden. Die Anzahl der Obertöne eines Klangs hat Auswirkung auf die Klangfarbe. Allgemein kann man sagen, dass Klänge mit weniger Obertönen als klarer, beziehungsweise reiner wahrgenommen werden und farbiger, je mehr Obertöne sie haben. Man kann somit Rückschlüsse auf die Klangfarbe der einzelnen Klaviere ziehen. Klavier 2 und 4 haben verglichen mit den anderen beiden Klavieren also den etwas reineren Klang[2].

Im Anhang befindet sich eine Matlab Datei, die alle c_j und f_j aller Klänge enthält.

6 Zusammenfassung

Nachdem beide Varianten der Prony Methode im Laufe der Arbeit ausführlich beschrieben wurden, erfolgt hier jeweils eine kurze Zusammenfassung:

6.1 Klassische Prony Methode

- obere Schranke $M \in \mathbb{N}$ bekannt
- Erstellen des Gleichungssystems mit Hankel Matrix zum Berechnen der Koeffizienten des Prony Polynoms
- Bestimmen der Nullstellen des Prony Polynoms mithilfe der Companion Matrix
- Berechnen der f_j mithilfe der Nullstellen des Prony Polynoms; $j = 1, ..., M$
- Bestimmen der c_j durch Lösen des Vandermonde Systems

6.2 Erweiterte Prony Methode

- obere Schranke M ist nicht bekannt und wird anhand der Anzahl der Messwerte abgeschätzt
- unveränderter Algorithmus bis zum Berechnen der Koeffizienten des Prony Polynoms
- spaltenweises Aufbauen der Hankel Matrix bis das Gleichungssystem lösbar ist
- M ist gleich der Anzahl der benötigten Iterationsschritte

6.3 Implementierung

Hinsichtlich der Anwendung auf Klänge von Klavieren liefert die erweiterte Prony Methode wesentlich aussagekräftigere Ergebnisse, da man mit ihr Rückschlüsse auf M ziehen kann. Die klassische Prony Methode ist ohne gegebenes M schwierig anzuwenden, da M so meist viel zu hoch gewählt wird. Auch mit der erweiterten Prony Methode kann man jedoch auch keine exakten Aussagen über die Anzahl der Teiltöne der jeweiligen Klavierklänge treffen, da die Ergebnisse für anders gewählte Abbruchkriterien für das Aufbauen der Hankel Matrix sehr stark variieren. Allerdings sind durchaus Unterschiede in der Klangfarbe von Klavieren festzustellen, wenn man die errechneten M in Relation zueinander betrachtet. Möglicherweise ließe sich durch eine geeignetere Abbruchbedingung die Implementierung der erweiterten Prony Methode in Matlab noch etwas verbessern. Mit der aktuellen Implementierung lässt sich dennoch ohne Probleme bestimmen, welcher der zwölf Klänge gespielt wurde. Es hat sich also bestätigt, dass die berechneten Teiltöne wie ein Fingerabdruck für den Klang sind.

A Anhang

Im Anhang befindet sich der Matlab Code zum Einlesen der zwölf Klänge mit erweiterter Prony Methode und Code, um einen Klang mithilfe dieser Methode zu identifizieren. Außerdem liegt eine CD bei mit diesen Codes, zusätzlichem Code der klassischen Prony Methode und den Ergebnissen der Variablen nach der Durchführung der erweiterten Prony Methode.

Klänge einlesen:

```
%Prony Methode + Erweiterung

%Einlesen des Audio Files
[one,Fs] = audioread('Prony/pianos/1_a0mono.wav');
one = downsample(one,20)';

[two,Fs] = audioread('Prony/pianos/1_a1mono.wav');
two = downsample(two,20)';

[three,Fs] = audioread('Prony/pianos/1_a2mono.wav');
three = downsample(three,20)';

[four,Fs] = audioread('Prony/pianos/2_a0mono.wav');
four = downsample(four,20)';

[five,Fs] = audioread('Prony/pianos/2_a1mono.wav');
five = downsample(five,20)';

[six,Fs] = audioread('Prony/pianos/2_a2mono.wav');
six = downsample(six,20)';

[seven,Fs] = audioread('Prony/pianos/3_a0mono.wav');
seven = downsample(seven,20)';

[eight,Fs] = audioread('Prony/pianos/3_a1mono.wav');
eight = downsample(eight,20)';

[nine,Fs] = audioread('Prony/pianos/3_a2mono.wav');
nine = downsample(nine,20)';

[ten,Fs] = audioread('Prony/pianos/4_a0mono.wav');
ten = downsample(ten,20)';

[eleven,Fs] = audioread('Prony/pianos/4_a1mono.wav');
eleven = downsample(eleven,20)';

[twelve,Fs] = audioread('Prony/pianos/4_a2mono.wav');
twelve = downsample(twelve,20)';

iterator = 1;
while(iterator <= 12)
switch iterator
    case 1
        Hk = one;
    case 2
        Hk = two;
    case 3
        Hk = three;
    case 4
        Hk = four;
    case 5
```

```
            Hk = five;
        case 6
            Hk = six;
        case 7
            Hk = seven;
        case 8
            Hk = eight;
        case 9
            Hk = nine;
        case 10
            Hk = ten;
        case 11
            Hk = eleven;
        case 12
            Hk = twelve;
        otherwise
            disp('falsche switch eingabe')
end
M = floor((size(Hk,2) + 1) /2) ;

%Erstellung des linearen Gleichungssystems (7)

%Spaltenweises Aufbauen der Matrix

hMatrix = Hk(1:M)';
n = 2;
fn = Hk(n: (M +n-1))';
hMatrix(:,n) = fn;
pk = [hMatrix(:,n-1) \ -hMatrix(:,n),1]';

while (norm(hMatrix*pk) > 10^14* 2* eps(max(svd(hMatrix))) )
 n = n+1;
  if (n <= M+1)
    fn = Hk(n:M+n-1)';
    pk = [hMatrix \ -fn;1];
    hMatrix(:,n) = fn;
  end
end

%Erstellen der Companion Matrix
cMatrix = compan(flipud(pk));

%Eigenwerte der companion Matrix
zj = (eig(cMatrix))';

%Berechnen von fj
fj = log(zj);

%Berechnen von cj

%Erstellen des Vandermonde Systems
vMatrix = fliplr(vander(zj))';
```

```
solv = (Hk(1:size(zj')))';

cj = vMatrix \ solv;

switch iterator
    case 1
        oneZerocj = cj;
        oneZerofj = fj;
    case 2
        oneOnecj = cj;
        oneOnefj = fj;
    case 3
        oneTwocj = cj;
        oneTwofj = fj;
    case 4
        twoZerocj = cj;
        twoZerofj = fj;
    case 5
        twoOnecj = cj;
        twoOnefj = fj;
    case 6
        twoTwocj = cj;
        twoTwofj = fj;
    case 7
        threeZerocj = cj;
        threeZerofj = fj;
    case 8
        threeOnecj = cj;
        threeOnefj = fj;
    case 9
        threeTwocj = cj;
        threeTwofj = fj;
    case 10
        fourZerocj = cj;
        fourZerofj = fj;
    case 11
        fourOnecj = cj;
        fourOnefj = fj;
    case 12
        fourTwocj = cj;
        fourTwofj = fj;
    otherwise
        disp('falsche switch eingabe')
end
iterator = iterator +1;
end
save('solutions');
```

Klang identifizieren:

```
%Prony Methode + Erweiterung

```

```
%Einlesen des Audio Files
[y,Fs] = audioread('Prony/pianos/1_a1mono.wav');

Hk = downsample(y,20)';
M = floor((size(Hk,2) + 1) /2) ;

%Erstellung des linearen Gleichungssystems (7)

%Spaltenweises Aufbauen der Matrix

hMatrix = Hk(1:M)';
n = 2;
fn = Hk(n: (M +n-1))';
hMatrix(:,n) = fn;
pk = [hMatrix(:,n-1) \ -hMatrix(:,n),1]';

while (norm(hMatrix*pk) > 10^14 * eps(max(svd(hMatrix)))&& n <= 60)
 n = n+1;
  if (n <= M+1)
    fn = Hk(n:M+n-1)';
    pk = [hMatrix \ -fn;1];
    hMatrix(:,n) = fn;
  end
end

%Erstellen der Companion Matrix
cMatrix = compan(flipud(pk));

%Eigenwerte der companion Matrix
zj = (eig(cMatrix))';

%Berechnen von fj
fj = log(zj);

%Berechnen von cj

%Erstellen des Vandermonde Systems
vMatrix = fliplr(vander(zj))';
solv = (Hk(1:size(zj')))';

cj = vMatrix \ solv;


isTone = 0 ;
if(isTone == 0)
    load 'solutions' 'oneZerocj'
    load 'solutions' 'oneZerofj'
    if((size(cj,1) == size(oneZerocj,1)) && (size(fj,2) == size(oneZerofj,2)))
        isTone = 1;
        for i = 1: 1: size(cj)
            if(cj(i) ~= oneZerocj(i)&& fj(i) ~= oneZerofj(i))
                isTone = 0;
```

```
            end
        end
        if(isTone ==1)
            disp('Klavier 1, Ton 0')
        end
    end
end

if(isTone == 0)
    load 'solutions' 'oneOnecj'
    load 'solutions' 'oneOnefj'
    if((size(cj,1) == size(oneOnecj,1)) && (size(fj,2) == size(oneOnefj,2)))
        isTone = 1;
        for i = 1: 1: size(cj)
            if(cj(i) ~= oneOnecj(i)&& fj(i) ~= oneOnefj(i))
                isTone = 0;
            end
        end
        if(isTone == 1)
            disp('Klavier 1, Ton 1')
        end
    end
end

if(isTone == 0)
    load 'solutions' 'oneTwocj'
    load 'solutions' 'oneTwofj'
    if((size(cj,1) == size(oneTwocj,1)) && (size(fj,2) == size(oneTwofj,2)))
        isTone = 1;
        for i = 1: 1: size(cj)
            if(cj(i) ~= oneTwocj(i)&& fj(i) ~= oneTwofj(i))
                isTone = 0;
            end
        end
        if(isTone == 1)
            disp('Klavier 1, Ton 2')
        end
    end
end

if(isTone == 0)
    load 'solutions' 'twoZerocj'
    load 'solutions' 'twoZerofj'
    if((size(cj,1) == size(twoZerocj,1)) && (size(fj,2) == size(twoZerofj,2)))
        isTone = 1;
        for i = 1: 1: size(cj)
            if(cj(i) ~= twoZerocj(i)&& fj(i) ~= twoZerofj(i))
                isTone = 0;
            end
        end
        if(isTone == 1)
            disp('Klavier 2, Ton 0')
```

```
        end
    end
end

if(isTone == 0)
    load 'solutions' 'twoOnecj'
    load 'solutions' 'twoOnefj'
    if((size(cj,1) == size(twoOnecj,1)) && (size(fj,2) == size(twoOnefj,2)))
        isTone = 1;
        for i = 1: 1: size(cj)
            if(cj(i) ~= twoOnecj(i)&& fj(i) ~= twoOnefj(i))
                isTone = 0;
            end
        end
        if(isTone == 1)
            disp('Klavier 2, Ton 1')
        end
    end
end

if(isTone == 0)
    load 'solutions' 'twoTwocj'
    load 'solutions' 'twoTwofj'
    if((size(cj,1) == size(twoTwocj,1)) && (size(fj,2) == size(twoTwofj,2)))
        isTone = 1;
        for i = 1: 1: size(cj)
            if(cj(i) ~= twoTwocj(i)&& fj(i) ~= twoTwofj(i))
                isTone = 0;
            end
        end
        if(isTone == 1)
            disp('Klavier 2, Ton 2')
        end
    end
end

if(isTone == 0)
    load 'solutions' 'threeZerocj'
    load 'solutions' 'threeZerofj'
    if((size(cj,1) == size(threeZerocj,1)) && (size(fj,2) == size(threeZerofj,2)))
        isTone = 1;
        for i = 1: 1: size(cj)
            if(cj(i) ~= threeZerocj(i)&& fj(i) ~= threeZerofj(i))
                isTone = 0;
            end
        end
        if(isTone == 1)
            disp('Klavier 3, Ton 0')
        end
    end
end
```

```
if(isTone == 0)
    load 'solutions' 'threeOnecj'
    load 'solutions' 'threeOnefj'
    if((size(cj,1) == size(threeOnecj,1)) && (size(fj,2) == size(threeOnefj,2)))
        isTone = 1;
        for i = 1: 1: size(cj)
            if(cj(i) ~= threeOnecj(i)&& fj(i) ~= threeOnefj(i))
                isTone = 0;
            end
        end
        if(isTone == 1)
            disp('Klavier 3, Ton 1')
        end
    end
end

if(isTone == 0)
    load 'solutions' 'threeTwocj'
    load 'solutions' 'threeTwofj'
    if((size(cj,1) == size(threeTwocj,1)) && (size(fj,2) == size(threeTwofj,2)))
        isTone = 1;
        for i = 1: 1: size(cj)
            if(cj(i) ~= threeTwocj(i)&& fj(i) ~= threeTwofj(i))
                isTone = 0;
            end
        end
        if(isTone == 1)
            disp('Klavier 3, Ton 2')
        end
    end
end

if(isTone == 0)
    load 'solutions' 'fourZerocj'
    load 'solutions' 'fourZerofj'
    if((size(cj,1) == size(fourZerocj,1)) && (size(fj,2) == size(fourZerofj,2)))
        isTone = 1;
        for i = 1: 1: size(cj)
            if(cj(i) ~= fourZerocj(i)&& fj(i) ~= fourZerofj(i))
                isTone = 0;
            end
        end
        if(isTone == 1)
            disp('Klavier 4, Ton 0')
        end
    end
end

if(isTone == 0)
    load 'solutions' 'fourOnecj'
    load 'solutions' 'fourOnefj'
    if((size(cj,1) == size(fourOnecj,1)) && (size(fj,2) == size(fourOnefj,2)))
```

```
        isTone = 1;
        for i = 1: 1: size(cj)
            if(cj(i) ~= fourOnecj(i)&& fj(i) ~= fourOnefj(i))
                isTone = 0;
            end
        end
        if(isTone == 1)
            disp('Klavier 4, Ton 1')
        end
    end
end

if(isTone == 0)
    load 'solutions' 'fourTwocj'
    load 'solutions' 'fourTwofj'
    if((size(cj,1) == size(fourTwocj,1)) && (size(fj,2) == size(fourTwofj,2)))
        isTone = 1;
        for i = 1: 1: size(cj)
            if(cj(i) ~= fourTwocj(i)&& fj(i) ~= fourTwofj(i))
                isTone = 0;
            end
        end
        if(isTone == 1)
            disp('Klavier 4, Ton 2')
        end
    end
end
```

Literaturverzeichnis

[1] Miroslav Großer. Die magie des klanges oder die wirkung der obertöne. http://www.stimmlabor.de/pdf/wirkungen-obertoene-obertonsingen.pdf.

[2] Franziska Bischoff. Die klangfarbe. http://klangschreiber.de/2013/02/15/die-klangfarbe/, 2013. besucht = 27.3.2017.

[3] Gerlind Plonka and Manfred Tasche. Prony methods for recovery of structured functions. *GAMM-Mitteilungen*, 37(2):239–258, 2014.

[4] Lineare gleichungssysteme in matrix-vektorform. http://www.dieter-heidorn.de/Mathematik/RP_LA_AG1/K3_Gleichungssysteme/K33_MatrixVektorForm/MatrixVektorForm.html. besucht: 26.3.2017.

[5] John W Layman. The hankel transform and some of its properties. *J. Integer Seq*, 4(1):1–11, 2001.

[6] Alan Edelman and H Murakami. Polynomial roots from companion matrix eigenvalues. *Mathematics of Computation*, 64(210):763–776, 1995.

[7] Dan Kalman. The generalized vandermonde matrix. *Mathematics Magazine*, 57(1):15–21, 1984.

[8] Udo Ahlvers. Nachrichtentechnik [NAT] - Kapitel 2: Zeitkontinuierliche Signale. https://www.hsu-hh.de/download-1.5.1.php?brick_id=oYTxKHRyFtd7LTBe, 2005. besucht: 23.3.2017.

[9] Behandlung der komplexen darstellung von wellen: Negative frequenzen und komplexe felder. http://www.bmo.physik.uni-muenchen.de/buecher/optik-lwp/Zusatz/Komplexe_Felder.pdf. besucht: 26.3.2017.

[10] Zeigerdarstellung von sinusförmigen wechselgrößen. `http://elektronik-kurs.net/elektrotechnik/wechselstromkreise/`. besucht: 26.3.2017.

[11] Philipp Ackermann. Computer und musik: eine einführung in die digitale klang-und musikverarbeitung. pages 12–18, 2013.

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, dass alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht und dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt wurde.

Ort, Datum Unterschrift