Räumliche Repräsentation - Untersuchung diverser Implementierungsansätze

Bachelorarbeit aus Aufnahmetechnik 1, SE

Nico Selim Seddiki

Betreuung: Univ.Prof. DI Dr. Alois Sontacchi

Graz, 14. November 2019

kunst uni graz

institut für elektronische musik und akustik

iem

Zusammenfassung

Räumliche Repräsentation stellt oftmals noch immer eine großen Rechenaufwand dar. Diese Arbeit beschäftigt sich mit unterschiedlichen Implementierungsansätzen wie FIR-Filter, uniforme und non-uniforme Faltungsalgorithmen in Hinblick auf Performance und Komplexität.

Inhaltsverzeichnis

1 Einleitung

Die Arbeit beschäftigt sich mit diversen Implementierungsansätzen wie Raumklang simuliert werden kann. Der RoomEncoder der IEM Plug-In Suite [1] bildet den Ausgangspunkt dieser Betrachtungen. Basierend auf einem geometrischen raumakustischen Modell (Spiegelquellenverfahren, Kapitel 2.1.3) ist es möglich, Schallquelle und Empfänger dynamisch in einem virtuellen Raum zu platzieren. Die Positionen von der Quelle und des Empfängers können automatisiert werden. Damit lässt sich zum Beispiel der Dopplereffekt [2] modellieren. Eine weitere Möglichkeit des Plugin stellt die Modellierung der Abstrahlcharakteristik der Quelle dar. Dies bedeutet, dass die virtuelle Schallquelle eine frequenz- und zeitabhängige Richtwirkung besitzen kann. Zusätzlich lassen sich über 200 Reflektionen in Echtzeit simulieren. Aufgrund der speziell verwendeten Filter Struktur (siehe Kapitel 3.1.2) besitzt der RoomEncoder auch eine geringe Latenz zwischen Ein- und Ausgangssignal. All diese Eigenschaften verursachen eine deutliche CPU Last. Für dynamische Quellen wäre dieser Rechenaufwand durchaus gerechtfertigt, für statische Quellen würde man jedoch einen Großteil der vorhanden Ressourcen einbüßen. Die dünn besetze (engl. sparse) Filterstruktur ist zwar sehr schnell und eignet sich besonders für kurze Impulsantworten (engl. Impulse Response bzw. IR genannt), bei längeren IRs steigt der Rechenaufwand jedoch stark an (siehe Kapitel 3.1.1). Basierend auf diesen Tatsachen kann man folgende fragen stellen:

— Kann man bei statischen Quelle-Empfänger Anordnungen nicht die IR berechnen und diese in einem Faltungs Plugin auf das Quellsignal anwenden?
— Für welche IR Länge eignet sich welcher Faltungsalgorithmus (in engl. Convolution Algorithmn) am besten?

Um diese Fragen zu beantworten werden dabei drei Faltungsalgorithmen untersucht: dünn besetzte Filterstruktur (Sparse-FIR-Filter), gleichmäßig partitionierte Faltung (engl. Uniformly Partitioned Convolution. Dabei werden Eingangssignal und Impulsantwort des Filters in Blöcke gleicher Größe unterteilt) und nicht gleichmäßig partitionierte Faltung (engl. Uniformly Partitioned Convolution . Die Filterimpulsantwort wird in Blöcke unterschiedlicher Größe partitioniert). Anhand von Messungen soll (Kapitel 4) deren Vorteile für unterschiedliche Filter bzw. Impulsantwortlängen erörtert werden.

1. https://plugins.iem.at/
2. https://de.wikipedia.org/wiki/Doppler-Effekt

Diese Arbeit ist in einem theoretischen und analytischen Teil gegliedert. Im Theorieteil werden zunächst die Grundlagen erläutert um die unterschiedlichen Algorithmen besser verstehen zu können:

— Raumakustische Theorie (besonders der geometrischen Raumakustik und Nachhallmodellen).
— Perzeption: Was zeichnet einen guten Raumklang aus.
— Grundlagen der Signalverarbeitung und Faltungsalgorithmen (Sparse-FIR, Uniformly und Non-Uniformly Partitioned Convolution).
— Komplexität der Algorithmen.

Im analytischen Teil wird ausschließlich auf die Messungen eingegangen und diese im Anschluss diskutiert:

— Messtechnische Grundlagen (Sampling und Instrumentation)
— Diskussion der Ergebnisse.

2 Theorie Teil

2.1 Raumakustische Theorie

Prinzipiell lässt sich die Raumakustik in 3 Teilbereiche gliedern:

- Wellentheoretische Raumakustik
- Statistische Raumakustik
- Geometrische Raumakustik

2.1.1 Wellentheoretische Raumakustik

Ausgangspunkt der wellentheoretischen Raumakustik sind die Wellengleichungen. Aufgrund der mathematischen Komplexität (Partielle Differentialgleichungen) lassen sich Lösungen für solche Gleichungen nur für einfache Raumformen berechnen. Man greift dann gerne auf dieses Konzept zurück, wenn die betrachtete Wellenlänge des Schalls in der Größenordnung des zu betrachtenden Raums liegt. Die ist besonders bei tiefen Frequenzen der Fall ist [GW14].

2.1.2 Statistische Raumakustik

In der statistischen Raumakustik werden zeitliche Energiedichteverteilungen in Räumen betrachtet. Dies ist mathematisch deutlich leichter zu bewerkstelligen, als dass man einen Schallstrahl, der mitunter an den Raumbegrenzungsflächen reflektiert wird, beobachtet. Von Interesse ist daher das diffuse Schallfeld und der Nachhall [GW14].

2.1.3 Geometrische Raumakustik

Grundlage für diesen Teilbereich der Akustik sind die Gesetze der Optik. Mit dem einfachen Modell des Schallstrahls lassen sich die ersten Raumreflektionen und der Direktschall modellieren. Der RoomEncoder basiert auf dem Konzept der Spiegelquelle, dass in Kapitel 2.1.7 etwas genauer beschrieben werden soll [GW14].

Ähnlich wie in der Optik, wird in der geometrischen Raumakustik die akustische Wellenausbreitung als gerade Schallstrahlen modelliert. Diese Schallstrahlen müssen nicht zwangsläufig die gleichen Energieanteile besitzen. Dadurch lassen sich beispielsweise unterschiedliche Abstrahlcharakteristiken modellieren. In Räumen ist zwangsläufig mit Störungen in der Schallausbreitung zu rechnen. Wie in der Optik wird unterscheiden zwischen [GW14]:

— *Reflexion*
— *Beugung*
— *Abschattung*
— *Brechung*

Wichtig bei der Unterscheidung zwischen den einzelnen Wellenphänomenen ist hierbei die Helmholtzzahl. Die Helholtzzahl ist definiert als das Verhältnis zwischen den geometrischen Abmessungen des Hindernisses und der betrachteten Wellenlänge λ [GW14].

2.1.4 Brechung

Die Konzepte der geometrischen Raumakustik werden für den mittleren und hohen Frequenzbereich zur Hilfe gezogen. Des Weiteren nehmen mit zunehmender Frequenz Reflexionen höherer Ordnung zu. Im Folgenden gehen wir davon aus, dass die Forderung nach großen reflektierenden Flächen erfüllt ist [ZZ93].

Wie in der Akustik tritt **Brechung** bei Mediumsübergängen auf (zum Beispiel zwischen unterschiedlichen Luftschichten). Hier gilt das Fermatsche Prinzip [3]: Jede Welle, wie beispielsweise Licht- oder Schallwelle, nimmt den Weg mit der kürzesten Laufzeit. Wenn die Ausbreitungsgeschwindigkeit bzw. Schallgeschwindigkeit c für verschiedene Wellenzüge unterschiedlich ist, so ändert sich die Richtung mit der sich die Welle ausbreitet. Lichtstrahlen werden die in Richtung von Bereichen mit geringer Ausbreitungsgeschwindigkeit (= optisch dichteres Medium) hin abgelenkt. Ein analoges Verhalten kann bei Schallstrahlen festgestellt werden. [ZZ93].

Wenn ein Schallstrahl unter einem Winkel ϕ_1 mit der Schallgeschwindigkeit c_1 in ein Medium mit ϕ_2 bzw. c_2 so gilt das Brechungsgesetz von Snellius [4] [ZZ93]:

$$c_1 \cdot sin(\phi_2) = c_2 \cdot sin(\phi_1) \tag{2.1}$$

$$\frac{sin(\phi_2)}{sin(\phi_1)} = \frac{c_2}{c_1} \tag{2.2}$$

Die Variablen ϕ_1 und ϕ_2 sind jene Winkel zwischen den in Ausbreitungsrichtung gedachten Schallstrahlen und der Flächennormalen. Ein- und ausfallender Strahl, sowie die Normale liegen in einer Ebene. Ist c in beiden Medien gleich, so tritt keine Brechung auf. Dies kann leicht anhand der Formeln (2.1) bzw. (2.2) nachvollzogen werden, in dem $c_1 = c_2$.

3. https://de.wikipedia.org/wiki/Fermatsches_Prinzip
4. https://de.wikipedia.org/wiki/Snelliussches_Brechungsgesetz

Dadurch ergibt sich für ein und -ausfallenden Strahl der selbe Winkel. Brechung kann aber auch dann entstehen, wenn die Schallgeschwindigkeit c aufgrund unterschiedlicher Temperaturverteilungen ortsabhängig ist oder die Windgeschwindigkeit einen Einfluss auf die Schallausbreitung hat [ZZ93]:

Mit zunehmender Höhe nimmt die Temperatur ab (dies entspricht in der Optik dem Übergang in ein optisch dichteres Medium). Hieraus folgt, dass der Schall nach oben abgelenkt und die Hörbarkeit vermindert wird. Andererseits nimmt mit der Höhe die Windgeschwindigkeit zu. Falls sich die Schallwelle in Windrichtung ausbreitet, so wird diese nach unten gebrochen. In der Optik würde dies dem Übergang vom optisch dichteren ins optisch dünnere Medium gleichkommen. Breitet sich der Schall gegen die Windrichtung aus, so ergibt sich der umgekehrte Sachverhalt [ZZ93].

Zusammengefasst kann man sagen, dass Brechung die Schallausbreitung im Empfangsmedium (z.B. ein Hindernis auf welches die Schallwelle trifft) beschreibt [ZZ93].

2.1.5 Absorption, Beugung, Reflexion, Streuung

Jedes Hindernis, auf das eine Schallwelle trifft hat eine unterschiedliche Schallkennimpedanz Z_0. Ein Teil der Schallenergie tritt in das Hindernis ein, der andere wird wiederum gestreut. Der Schalleintritt wird als **Absorption** bezeichnet. Bei der **Streuung** muss unterschieden werden, inwieweit das Hindernis die Schallwelle ablenkt. Von **Beugung** wird dann gesprochen, wenn die Welle nur wenig in ihrer Ausbreitung gestört wird [ZZ93].

Bei einer starken Ablenkung spricht man von **Reflexion**. Dies ist besonders dann der Fall, wenn die Schallwelle zur Schallquelle zurückgeworfen wird. Man unterscheidet regulären (an *glatten* Oberflächen) und diffusen Reflexionen (an *grob* gegliederten Oberflächen). Voraussetzung für erstere ist, dass die reflektierende Fläche größer ist als die betrachtete *Wellenlänge*. Dieser Sachverhalt ist bei hohen und mittleren Frequenzen gegeben. Außerdem gilt für solche Reflexionen das Reflexionsgesetz im strengen Sinne: *Einfallswinkel* = *Reflexionswinkel*. Dabei befinden sich der einfallende bzw. reflektierte Strahl und das Einfallslot in einer Ebene. Bei diffuser Reflexion werden die an der Oberfläche reflektierten Strahlen gestreut zurückgeworfen (Lambertsche Gesetz [5]) [ZZ93].

Nach dem Energiesatz muss die auf der Grenzfläche auftretenden Energie gleich der Summe aus absorbierter und gestreuter Energie sein. In beiden Medien werden gleiche Bewegungsgrößen (zum Beispiel Schallschnelle) gefordert. Wenn aber Schallwellen auf ein Hindernis auftreffen, so können sich die Schallkennimpedanzen um mehrere Zehnerpotenzen unterschieden. Die Folge hiervon ist, dass ein Großteil der Energie reflektiert [6] wird. Nur ein kleiner Anteil davon wird in den Festkörper eingekoppelt [ZZ93].

In diesem besitzt die Schall- bzw. Feststoffwelle die Schnelle v_F. An der Grenzschicht ergibt sich für die Luftmoleküle durch die Überlagerung der hin und -rücklaufenden Welle die Schnelle v_L. Ist Z_0 im Hindernis größer als in der Luft, so wird die Welle gegenphasig zurück reflektiert. Im umgekehrten Fall wird diese gleichphasig zurückgeworfen [ZZ93].

5. https://de.wikipedia.org/wiki/Lambertsches_Gesetz
6. Man spricht auch von einer **schallharten** Reflexion

Ist das Hindernis klein gegenüber der Wellenlänge λ, so kann dieses durch eine kleine starre Kugel ersetzt werden. Dabei wird jeder Kugel ein Streuquerschnitt zugeordnet, der kleiner als der geometrische Querschnitt ist. Das gilt aber nur, wenn $Z_{Hindernis} > Z_{Luft}$. Andernfalls ist der äquivalente Streuquerschnitt größer als der geometrische. Infolgedessen beeinflussen viele Gasblasen die Schallausbreitungen in Flüssigkeiten stark, hingegen kleine Wassertropfen in der Luft nicht [ZZ93].

Im Schallfeld überlagern sich nun die ungestörte Schallwelle bzw. **Primärwelle** und die vom Hindernis abgestrahlte **Beugungswelle** [ZZ93].

Bei kleinen Hindernissen ist die Energie der Beugungswelle wesentlich kleiner, als die der Primärwelle. Die Schallwelle wird sich relativ ungestört ausbreiten können, der Effekt der Abschattung ist gering. Hindernisse die groß gegenüber der Wellenlänge sind, führen zu Schallfeldstörungen. Hierbei kommt es zu **Reflexion** (tritt bei der zur Schallwelle zugewandten Seite auf) und **Beugung** (tritt an den Rändern auf). Der Bereich hinter dem Hindernis wird mit weniger Energie beschallt, als die Vorderseite oder Ränder. Man spricht auch von **Abschattung** [ZZ93].

2.1.6 Dämpfung, Dissipation, Transmission

Wie im Unterkapitel 2.1.5 bereits geschildert wird an einem Hindernis die Schallenergie teilweise reflektiert. Der absorbierte Anteil wird partiell irreversibel in Wärme umgewandelt oder als Körperschall im Medium weitergeleitet. Dieser Vorgang wird **Dissipation** bzw. **Transmission** genannt. Von **Dämpfung** wird dann gesprochen, wenn einer Schwingung Energie (z.B Gitarrensaite) entzogen oder wenn durch vergrößern einer Masse eine Bewegungsgröße (bspw. eine schwingende Membran) reduziert wird [ZZ93]

2.1.7 Ebene Spiegelungen:

In einem Raum werden Schallwellen mehrmals an den Wänden reflektiert. Jede Reflexion an einer schallharten Wand bzw. Fläche kann wiederrum als Ausgangspunkt einer Schallquelle angesehen werden. Diese mehrfachen Reflexionen können mit dem Modell der Spiegelquellen dargestellt werden. Wie man anhand der Abbildung (1) sehen kann, erhaltet man für einen rechteckförmigen Raum eine Reihe von Spiegel- bzw. Ersatzquellen [MC03].

Das in dem Raum resultierende Schallfeld lässt sich aus den Schallanteilen der Originalquelle und den Spiegelquellen für geometrisch einfache Raumformen gut annähern. Die Verzögerung zwischen beiden Anteilen wird dabei durch die Laufstrecke bzw. -zeit, der von der Spiegelquelle ausgehende Schallwelle zum Beobachtungspunkt ausgedrückt [MC03].

Angenommen, die Schallquelle sendet einen kurzen Impuls aus: Für die ersten Rückwürfe wird der Eintreffzeitpunkt der Impulse von der Lage des Schallsenders bzw. Schallempfängers bestimmt (siehe Abbildung (1). Für Spiegelquellen höherer Ordnung verschwimmen diese Unterschiede immer mehr [MC03].

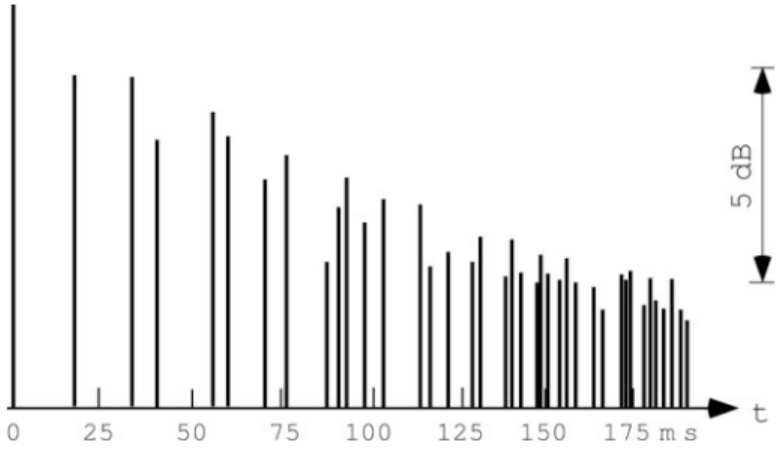

Abbildung 1 – Zeitliche Abfolge von Rückwürfen in einem von ebenen Flächen begrenzten Raum. Quelle: [Moe15]

Die Anzahl der N Impulse bis zum Zeitpunkt t, wobei $t_0 = t = 0$ dem Sendezeitpunkt der originalen Schallquelle entspricht, kann durch die Anzahl der Quellen innerhalb einer Kugel mit dem Radius $r = c \cdot t$ abgeschätzt werden [MC03].

Die Zahl der Rückwürfe entspricht dem Verhältnis aus Kugel- und Raumvolumen (V_S bzw. V):

$$N = \frac{V_s}{V} = \frac{\frac{4\pi}{3} \cdot (c \cdot t)^3}{V} \tag{2.3}$$

Mit der Zeit nimmt die Dichte der eintreffenden Impulse zu:

$$\frac{\Delta N}{\Delta t} \approx \frac{\mathrm{d}N}{\mathrm{d}t} = \frac{4\pi (c \cdot t)^2 \cdot c}{V} \tag{2.4}$$

Jedoch nimmt die Energie dieser eintreffenden Impulse mit wachsender Zeit ab:

$$\mathrm{E}_{ein} \approx \frac{1}{(c \cdot t)^2} \tag{2.5}$$

Die Energie E in einem Raumpunkt lässt sich aus dem Produkt der Anzahl eintreffender Impulse pro Zeiteinheit zu ihrer jeweiligen Energie berechnen. Die Energiedichte nimmt mit einem zeitlichen konstanten Betrag ab [MC03]:

$$E = E_{ein} \cdot \frac{\Delta N}{\Delta t} = \text{konstant} \tag{2.6}$$

Je kleiner die Raumdimensionen gegenüber der Entfernung zur Spiegelquelle werden, desto geringer wird der Einfluss des Beobachtungszeitpunkt. Aus diesem Grund würde man sich nun konstante örtliche und zeitliche Energieverläufe erwarten. Die Praxis zeigt aber, dass aufgrund von Raummoden, der Dämpfung längs der Ausbreitungswege der Schallquelle und Absorption an den Wänden dies nicht zu erwarten ist [MC03].

2.2 Raumakustische Messgrößen

Raumakustische Messgrößen dienen dazu, Raumklang unter objektiven Gesichtspunkten zu betrachten. Die im Folgenden beschriebenen Messgrößen können für die Charakterisierung von raumakustischen Gütemaßen herangezogen werden. Im Wesentlichen lassen sich diese durch das Verhältnis der Energie des Direktschalls, der Anfangsreflexionen und des Nachhalls zur Gesamtenergie des Schallereignisses herleiten. Da sich jeder Raum sehr gut als LZI (**l**ineares **z**eit **i**nvariantes)-System approximieren lässt, reicht in der Regel die Raumimpulsantwort $h(t)$ (zeit-kontinuierlich) bzw. $h[n]$ (zeit-diskret) aus, um diesen zu beschreiben[7]. Mittels der Impulsantwort können dann jene Parameter abgeleitet werden, die das Ausklangsverhalten des Raumes beschreiben. Beispielsweise wären die Schallenergiedichte $w(t)$ (die der quadrierten Impulsantwort $h^2(t)$ entspricht) oder die kumulierte Schallenergiedichte [GW14]:

$$w(t) \sim h^2(t) \tag{2.7}$$

$$W(t) = \int_0^t h^2(\tau) d\tau \tag{2.8}$$

Die kumulierte Schallenergiedichte entspricht der Summation/Mittlung jener Energiedichte über ein bestimmtes Zeitintervall $(0, t)$. Möchte man zum Beispiel das Trägheitsverhalten des Gehörs in die Berechnungen miteinbeziehen, so kann man dies durch einführen einer Zeitkonstante τ_0 erreichen. Die daraus resultierende Schallenergiedichte/Schallintensität I_{τ_0} beträgt [GW14]:

$$I_{\tau_0} \sim \int_0^t h^2(\tau) e^{\frac{\tau - t}{\tau_0}} d\tau \tag{2.9}$$

Wenn man sich den Anteil des Direktschalls zur Gesamtenergie, die am Ort des Hörers auftritt betrachtet, so ist bemerkbar, dass dieser nur 5 bis 15 Prozent beträgt. Es sind vor allem Anfangsreflexionen, die den größten Betrag zur Gesamtenergie leisten. Daraus lässt sich schließen, dass diese für das Nachhallempfinden verantwortlich sind. Außerdem ist zu beachten, dass diese vom Ort abhängig sind [GW14].

7. Diese ist typischerweise ohne Richtungsinformationen verfügbar, was zur Beschreibung des subjektiven Raumeindrucks nicht ausreichend ist.

2.2.1 Diffuse Schallfeld

Das Schallgeschehen in einem Raum lässt sich mit dem Befüllen eines undichten Gefäßes vergleichen. Beim Einschalten der Schallquelle beginnt sich der Raum allmählich mit Schallenergie zu *füllen* bis ein Gleichgewichtszustand (= eingeschwungener Zustand) erreicht ist. Der Zufluss ist gleich dem Abfluss. Dieser erklärt sich durch die Absorption (= Energieentzug) an den Raumbegrenzungsflächen. Schaltet man die Quelle nach dem eingeschwungen Zustand ab, so sinkt die Schallenergie. 3 Bereiche spielen hierbei eine Rolle [MC03]:

— Der Anhall ist jener Bereich indem sich das Schallfeld/ die Schallenergie aufbaut.
— Der stationärer Zustand ist gleich dem Gleichgewichtszustand.
— Der Nachhall ist jener Bereich, bei dem die Schallenergie langsam abnimmt.

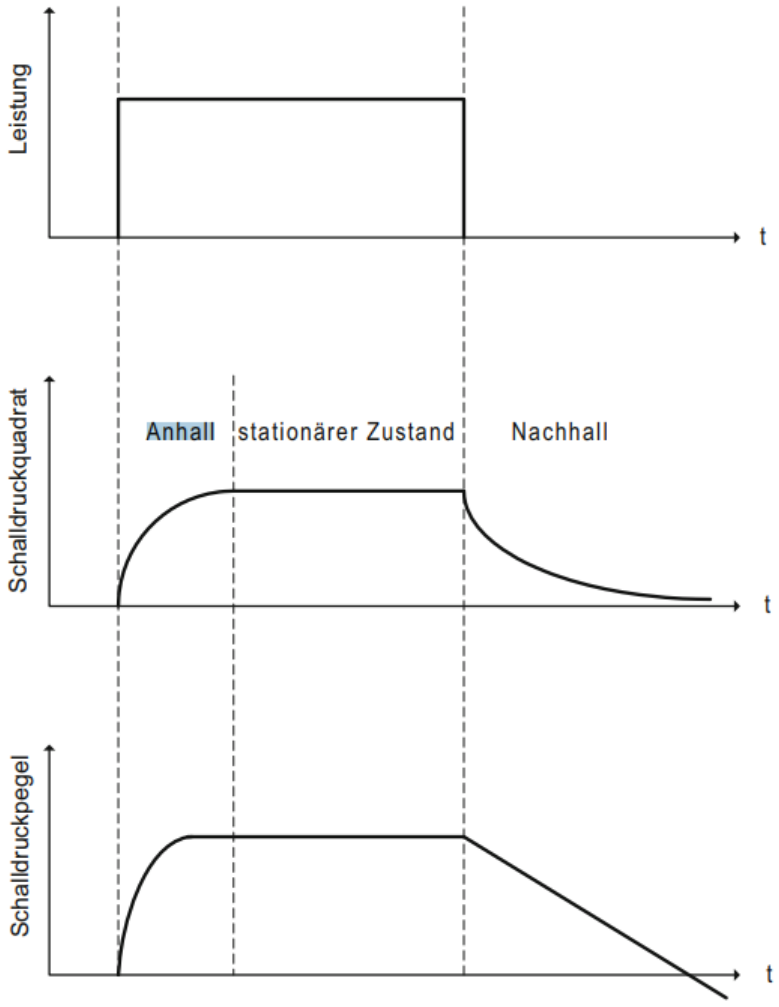

Abbildung 2 – Prinzipverlauf des diffusen Schallfeldes über die Teit t. Quelle: [Moe15]

Die Energiebilianz lässt sich auch in folgender Weise ausdrücken:

$$P \cdot \Delta t = V \cdot \Delta E + P_V \cdot \Delta t \tag{2.10}$$

Hierbei ist P die zugeführte Leistung, $P \cdot \Delta t$ die im Zeitintervall Δt zugeführte Energie E, ΔE die im Raum gespeicherte Energiedifferenz, P_V die Verlustleistung und $P_V \cdot \Delta t$ die abgegebene Energie. Darüber hinaus ist es sinnvoll anzunehmen, dass P_V proportional zu der im Volumen V enthaltenen Energie E ist [MC03]:

$$P_V \sim E \cdot V \tag{2.11}$$

Zusätzlich wird für P_V eine Verlust-Raumkonstante γ, die abhängig von der absorbierenden Fläche ist, berücksichtigt.

$$P_V = \gamma \cdot E \cdot V \tag{2.12}$$

Gleichung (2.11) bzw. (2.12) können nun in 2.10 zusammengefasst werden:

$$P \cdot \Delta t = V \cdot \Delta E + P_V \cdot \Delta t = V \cdot \Delta E + \gamma \cdot E \cdot V \cdot \Delta t \tag{2.13}$$

Aus dem Verhältnis von ΔE und Δt resultiert:

$$\frac{\Delta E}{\Delta t} = \frac{P}{V} - \gamma \cdot E \tag{2.14}$$

Falls ΔE bzw. Δt sehr klein sind, ergibt sich für Gleichung (2.14):

$$\frac{\mathrm{d}E}{\mathrm{d}t} = \frac{P}{V} - \gamma \cdot E \tag{2.15}$$

Die Energiedichte ist nur indirekt aus dem Schalldruck bestimmbar. Die Frage ist nun, wie Druck und Energiedichte miteinander verknüpft sind. Hierfür wird angenommen, dass, auf Grund des gleichmäßigen Schalleinfalls aus allen Raumrichtungen, die Schallschnelle im zeitlichen und örtlichen Mittel gegen Null konvergiert. Dies bedeutet, dass der Raum im Wesentlichen nur potentielle Energie speichert. Der Druck-Effektivwert $\tilde{p}^2$ kann durch

$$E = \frac{\tilde{p}^2}{\rho \cdot c^2} \tag{2.16}$$

ausgedrückt werden [MC03].

2.2.2 Nachhall

Schaltet man die Schallquelle nun aus, so nimmt die Schallenergie im Raum langsam ab. Die Dauer dieses Prozesses ist von γ abhängig. Ist γ groß, so erfolgt der Abklingvorgang rasch, andernfalls langsam. Für E in Abhängigkeit von $P = 0$ [MC03]:

$$\frac{\mathrm{d}E}{\mathrm{d}t} = -\gamma \cdot E \tag{2.17}$$

Die Lösung der obigen inhomogenen Differentialgleichung mit Anfangsbedingung $E(t = 0) = E_0$ ist durch:

$$E = E_0 \cdot e^{-\gamma \cdot t} \tag{2.18}$$

gegeben. Für den Effektivwertes des Schalldrucks gilt:

$$\tilde{p}^2(t) = \tilde{p}^2(0) \cdot e^{-\gamma \cdot t} \tag{2.19}$$

Der Schalldruckpegel $L(t)$ berechnet sich zu (vgl. [MC03]):

$$L(t) = 10 \cdot \log\left(\frac{\tilde{p}^2(t)}{\tilde{p}^2(0)}\right) = L(t = 0) - \gamma \cdot t \cdot 10 \cdot \log(e) \tag{2.20}$$

Aus Gleichung (2.20) ist ersichtlich, dass der Schalldruckpegel linear mit der Zeit abfällt. Bei ausreichend diffusen Schallfeldern kann dieser Pegel-Zeitverlauf in den Messergebnissen wiedergefunden werden, siehe Abbildung (2). Die Konstante γ kann im Allgemeinen nicht über die Pegel-Zeit-Gerade berechnet werden, da in der Praxis die Messkurven in den seltensten Fällen (annähernd) eine Gerade darstellen. Die Nachhallzeit ist definiert als jene Zeit T_{60}, welche benötigt wird, um einen Pegelabfall von 60db zu erreichen [MC03]. Es folgt:

$$60 = \gamma \cdot T \cdot 10 \cdot \log(e) \tag{2.21}$$

$$\gamma = \frac{60}{T \cdot 10 \cdot \log(e)} \tag{2.22}$$

In der Praxis bestimmt man nur jene Zeit T_{30}, die für einen Pegelabfall von 30db benötigt wird. Die Zeit T_{30} wird auch als halbe Nachhallzeit betrachtet. Eine direkte Bestimmung von T_{60} besitzt den Nachteil, dass im Vergleich zur Messung von T_{30} eine wesentlich besser Isolierung gegenüber Fremdgeräuschen gegeben sein muss [MC03].

2.2.3 Frühe Nachhallzeit

Für das Nachhallempfinden sind hauptsächlich Anfangsreflexionen verantwortlich. In einem Raum ist oftmals die vorhandene Dynamik unzureichend um große Pegelabfälle zu registrieren, da diese früher unter die Mithörschwelle fallen. Da der Anfang des Abklingvorganges besser wahrgenommen wird, hat man sich dazu entschlossen den -10dB Abfall zu extrapolieren. Wichtig ist hierbei, dass dieser ortsabhängig ist und sich nicht unbedingt zur Verallgemeinerung der frühen Nachhallzeit eignet [MC03].

2.3 Raumklangs Simulation

Dieser Abschnitt soll eine kleinen Überblick geben, auf welche Art und Weise eine Modellierung von Delays bzw. Nachhall möglich ist. Für weitere Informationen sei auf die entsprechende Fachliteratur verwiesen wie [Smi] oder [Zoe]. Die Gliederung diese Kapitels orientiert sich an [Smi]. Alle Bilder sind ebenso von dort entnommen. [8]

2.3.1 Verzögerungsglieder

Ein Verzögerungsglied (engl. Delay Element) ist der Grundbaustein vieler Audioeffekte wie Phaser, Flanger, Chorus oder künstlicher Nachhall. In der einfachsten Form stellt dieses eine Verzögerung zwischen Ein und Ausgang (uni-direktional) her.

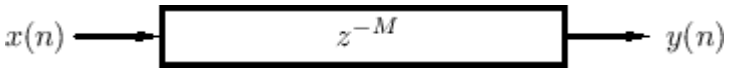

Abbildung 3 – Strukturbild Delay Line. Quelle: [Smi]

Mathematisch kann dies folgendermaßen formuliert werden:

$$Y(z) = X(z) \cdot z^{-M} \tag{2.23}$$

Wenn man mehrere solcher Verzögerungsglieder kaskadiert ergibt sich eine sogenannte Delay Line. Softwareseitig lässt sich diese mit einem Ring bzw. zirkulärer Buffer [9] implementieren [10]. Solche Delay Lines eignen sich hervorragend um die Ausbreitung von akustischen Wellen zu simulieren. Mittels zusätzlichen Filtern können frequenzabhängige Effekte wie Dissipation, Dispersion oder Absorption nach modelliert werden.

8. https://www.dsprelated.com/freebooks/pasp/Acoustic_Modeling_Digital_Delay.html
9. https://en.wikipedia.org/wiki/Circular_buffer
10. The Audio Programmer: https://bit.ly/2YUT9ux

2.3.2 Waveguides

Ein Waveguide ist eine bidirektionale Delay-Line, für die zusätzlich eine Wellenimpedanz definiert ist. Dadurch lassen sich im Gegensatz zur unidirektionale Delay-Line eindimensionale akustische Systeme, wie beispielsweise die Seite einer Gitarre modellieren .

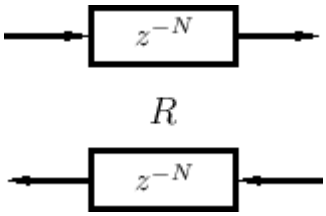

Abbildung 4 – Strukturbild Wave Guide. Quelle: [Smi]

2.3.3 Tapped Delay Line

Eine Tapped Delayline (kurz TDL) ist eine Erweiterung der Delayline mit einem sogenannten "Tap". Innerhalb der TDL wird das Signal abgezweigt, eventuell mit einem Skalierungsfaktor versehen und, anschließend mit den anderen verzögerten Signalen wieder aufsummiert (vgl. Abbildung 5).

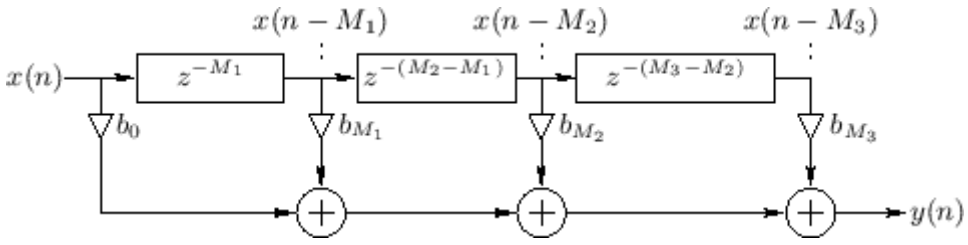

Abbildung 5 – Strukturbild Tapped Delay Line. Quelle: [Smi]

Im allgemeinen Fall hat eine TDL einen solchen *Tap* nach jedem Verzögerungsglied. Dadurch ergibt sich die Struktur eines kausalen FIR-Filters (Finite Impulse Response Filter, also ein Filter mit endlicher Impulsantwort, siehe Abbildung 6). Für zeitliche Indizes $n < 0$ wird angenommen, dass $x[n] = 0$ gilt.

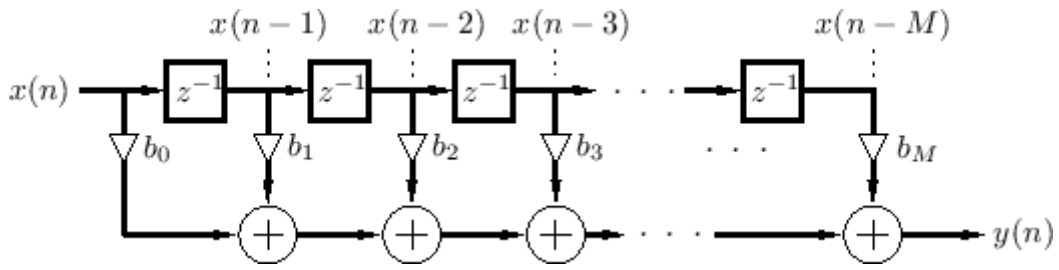

Abbildung 6 – Strukturbild Kausales FIR Filter. Quelle: [Smi]

Die Differenzengleichung einer solchen Filterstruktur lautet:

$$y[n] = \sum_{m=0}^{M} b_m \cdot x[n-m] \tag{2.24}$$

$$Y(z) = \sum_{m=0}^{M} b_m \cdot X(z) \cdot z^{-m} \tag{2.25}$$

$$\tag{2.26}$$

Für die Übertragungsfunktion gilt:

$$H(z) = \frac{Y(z)}{X(z)} = \sum_{m=0}^{M} b_m \cdot z^{-m} \tag{2.27}$$

2.3.4 Kammfilter

Kammfilter (engl. Comb Filter) bilden die Grundlage für die Echosimulation und vielen weiteren Audioanwendungen. Man unterscheidet zwischen:

— Kammfilter mit Vorwärtskopplung (engl. Feedforward Comb Filter) [11]
— Kammfilter mit Rückwärtskopplung (engl. Feedback Comb Filter) [12]

2.3.5 Vorwärts gekoppelter Kammfilter

Beim vorwärts gekoppelten Kammfilter kann das Ausgangssignal als Linearkombination von dem direkten und dem verzögerten Signal dargestellt werden. Die Differenzengleichung lautet:

$$y[n] = b_0 \cdot x[n] + b_M \cdot x[n-M] \tag{2.28}$$

Mit solch einem Filter kann man auf sehr einfache Art und Weise ein abklingendes Echo simulieren [Smi]. Der Verstärkungsfaktor $|b_M|$ ist hierbei immer negativ. Damit kann die Luftabsorption nachgebildet werden (Abbildung 7).

11. https://www.dsprelated.com/freebooks/pasp/Feedforward_Comb_Filters.html
12. https://www.dsprelated.com/freebooks/pasp/Feedback_Comb_Filters.html

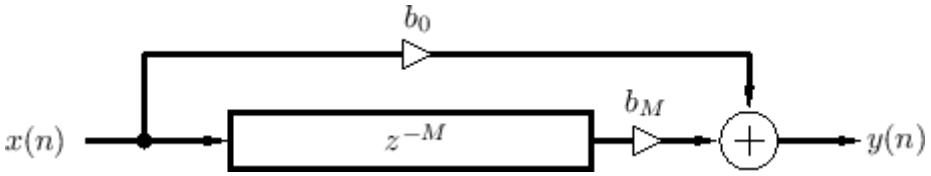

Abbildung 7 – Strukturbild Feedforward Comb Filter. Quelle: [Smi]

2.3.6 Rückwärts gekoppelter Kammfilter

Statt einem Vorwärtszweig wird bei einem rückwärts gekoppelten Kammfilter ein Rückkopplungszweig verwendet. Mathematisch kann dieser Filter durch:

$$y[n] = b_0 \cdot x[n] + a_M \cdot y[n - M] \tag{2.29}$$

modelliert werden (Abbildung 8). Wie man anhand der Differenzgleichung sehen kann handelt es sich bei einem rückwärts gekoppelten Kammfilter um ein System mit einer unendlich langen Impulsantwort (engl. für Infinite Impulse Response Filter, kurz IIR). Durch die Rückkopplung lassen sich somit eine mehrere abklingende Echos hintereinander modellieren.

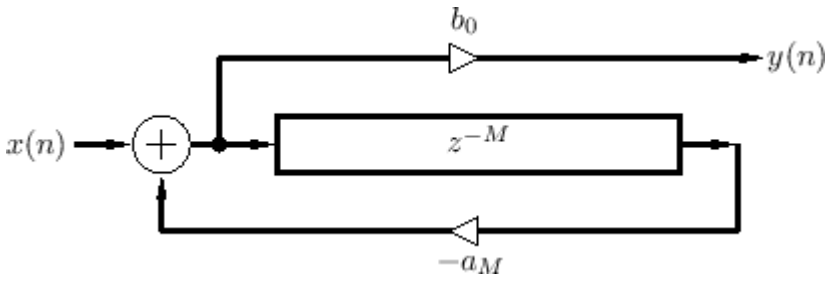

Abbildung 8 – Strukturbild Feedback Comb Filter. Quelle: [Smi]

2.3.7 Feedback Delay Networks

Feedback Delay Networks (kurz FDN) sind eine Erweiterung des Feedback Comb Filter. Im Gegensatz zu einer Delay Line wird eine Delay Matrix verwendet (siehe der nachfolgende Abschnitt)

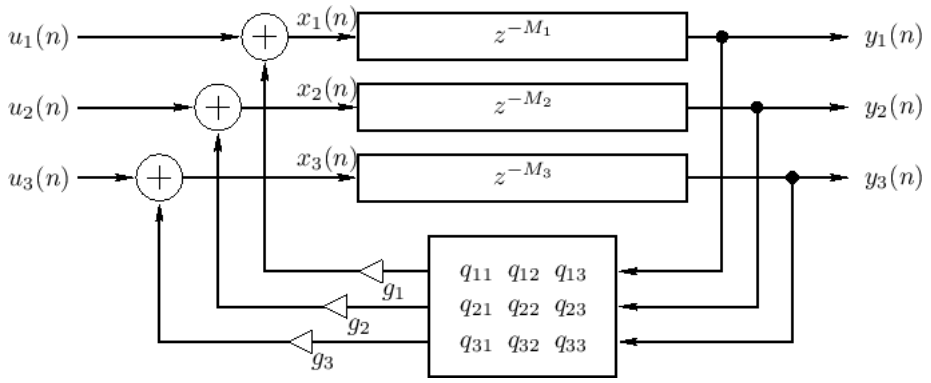

Abbildung 9 – Strukturbild Feedback Delay Network. Quelle: [Smi]

2.3.8 Allpass Filter

Bei der Allpass Filterstruktur wird ein vorwärts und ein rückwärts gekoppelter Kammfilter kaskadiert. Da dieser Filter eine besondere Rolle im Bereich der künstlichen Generierung von Nachhall spielt, wird im Folgenden näher auf diesen eingegangen. Ein System, mit Allpass Charakter, dämpft bzw. verstärkt alle Frequenzen gleich stark, verzögert aber diese unterschiedlich.

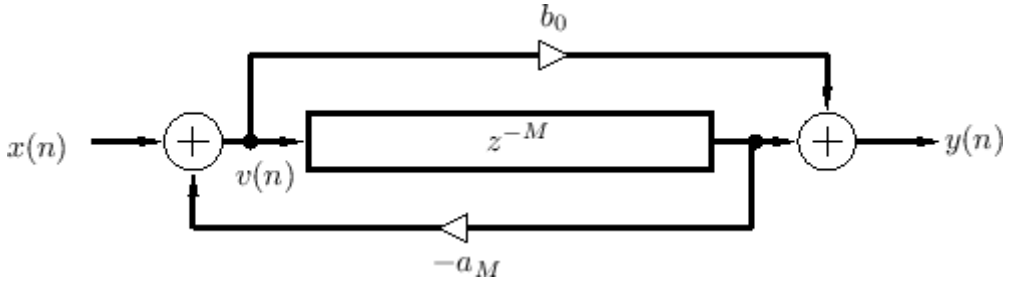

Abbildung 10 – Strukturbild Allpass. Quelle: [Smi]

Für die in Abbildung 10 dargestellte Allpassfilterstruktur kann man folgende Gleichungen an den beiden Summationspunkten angeben:

$$v[n] = x[n] - a_M \cdot v[n - M] \tag{2.30}$$
$$y[n] = b_0 \cdot v[n] + v[n - M] \tag{2.31}$$

Transformiert man (2.30) und (2.31) in den z-Bereich, so ergibt sich:

$$V(z) = X(z) - a_M \cdot z^{-M} \cdot V(z) \tag{2.32}$$
$$V(z) = \frac{X(z)}{1 + a_M \cdot z^{-M}} \tag{2.33}$$
$$Y(z) = b_0 \cdot V(z) + z^{-M} \cdot V(z) \tag{2.34}$$

$$Y(z) = b_0 \cdot \frac{X(z)}{1 + a_M \cdot z^{-M}} + z^{-M} \cdot \frac{X(z)}{1 + a_M \cdot z^{-M}} \tag{2.35}$$

$$Y(z) = X(z) \cdot \frac{b_0 + z^{-M}}{1 + a_M \cdot z^{-M}} \tag{2.36}$$

Die Übertragungsfunktion für die Allpassfilterstruktur lautet somit:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + z^{-M}}{1 + a_M \cdot z^{-M}} \tag{2.37}$$

Möchte man für diesen Filter das Frequenzverhalten analysieren, so wählt man $z = e^{j\omega T}$. Hierbei entspricht ω der Kreisfrequenz (Radiant pro Sekunde) und T der Abtastperiode (in Sekunden):

$$H(e^{j\omega T}) = \frac{Y(e^{j\omega T})}{X(e^{j\omega T})} = \frac{b_0 + e^{-j\omega MT}}{1 + a_M \cdot e^{-j\omega MT}} \tag{2.38}$$

$$\left|H(e^{j\omega T})\right| = \left|\frac{Y(e^{j\omega T})}{X(e^{j\omega T})}\right| = \left|\frac{b_0 + e^{-j\omega MT}}{1 + a_M \cdot e^{-j\omega MT}}\right| \tag{2.39}$$

Wenn $b_0 = a_M$ ist so folgt daraus:

$$\left|H(e^{j\omega T})\right| = \left|\frac{Y(e^{j\omega T})}{X(e^{j\omega T})}\right| = \left|\frac{b_0 + e^{-j\omega MT}}{1 + b_0 \cdot e^{-j\omega MT}}\right| = \left|\frac{b_0 + e^{-j\omega MT}}{1 + b_0 \cdot e^{-j\omega MT}}\right| = 1 \tag{2.40}$$

3 Faltungsalgorithmen

3.1 FIR Filter - Allgemein

Die einfachste Art einen Raumklang zu simulieren besteht darin, FIR bzw. Direktform Filter, also Filter mit endlicher Impulsantwort (engl. Finite Impulse Response) zu verwenden. Diese werden auch als sogenannte Tapped Delay Lines (kurz TDL) bezeichnet (siehe Abbildung 6). Dabei lässt sich jede Raumreflexion durch ein Verzögerungsglied modellieren. Diese Filter sind nicht nur in Audiosignalverarbeitung wichtig, sondern auch in der Systemtheorie für die Beschreibung linearer zeitinvarianter Systeme (abgekürzt LZI bzw. auf engl. linear time invariant systems, kurz LTI). Für die genaue Herleitung der Eigenschaften solcher System sei auf [OS14] verwiesen. Ein LZI System lässt sich durch seine Impulsantwort $h[n]$ wie folgt beschreiben:

$$y[n] = x[n] * h[n] = \sum_{n=-\infty}^{\infty} x[k]h[n-k] = \sum_{n=-\infty}^{\infty} x[n-k]h[n] \tag{3.1}$$

Das Ausgangssignal ist somit das Ergebnis der (diskreten) Faltung der Eingangsfolge $x[n]$ mit der Impulsantwort $h[n]$ des Systems. Man unterscheidet zwischen:

— Faltung von Eingangsfolgen endlicher Länge (Standardverfahren wenn die Länge der Eingangsfolge im vorhinein bereits bekannt ist)
— Faltung von Eingangsfolgen unbestimmter Länge.

Letzterer Fall ist vor allem für Audioanwendungen von Interesse, da im vorhinein nicht bekannt ist, wie lange $x[n]$ [13] sein wird. Für die Betrachtungen wird davon ausgegangen, dass sich die Länge von $h[n]$ nicht ändert. Weiteres wird angenommen, dass $x[n] = 0$ für $n < 0$ gilt. Mittels Matrix-Vektor Schreibweise [Wef14] kann Gleichung 3.1 in kompakter Form angeschrieben werden:

$$\mathbf{y} = \mathbf{H} \cdot \mathbf{x} \tag{3.2}$$
$$\mathbf{y} = (y_0, y_1, y_2, ..., y_{M-1})^T \tag{3.3}$$
$$\mathbf{x} = (x_0, x_1, x_2, ..., x_{N-1})^T \tag{3.4}$$

Die Sequenzen $\mathbf{x}$ und $\mathbf{y}$ müssen nicht unbedingt die gleichen Dimensionen haben. Im Folgenden besitzt $\mathbf{H}$ die Dimension $M \times M$ und eine Toeplitz-Struktur [14]. Des Weiteren

13. Streng genommen entspricht $x[n]$ einem Element des Signals bzw. Signalvektors $\mathbf{x}$. Im Folgenden werden jedoch beide Ausdrücke gleichwertig verwendet
14. https://en.wikipedia.org/wiki/Toeplitz_matrix

wird angenommen, dass $h[n] = 0$ für $n < 0$ gilt (kausale Impulsantwort).

$$\mathbf{H} = \begin{bmatrix} h_0 & 0 & 0 & \cdots & 0 & 0 \\ h_1 & h_0 & 0 & \cdots & 0 & 0 \\ h_2 & h_1 & \ddots & \ddots & \ddots & 0 \\ h_3 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ h_{M-1} & \cdots & h_3 & h_2 & h_1 & h_0 \end{bmatrix} \tag{3.5}$$

Eine weitere Möglichkeit (3.1) anzuschreiben, wäre als ein Polynom der Form

$$X(z) = \sum_{n=0}^{N-1} x[n] z^{-n} \tag{3.6}$$

$$H(z) = \sum_{n=0}^{N-1} h[n] z^{-n} \tag{3.7}$$

$$Y(z) = X(z) \cdot H(z) \Leftrightarrow y[n] = x[n] * h[n] \tag{3.8}$$

Hierbei handelt es sich um die z-Transformierte der Eingangs/Ausgangsfolge bzw. der Impulsantwort [OS14]. Transformiert man diese , so lässt sich die Faltung im Zeitbereich als eine Multiplikation im Frequenzbereich anschreiben (Faltungstheorem).

3.1.1 Komplexität - FIR

Anhand von Gleichung (3.1) kann nachvollzogen werden, dass N Multiplikationen und $N - 1$ Additionen für die Berechnung von einem Ausgangswert benötigt werden. Die Gesamtanzahl aller Operationen für ein Ausgangswert beträgt:

$$T_{ein}(N) = N + (N - 1) = 2N - 1 \in \mathcal{O}(N) \tag{3.9}$$

Für N Ausgangswerte ergibt sich:

$$T_{alle}(N) = N^2 + N(N - 1) = 2N^2 - N \in \mathcal{O}(N^2) \tag{3.10}$$

Somit gehört die Faltungssumme zu jener Klasse von Algorithmen, deren Komplexität quadratisch ansteigt. Dieser Algorithmus eignet sich somit besser für kleine Filterlängen, da für sehr große N eine Vielzahl von Rechenoperationen benötigt werden. Dies kann man anhand des Folgenden Beispiels zeigen [Gar95]: Typische Raumimpulsantworten haben Längen von 1 bis 2 Sekunden. Für eine zweisekündige Impulsantwort benötigt man, bei einer Abtastrate von 44.1kHz, 88.200 Filter Koeffizienten. Der wesentliche Vorteil eines solchen Filters ist die minimale Latenz von einem Sample (sobald der Filter eingeschwungen ist) zwischen Ein- und Ausgang.

3.1.2 Sparse-FIR Filter

Bei Sparse FIR-Filtern werden Multiplikationen mit 0 nicht berücksichtigt. Das einzige was sich ändert ist, an welcher Stelle sich die Verzögerungsglieder in der TDL befinden. Im Falle des RoomEncoders bleibt die Anzahl der simulierten Reflexionen unabhängig unabhängig von der Länge der Nachhallzeit des virtuellen Raumes.

3.2 FFT - Allgemein

Wie bereits geschildert, besitzen FIR Filter den Nachteil, dass diese, mit zunehmender Filterlänge, sehr rechenintensiv werden. Dies ist mit der Komplexität von $\mathcal{O}(N^2)$ zu begründen. Für lange Filterlängen eignet sich die Diskrete Fouriertransformation (DFT) bzw. Schnelle Fouriertransformation (engl. Fast Fourier Transform, kurz FFT) besonders gut. Man transformiert hierbei vom Zeitbereich in den Frequenzbereich. Mit dieser Methode lassen sich z.B. Faltungen in eine Multiplikation überführen (siehe Abschnitt 3.1 besonders Gleichung 3.1). Durch die blockweise Verarbeitung, sowie durch das Ausnutzen bestimmter Transformationseigenschaften (siehe [OS14]) kann bei der Berechnung an Effizienz gewonnen werden. Für die folgenden Betrachtungen wird die Annahme getroffen, dass für die Länge N des Eingangssignals gilt: $N = 2^k$.

3.2.1 Decimation In Time Algorithmus - Herleitung Teil 1

Sei die Sequenz x mit Länge $N = 2^k$ und $x[n] < 0$ für $n < 0$ gegeben. Die Eingangsfolge ist zerlegbar in 2 Teilfolgen mit einer Länge von jeweils $\frac{N}{2}$. Die erste Teilfolge g soll hierbei alle geraden Folgenglieder und die zweite Teilfolge h soll alle ungeraden Folgeglieder enthalten. Mathematisch formuliert bedeutet dies [OS14]:

$$g[n] := x[2n] \tag{3.11}$$
$$h[n] := x[2n+1] \tag{3.12}$$

Für die DFT beider Signal folgt somit :

$$G[k] = \sum_{n=0}^{N/2-1} g[n] W_{N/2}^{kn} = \sum_{n=0}^{N/2-1} x[2n] W_{N/2}^{kn}, \qquad k = 0, 1, 2, ..., N/2-1 \tag{3.13}$$
$$H[k] = \sum_{n=0}^{N/2-1} h[n] W_{N/2}^{kn} = \sum_{n=0}^{N/2-1} x[2n+1] W_{N/2}^{kn}, \qquad k = 0, 1, 2, ..., N/2-1 \tag{3.14}$$
$$W_{N/2} = e^{-j\frac{2\pi}{\frac{N}{2}}kn} \tag{3.15}$$

Für den ungeraden Anteil ist die Eingangsfolge zeitlich um einen Wert verschoben. Für $X[k]$ ergibt sich:

$$X[k] = G[k] + W_N^k H[k], \; k = 0, 1, 2, ..., N-1 \tag{3.16}$$

Es sei hier angemerkt, dass $X[k]$ für $k = 0, 1, 2, ..., N-1$ berechnet wird, obwohl $G[k]$ und $H[k]$ nur für $k = 0, 1, 2, ..., N/2-1$ definiert sind. Dieses Problem kann gelöst werden, indem $G[k]$ und $H[k]$ periodisch (Periodenlänge N/2) auf die Menge {0,...,N-1} fortgesetzt wird. Gleichung (3.16) kann auch folgendermaßen ausgedrückt werden [OS14]:

$$X[k] = G[\text{k mod } N/2] + W_N^k H[\text{k mod } N/2],\ k = 0, 1, 2, ..., N-1 \tag{3.17}$$

Nachdem die zwei kürzeren Teilfolgen berechnet worden, können diese aufsummiert werden, um wieder die ursprüngliche Eingangsfolge $X[k]$ zu erhalten. Graphisch lässt sich dies durch ein sogenanntes *Schmetterlingsdiagramm* (siehe Abbildung 11) darstellen. Hierbei handelt es sich um den Signalflussgraphen des oben geschilderten Algorithmus [OS14].

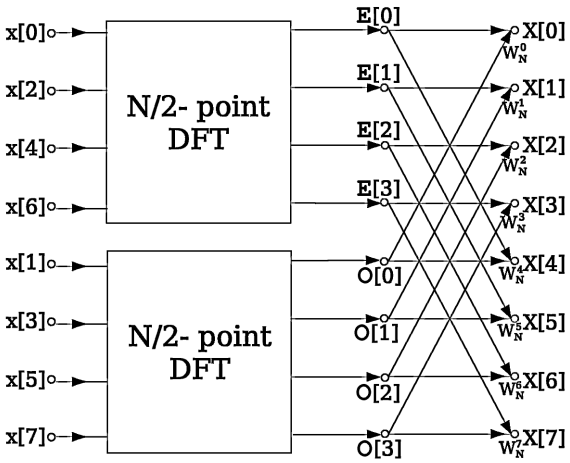

Abbildung 11 – Schmetterlingsgraph. Quelle: https://en.wikipedia.org/wiki/Butterfly_diagram

3.2.2 Decimation in Time Algorithmus - Herleitung Teil 2

Verglichen mit der Gesamtzahl der Rechenoperationen bei der Faltungssumme (siehe Gleichung (3.1)) ergeben sich für den Decimation In Time Algorithmus $2 \cdot \left(\frac{N}{2}\right)^2$ Multiplikationen und $2 \cdot \left(\frac{N-1}{2}\right) \cdot N$ Additionen. Diese Ausdrücke rühren daher, dass man die Eingangsfolge der Länge N in zwei Teilfolgen der Länge $\frac{N}{2}$ zerlegt. Betrachtet man den geraden und ungeraden Teil gemeinsam, so erhält man für einen *einzelnen* Ausgangswert $2 \cdot \frac{N}{2}$ Multiplikationen. Da aber die beiden Teilfolgen nur noch mehr halb so lang sind wie die ursprüngliche sind, wird nicht über N sondern über $\frac{N}{2}$ Werte aufsummiert. Analog verhält es sich mit der Addition. Bei der Betrachtung der Komplexität kann man für große N folgende Vereinfachungen durchführen [OS14]:

Bis jetzt wurde angenommen, dass die Länge von $N = 2^k$ ist. Aus dieser Annahme folgt, dass auch $N/2 \in \mathbb{N}_g$ (also Element aller gerade natürlichen Zahlen) ist. Dementsprechend

kann man auch die beiden $N/2$ Folgen ($G[k]$ und $H[k]$) in zwei weitere Sequenzen der Länge $N/4$ unterteilen [OS14].

$$G[k] = \sum_{n=0}^{N/2-1} g[n]W_{N/2}^{kn} = \sum_{n=0}^{N/4-1} g[2m]W_{N/2}^{2km} + \sum_{n=0}^{N/4-1} g[2m+1]W_{N/2}^{(2m+1)k} \tag{3.18}$$

$$= \sum_{n=0}^{N/4-1} g[2m]W_{N/4}^{km} + W_{N/2}^{k} \sum_{n=0}^{N/4-1} g[2m+1]W_{N/4}^{km} \tag{3.19}$$

$$H[k] = \sum_{n=0}^{N/4-1} h[2m]W_{N/4}^{km} + W_{N/2}^{k} \sum_{n=0}^{N/4-1} h[2m+1]W_{N/4}^{km} \tag{3.20}$$

Dieser Prozess des Aufteilens in kleine Folgen kann man z.B. rekursiv solange wiederholen, bis nur noch 2 Punkt DFTs übrig sind. In jedem Rechenschritt setzt man also diese kleineren Blöcke bzw. Frames in größere zusammen, bis man zum Schluss die Originale DFT von $X[k]$ erhält. Da die Länge $N = 2^k$ angenommen worden ist lässt sich die Anzahl der Iterationen für **einen Wert** sehr einfach berechnen [OS14]:

$$N = 2^k \Rightarrow k = \log_2(N) \tag{3.21}$$

Für N Samples folgt:

$$N \cdot k = N \cdot \log_2(N) \tag{3.22}$$

Die Komplexität der FFT (die mit Hilfe des Decimation in Time Algorithmus berechnet werden kann), ergibt sich zu $\mathcal{O}(N \cdot \log_2(N))$ berechnet werden. Im vorangegangen Beispiel wurde aber nur die Anzahl der nötigen Rechenoperationen für die eigentliche Transformation hergeleitet. Nicht beachtet hingegen wurden [OS14]:

— Transformation des Eingangsblöcke $x[n]$ und Impulsantwort $h[n]$,
— Multiplikation von $X[k]$ und $H[k]$. Man erhaltet somit $Y[k]$,
— Rücktransformation von $Y[k]$ [15],
— Zusammenfassen der Ausgangsblöcke im Zeitbereich,
— Speicherbedarf.

15. Sowohl die Hin als auch Rücktransformation benötigt $\mathcal{O}(N \cdot \log_2(N))$ Operationen

3.3 Blockweise Faltung

Wie am Anfang des Kapitels geschildert liegt ein großer Vorteil der FFT darin, dass durch die blockweise Verarbeitung des Eingangssignals $x[n]$ die Anzahl an Operationen erheblich verringert werden kann. Ein Nachteil ist jedoch, dass bevor die entsprechenden Berechnungen durchgeführt werden können, eine entsprechende Anzahl von Samples im Ein- bzw. Ausgangspuffer vorhanden sein müssen. Dadurch ergibt sich eine entsprechende Verzögerung zwischen Ein- und Ausgang. Um diesen Nachteil auszugleichen, wird das Eingangssignal $x[n]$ bzw. die Impulsantwort $h[n]$ in Blöcke mit gleicher Länge L unterteilt [OS14].

Im Folgenden wird die Annahme getroffen, dass die Impulsantwort eine konstante und die Eingangsfolge eine unbestimmte Länge aufweist. Jeder Block der Eingangssequenz wird dabei mit der Impulsantwort gefaltet [OS14].

Für ein LZI System gilt:

$$y[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k] \tag{3.23}$$

$$y[n] = 0, \text{ für } \forall n < 0 \text{ und } n > \underbrace{(L+P-1)}_{=N} - 1 \tag{3.24}$$

Die Folge $x[n]$ hat die Länge L, $h[n]$ die Länge P. Bei N handelt es sich um die maximale Länge der resultierenden Ausgangsfolge $y[n]$. Durch die entsprechende Wahl von $N = L + P - 1$ wird zeitliches Aliasing vermieden [OS14]. Die DFT von Gleichung (3.23) lautet:

$$Y[k] = X[k] \cdot H[k] \tag{3.25}$$

Wobei $Y[k]$ N Samples lang sein sollte. Dies erreicht man, indem man die Eingangssequenz und die Impulsantwort *zero-padded*.

3.3.1 Overlap-Add

Zunächst wird ein Eingangsblock $x_i[n]$ der Länge L auf die Länge N mit Nullen erweitert [16]. Anschließend wird dessen FFT $X_i[k]$ berechnet. In gleicher Weise wird mit der Impulsantwort $h[n]$ der Länge P verfahren. Man erhält $H[k]$ der Länge N. Durch die Multiplikation von $X_i[k]$ mit $H[k]$ (Faltungstheorem) resultiert $Y_i[k]$. Das Ergebnis wird zurück transformiert ($y_i[n]$) und hat die Länge N, wobei maximal $L + P - 1$ Ergebniswerte ungleich Null sind. Jeder einzelne Ausgangsblock $y_i[n]$ überlappt sich mit dem nächsten Block um $P - 1$ Werte die zum Schluss alle aufsummiert werden um $y[n]$ zu erhalten [OS14]. Der Algorithmus ist in Abbildung 12 schematisch dargestellt.

$$y[n] = \sum_{i=0}^{M} y_i[n] \tag{3.26}$$

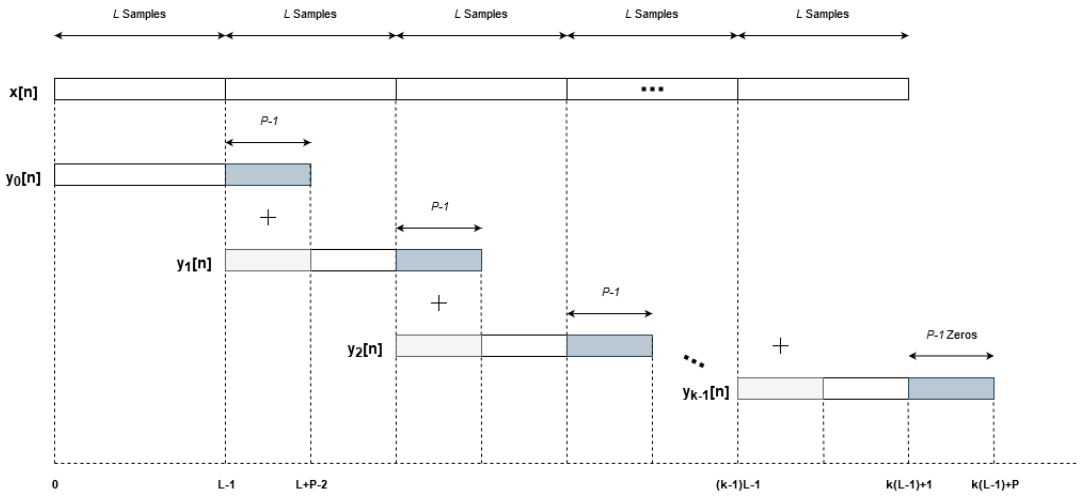

Abbildung 12 – Schematische Darstellung OLA-Algorithmus

16. Dieser Vorgang wird auch als *zero-padding* bezeichnet

3.3.2 Overlap-Save

Wie bei der Overlap-Add Methode wird zunächst die Eingangsfolge $x[n]$ in Segmente $x_i[n]$ der Länge L zerlegt und dann mit der Impulsantwort $h[n]$ zirkulär gefalten. Aufgrund der Faltung von Sequenzen der Länge L mit Sequenzen der Länge P folgt, dass die ersten P-1 Elemente des entsprechenden Anfangsblocks $y_i[n]$ auf Grund von zeitlichen Aliasing inkorrekt sind. Die restlichen Samples werden jedoch richtig berechnet und entsprechen dem Ergebnis der linearen Faltung [OS14]. Der Algorithmus ist in Abbildung 13 schematisch dargestellt.

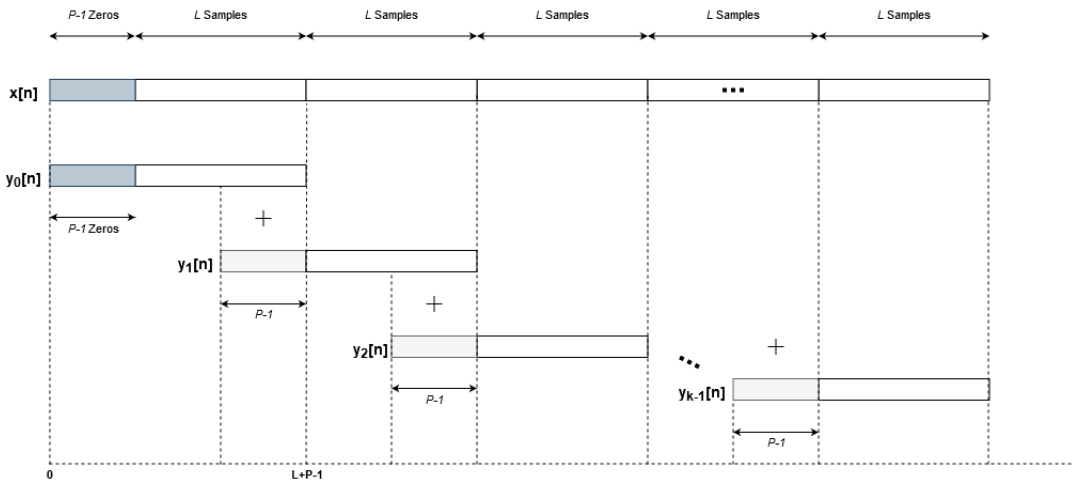

Abbildung 13 – Schematische Darstellung OLS-Algorithmus

3.3.3 Berechnung der Komplexität

Für einen einzelnen Eingangswert ergibt sich [Wef14]:

$$T_{\text{OLS-FFT}}(L,P,N) := \frac{1}{L} \cdot [T_{\text{FFT}}(N) + T_{\text{CMUL}}(N) + T_{\text{IFFT}}(N)] \tag{3.27}$$

$$T_{\text{OLA-FFT}}(L,P,N) := \frac{1}{L} \cdot [T_{\text{FFT}}(N) + T_{\text{CMUL}}(N) + T_{\text{IFFT}}(N) + T_{\text{ADD}}(L+P-1)] \tag{3.28}$$

Für alle N Punkte ergibt sich:

$$T_{\text{All OLS-FFT}}(L,P,N) = T_{\text{All OLA-FFT}}(L,P,N) = T_{\text{FFT}}(N) \tag{3.29}$$

Hierbei gehören die Operationen ADD, CMUL (komplexe Multiplikation) und CMAC (komplexe Multiplikation und Addition) zur Klasse jener Algorithmen, die eine lineare Laufzeit aufweisen, also $\mathcal{O}(N)$ gilt [Wef14].

Für die Komplexität der einzelnen Rechenoperationen ergibt sich im Folgenden:

$$T_{\text{CMUL}}(N) = 6N \tag{3.30}$$
$$T_{\text{ADD}}(N) = T_{\text{MUL}}(N) = N \tag{3.31}$$
$$T_{\text{CADD}}(N) = 2 \cdot T_{\text{ADD}}(N) = 2N \tag{3.32}$$
$$T_{\text{CMAC}}(N) = T_{\text{CMUL}}(N) + T_{\text{CADD}}(N) = 8N \tag{3.33}$$

Sowohl Overlap-Add und Overlap-Save gehören zur Familie jener Algorithmen, die mit einer Komplexität von $\mathcal{O}(N\log_2 N)$ arbeiten. Zusammenfassend lässt sich schließen, dass das OLA bzw. OLS Verfahren bei zunehmender Blocklänge deutlich effizienter arbeitet. Durch Partitionierung der Impulsantwort und dem Wiederverwenden von bereits berechneten Spektren kann deren Effizienz noch weiter gesteigert werden [Wef14].

3.4 Gleichmäßig Partitionierte Faltung

Bei der blockweisen Faltung (siehe Kapitel 3.3) wird nur das Eingangssignal in Blöcke gleicher Länge unterteilt. Gleichmäßig partitionierte Faltungsalgorithmen (engl. uniformly partitioned convolution) unterteilen das Eingangssignal und die Impulsantwort des Filters in gleich lange Partitionen $x_i[n]$ und $h_i[n]$ [17]. Die daraus resultierenden Teilfilter können mit schnellen Faltungsalgorithmen (engl. Fast Convolution) wie der DFT/FFT implementiert werden. Im Vergleich zu Faltungsalgorithmen, die eine solche Partitionierung der Impulsantwort nicht vornehmen, ist bemerkbar, dass bei zunehmender Filterlänge die Berechnung ineffizienter wird. Der Grund hierfür ist, dass ein Eingangsblock mit mehr Nullen aufgefüllt werden muss [Wef14].

3.4.1 Motivation

Sei nun P [18] die Länge des Filters und L die Länge des Eingangsblocks. Um eine N-Punkt DFT zu berechnen, müssen beide Folgen mit Nullen aufgefüllt werden, bis die Länge N erreicht ist. Dies dient zur Vermeidung von zeitlichen Aliasing [19]. Für kleine P hält sich das Padding in Grenzen, man kann den Eingangsblock ohne gröbere Latenz mit der Impulsantwort falten. Für große P ist diese nicht mehr zu vernachlässigen, schließlich soll die Faltung echtzeitfähig sein. Durch die Partitionierung des Filters kann also eine lange Impulsantwort in kleinere unterteilt werden. Um die Berechnung effizienter zu machen, können bereits berechnete Spektren der Eingangsblöcken für alle Teilfilter wieder verwendet werden. Dadurch lässt sich die Anzahl an Hin und Rücktransformationen reduzieren. Damit die Berechnung nur im Frequenzbereich erfolgen und somit die Effizienz der FFT am besten ausgenutzt werden kann, muss für L und N folgende Bedingungen gelten [Wef14]:

— Sei P die Länge von $h_i[n]$ und L die Länge von $x_i[n]$. Dann gelte:

$$P = 2 \cdot L \tag{3.34}$$

— Sei $\mathcal{P}$ die Anzahl der Filterpartitionen:

$$\mathcal{P} = \lceil \frac{L}{P} \rceil \tag{3.35}$$

Zu beachten ist, dass bei der Aufsummierung der Subfilter sich die ursprüngliche Filterantwort ergeben sollte. Außerdem wird jede Partition um einen Offset $M = n \cdot P$ mit $n = 0, \cdots, \mathcal{P} - 1$ versetzt [Wef14].

17. Es kann durchaus der Fall sein, dass die einzelnen Blöcke des Eingangssignal und der Impulsantwort nicht die gleiche Auflösung besitzen. Die Länge von $x_i[n]$ muss nicht gleich der Länge von $h_i[n]$ sein. Für alle folgenden Betrachtungen wird zur Vereinfachung jedoch hiervor ausgegangen.
18. Um die Abkürzungen im Text konsistent zu halten, wurde eine zu [Wef14] abweichende Notation gewählt. Die Länge einer Filterpartition B in Abbildung 14 entspricht hier P, die Länge eines Eingangsblock N wird mit L bezeichnet und die Anzahl der Filterpartitionen wird hier mit $\mathcal{P}$ statt P angeschrieben.
19. Siehe [OS14], Kapitel 8 "The Discrete Fourier Transform", S.688 - 700

3.4.2 Allgemein

Die folgenden Betrachtungen gehen von einer Uniformly Partitioned Convolution aus, die mittels eines FFT basierten OLS Faltungsalgorithmus implementiert worden ist. Hierbei wird das Filter in Segmente gleicher Länge unterteilt, die DFT/FFT berechnet und mittels OLS wieder rekombiniert. In Abbildung 14 ist der Algorithmus schematisch dargestellt. Zunächst wird das Eingangssignal in gleich lange Blöcke der Länge L unterteilt und mit einer P langen Impulsantwort gefiltert. Die Anzahl der Teilfilter kann mittels Gleichung (3.34) bzw. (3.35) berechnet werden. Abbildung 14 zeigt eine Variante des Algorithmus, der reellwertige Eingangsdaten mittels einer R2C-(= Real to Complex, deutsch "Real zu Komplex") Transformation verarbeitet und sich symmetrische DFT Spektren zunutze macht [Wef14].

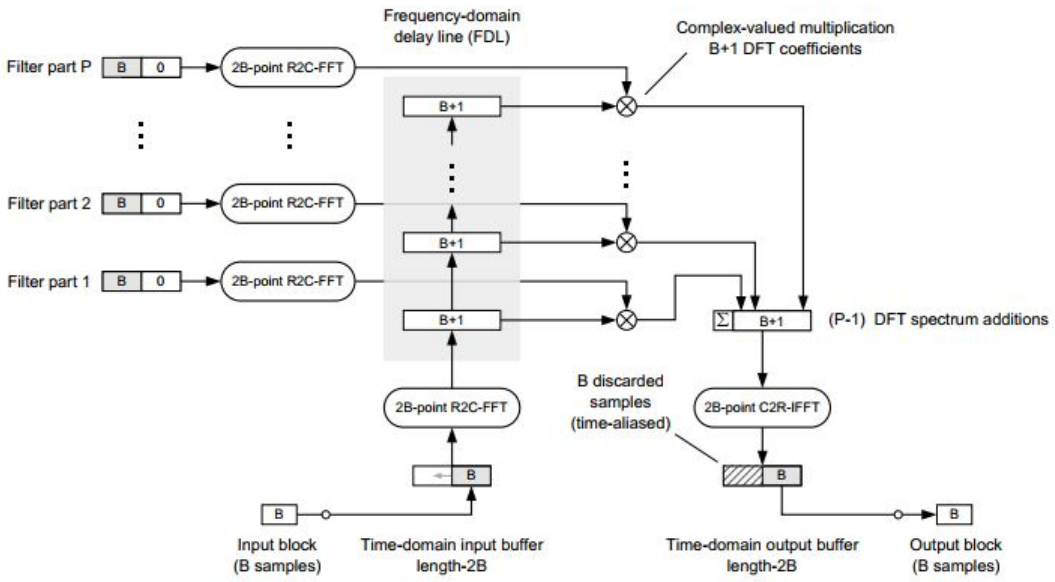

Abbildung 14 – Schematische Darstellung des Uniformly Partition Overlap Save Algorithmus (UPOLS) mit uniform partitionierten Filter. Quelle: [Wef14]

Transformationsgröße: Dieser Standard Algorithmus verwendet eine feste Transformationsgröße $N = 2 \cdot L$. Hierdurch lässt sich zeitliches Aliasing vermeiden [OS14]. Wählt man für die Verzögerung der Teilfilter zusätzlich $P = L$, so lässt sich diese ebenso wie Akkumulation zur Gänze im Frequenzbereich realisieren [Wef14].

Verzögerung und Akkumulation im Frequenzbereich: Sowohl das Eingangssignal, als auch die Filterpartitionen verwenden dieselbe Auflösung $L = P$. Wie bereits erwähnt, lassen sich hiermit die einzelnen Verzögerungen der Teilfilter im Frequenzbereich realisieren. Von jeden Eingangsblock wird die DFT nur ein einziges Mal berechnet. Die jeweiligen Spektren können dann für alle Filterpartitionen wiederverwendet und die Teilfilterergebnisse im Frequenzbereich aufsummiert werden. Die Folge hiervon ist, dass zum Schluss nur noch eine Rücktransformation berechnet werden muss, um einen Ausgangsblock zu erhalten. Somit kann Rechenleistung gespart werden [Wef14]. Die Realisierung der Teilfilter Verzögerungen im Frequenzbereich wird auch als *frequency-domain delay line*, kurz FDL bezeichnet. Im Grund genommen handelt es sich um ein Shift-Register für Spektren [Wef14]

Filter Verarbeitung

— Zunächst wird das Filter nach Gleichung 3.35, in gleich lange Partitionen der Länge P unterteilt.
— Danach wird jeder Teilfilter auf die Länge $2 \cdot P$ [20] mit Nullen aufgefüllt und in den Frequenzbereich transformiert.

Verarbeitung

— Der Eingangsbuffer arbeitet wie ein $2L$ langes gleitendes Fenster. Mit jedem Eingangsblock wird die rechte Hälfte des Buffers nach links geschoben. Ein neuer Block wird anschließend an die ursprüngliche Stelle gespeichert.
— Alle DFT Spektren in der FDL werden um eine Position verschoben.
— Eine $2L$ R2C FFT wird nun vom Eingangsbuffer berechnet. Man erhaltet $L + 1$ DFT Koeffizienten. Das Ergebnis wird nun im ersten FDL Slot gespeichert.
— Die Spektren der Teilfilter werden paarweise mit den Eingangsspektren multipliziert und im Frequenzbereich akkumuliert.
— Von den akkumulierten Spektren wird anschließend eine $2L$ C2R-IFFT berechnet. Die linke Hälfte des $2L$-langen Ausgangsblocks kann verworfen werden (zeitliches Aliasing), die rechte Hälfte wird als nächster Ausgangsblock retourniert.

20. siehe Kapitel 3.4.1

3.4.3 Berechnung der Komplexität

Für rein reelle Signale gilt für die Komplexität des Algorithmus [Wef14]:

$$T_{\text{UPConv}}\left(P, L\right) := \mathcal{P} \cdot T_{\text{FFT-R2C}}\left(2P\right) = \lceil \frac{L}{P} \rceil \cdot T_{\text{FFT-R2C}}\left(2P\right) \tag{3.36}$$

Durch einsetzen von Gleichung (3.22) in (3.36) mit einem Skalierungsfaktor k und konstantem B ergibt sich: [21]

$$T_{\text{UPConv}}\left(P = \text{konstant}, L\right) = \lceil \frac{L}{P} \rceil \cdot k \cdot 2P \cdot \underbrace{\log_2\left(2P\right)}_{\text{konstant}} \tag{3.37}$$

$$T_{\text{UPConv}}\left(P = \text{konstant}, L\right) = 2kL + \mathcal{O}(1) \tag{3.38}$$

Somit gilt $T_{\text{UPConv}}\left(P = \text{konstant}, L\right) \in \mathcal{O}(N)$. Die Komplexität für einen Ausgangswert kann abgeschätzt werden mit:

$$T_{\text{UPConv}}\left(P, L\right) = \frac{1}{P} \cdot [T_{\text{FFT-R2C}}\left(2P\right) + T_{\text{IFFT-R2C}}\left(2P\right) + T_{\text{CMUL}}(P+1) + (\mathcal{P}-1) \cdot T_{\text{CMAC}}(P+1)] \tag{3.39}$$

$$T_{\text{UPConv}}\left(P, L\right) = \frac{1}{P} \cdot [2kP\log_2\left(2P\right) + 2kP\log_2\left(2P\right) + 6(P+1) + 8(\mathcal{P}-1)(P+1)] \tag{3.40}$$

$$T_{\text{UPConv}}\left(P, L\right) = \frac{1}{P} \cdot \left[2kP\log_2\left(2P\right) + 2kP\log_2\left(2P\right) + 6(P+1) + 8\left(\lceil \frac{L}{P} \rceil - 1\right) \cdot (P+1)\right] \tag{3.41}$$

Um die Berechnung noch einfacher zu gestalten, werden die folgenden Annahmen getroffen:

$$P + 1 \approx P \tag{3.42}$$

$$\lceil \frac{L}{P} \rceil - 1 \approx \frac{L}{P} \tag{3.43}$$

Einsetzung in Gleichung (3.41) ergibt:

21. Der Grund, warum aus dem multiplikativen Faktor in Gleichung (3.37) bzw. (3.38) ein additiver Term wird liegt daran, dass dieser konstant ist. Die Komplexität des Algorithmus kann demnach mit $\mathcal{O}(1)$ abgeschätzt werden. Für nähere Informationen zur $\mathcal{O}$-Notation sei auf [CLRS09] (insbesondere Kapitel 3) und auf https://www.algorithm-archive.org/contents/notation/notation.html verwiesen

$$T_{\text{UPConv}}(P, L) = \frac{1}{P} \cdot \left[4kP\,(1 + \log_2 P) + 6P + 8P\lceil \frac{L}{P} \rceil \right] \tag{3.44}$$

$$T_{\text{UPConv}}(P, L) = 4k + 4k\log_2 P + 8\frac{L}{P} + 6 \tag{3.45}$$

$$T_{\text{UPConv}}(P, L) = 4k\log_2 P + 8\frac{L}{P} + \mathcal{O}(1) \tag{3.46}$$

Man erkennt, dass $T_{\text{UPConv}}(P = \text{konstant}, L) \in \mathcal{O}(L)$ gilt.

3.5 Nicht Gleichmäßig Partitionierte Faltung

FIR-Filter als auch FFT basierte Faltungsalgorithmen haben ihre Vor- und Nachteile. Erste eignen sich besonders für kurze Impulsantworten und weisen eine geringe Latenz zwischen Ein- und Ausgang auf. Hierfür steigt deren Komplexität bei zunehmender Filterlänge mit $\mathcal{O}(N^2)$ an. Die auf der FFT basierenden Blocktransformationen, wie das OLA oder OLS Verfahren, können hier ihre Stärken ausspielen, weisen jedoch eine erhebliche Verzögerung zwischen dem Ein- und Ausgang auf. Für zeit-unkritische Berechnungen stellt dies kein Problem dar, für Echtzeitanwendungen ist dies nicht erwünscht. Gardner [Gar95] beschreibt in seinem Paper eine Hybridlösung, die sich die Effizienz der FFT bei großen Blockgrößen, als auch die geringe Latenz des FIR Filter zunutze macht. Dieser Algorithmus wird als Zero-Delay Convolution bezeichnet. Hierbei wird die Impulsantwort des Filters $h[n]$ in Partitionen unterschiedlicher Größe unterteilt. Die Blockgröße ist so gewählt, dass die Komplexität pro Ausgangssample und die Verzögerung zwischen Ein- und Ausgang möglichst minimiert wird. Bei diesem Algorithmus handelt es sich um einen Non-Uniforme Partitionierte Faltung (engl. non-uniformly partitioned convolution, im Nachfolgenden mit NUP abgekürzt). In Abbildung 15 ist der Algorithmus schematisch dargestellt.

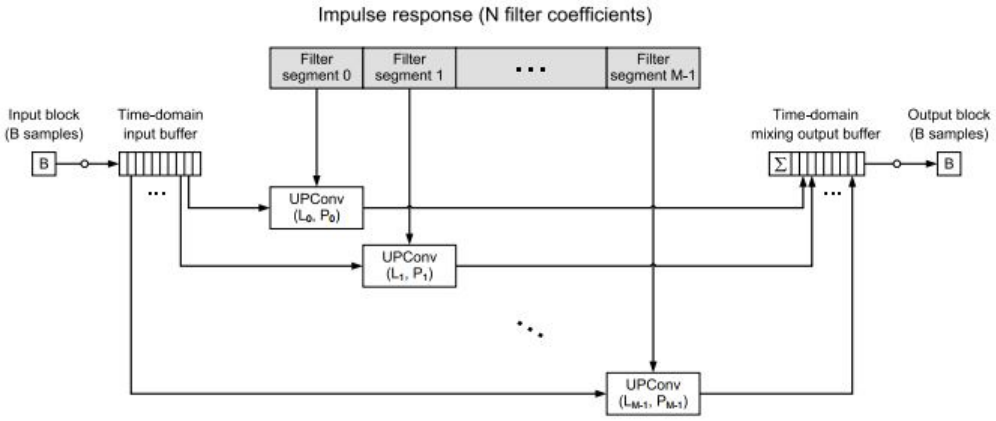

Abbildung 15 – Block Diagramm einer Non-Uniformly Partitioned Convolution Quelle: [Wef14]

3.5.1 Minimum-Cost Algorithmus

Bei der Analyse des Algorithmus wird angenommen, dass die Länge von $h[n] = 8N$ ist. Dabei wird die Impulsantwort nach folgendem Schema unterteilt (siehe Abbildung 16).

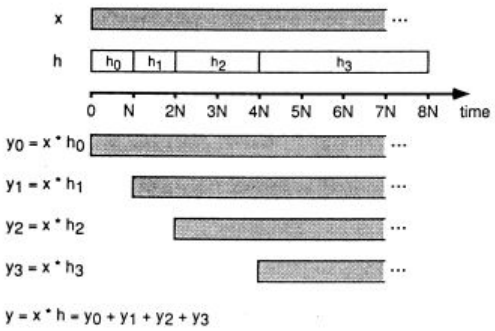

Abbildung 16 – Unterteilung des Filters in 3 Teilfilter. Quelle: [Gar95]

Insgesamt ergeben sich 4 unterschiedlich lange Subblöcke mit $h_0[n] = N$, $h_1[n] = N$, $h_2[n] = 2N$ und $h_3[n] = 4N$. Jeder einzelne Frame wird dabei um die entsprechende Anzahl an Abtastwerten verzögert (außer $h_0[n]$). Anschließend wird jedes Sample der Eingangsfolge $x[n]$ mit den Teilfiltern gefalten. Für kleine N ist der Direktform Filter wesentlich effizienter (ca. $N = 32$ bzw. $N = 64$). Hieraus folgt, dass $y_0[n]$ mittels eines FIR Filters im Zeitbereich gefalten wird. Alle anderen Blöcke werden per OLA bzw. OLS berechnet. Dies bedeutet, dass bei der Berechnung von $y_0[n]$ genug Zeit bleibt, um N Samples in den Eingangsbuffer zu akkumulieren, um die entsprechende Blocktransformation für $y_1[n] = \text{IFFT}\,(X[k] \cdot H_1[k])$ durchzuführen (innerhalb von N Samples). Genauso verfährt man mit $y_2[n]$ und $y_3[n]$ [Gar95].

Gardner [Gar95] bezeichnet dies als Minimum-Cost Algorithmus, da die Länge der Blöcke so gewählt worden ist, sodass eine möglichst geringe Latenz zwischen dem Ei- und Ausgang zustande kommt. Der Algorithmus ist in dieser Form nicht realisierbar, da alle Blockfaltungen innerhalb kürzester Zeit berechnet werden müssen. Jeder M Punkt Block Convolver muss zunächst auf M Input Samples warten bis diese in den entsprechenden Buffer hinein geladen worden sind. Selbst wenn der Prozessor schnell genug wäre, würde man dessen Rechenleistung nicht effizient ausnutzen, da dann die meiste Zeit darauf verwendet werden würde, um Abtastwerte in den Input Buffer zu laden. Bei der praktischen Implementierung wird dafür gesorgt, dass der Prozessor über die Zeit gesehen gleichmäßig ausgelastet wird. Insbesondere bedeutet dies, dass die Rechenzeit zu gleichen Teilen auf die Akkumulierung der Eingangswerte und der eigentlichen Berechnung der Faltung aufgespalten wird.

3.5.2 Scheduling

Jede Faltung eines Eingangblocks mit einer Filter Partition kann als eigener Prozess angesehen werden, welcher in einem bestimmten Zeitfenster abgearbeitet werden muss. Bei einer Filterlänge von $16N$ lässt sich $h[n]$ mit Hilfe des Gardner Schemas in Partitionen

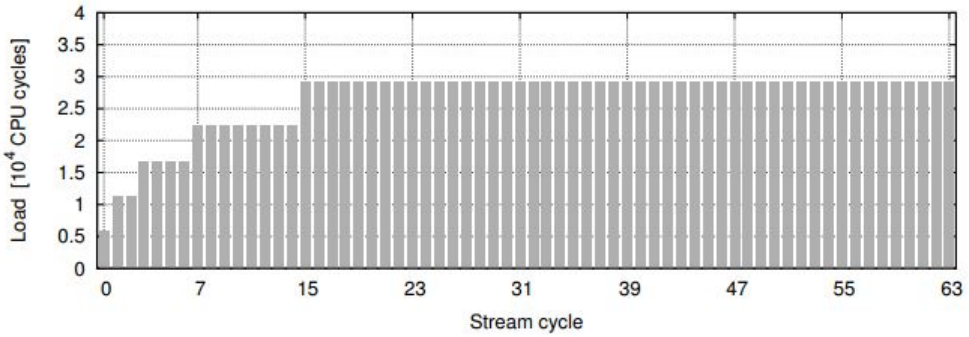

Abbildung 17 – CPU Last Non-Uniformly Partition Convolution. Man erkennt die gleichmäßige Auslastung der CPU. Quelle: [Wef14]

folgender Länge zerlegen [Gar95]. Mit dieser kann eine gleichmäßige CPU Auslastung erreicht werden.

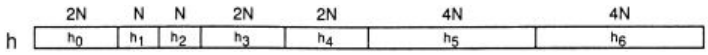

Abbildung 18 – Zerlegung von $h[n]$ in Teilfilter. Quelle: [Gar95]

Eine Faltung mit $h_0[n]$ stellt insofern eine Besonderheit dar, da bei jeder Abtastperiode ein Zwischenergebnis bereit stehen muss. Der geeignete Zeitpunkt dafür wäre, wenn es zu einem Sample Interrupt [22] kommt. In allen anderen Fällen wird eine Block Faltung durchgeführt, die nach allen N Interrupts neu geplant wird. Jedem Block wird eine Priorität, von hoch nach niedrig zugewiesen: N (hoch), $2N$ (mittel) und $4N$ (niedrig). In Abbildung 19 sieht man, dass zum Zeitpunkt $t = 4N$ Blöcke auf allen Prioritätsleveln abgearbeitet werden müssen. Zuerst wird die Faltung mit $x * h_1[3N, N]$, dann mit $x * h_2[3N, N]$ berechnet. Für beide Berechnungen wird der selbe Eingangsblock verwendet, man muss sich also nur einmal die FFT von $x[n]$ berechnen und kann somit die Faltung bzw. Multiplikation mit $h_2[3N, N]$ schneller machen [Gar95].

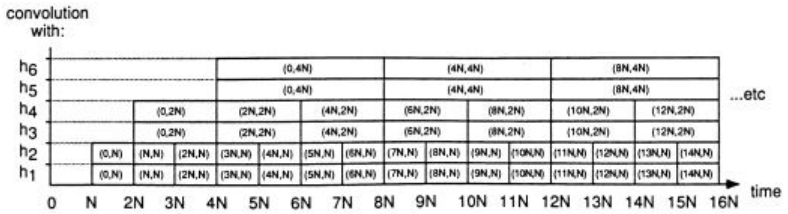

Abbildung 19 – Schedule der einzelnen Blockfaltungen. Quelle: [Gar95]

22. Im Allgemeinen versteht man unter einem Interrupt eine kurzzeitige Unterbrechung eines laufenden Programms bzw. laufender Routine um einen zeitkritischen Vorgang abzuarbeiten. Einen solchen stellt die Faltung mit der Impulsantwort $h_0[n]$ dar. Siehe hierfür https://de.wikipedia.org/wiki/Interrupt

3.5.3 Berechnung der Komplexität

Wie bereits in Kapitel 3.4 geschildert, werden alle (bis auf $h_0[n]$) Teilfilter mittels FFT basierten OLA/OLS Verfahren implementiert. Sei L_i die Länge eines Eingangsblocks, P_i die Länge einer Filterpartition und N_i die Länge der resultierende DFT/FFT, dann gelte [Wef14]:

$$L_i = P_i \tag{3.47}$$
$$N_i = 2L_i = 2P_i \tag{3.48}$$

Des weiteren soll gelten:

$$P, P_i \in \{2^i | i \in \mathbb{N}\} \implies P_i, N_i \in \{2^i | i \in \mathbb{N}\} \tag{3.49}$$

Sei nun $\mathcal{P}$ definiert als eine non-uniforme Filter Partition, dann gilt [Wef14]:

$$\mathcal{P} := \left(P_0^{P_0}, \cdots, P_{M-1}^{P_{M-1}}\right) \tag{3.50}$$

Eine Filterpartition lässt sich somit aus mehreren Segmenten mit der Länge P darstellen. Für einen Ausgangswert lässt sich die Komplexität $T_{NUPConv}$ (NUP = Non-Uniformly Partitioned Convolution) [Wef14] mit:

$$T_{NUPConv}(\mathcal{P}) := \sum_{i=0}^{M-1} \left(T_{UPConv}(P_i, L_i \cdot \mathcal{P}_i) + \frac{T_{ADD}(P_i)}{P_i}\right) \tag{3.51}$$

angeben. Eine einzelne Filterpartition P_i kann als eine uniforme Faltung mit der Block- und Teilfilterlänge $L_i = P_i$ angesehen werden. Summiert man die individuellen Segmente auf, so ergibt sich die ursprüngliche Impulsantwort. Der Ausdruck $\frac{T_{ADD}(P_i)}{P_i}$ berücksichtigt den zusätzlichen Aufwand, wenn man die einzelnen Segmente aufaddiert. Zur Vereinfachung wird dieser Term im folgenden nicht berücksichtigt (siehe [Wef14]). Man erhält:

$$T_{NUPConv}(\mathcal{P}) := \sum_{i=0}^{M-1} T_{UPConv}(P_i, L_i \cdot \mathcal{P}_i) \tag{3.52}$$

3.6 Zeitabhängigkeit

Bei nicht gleich großen Filterpartitionen ergeben sich gewisse Zeitabhängigkeiten (Abbildung 20). Um das Gesamtergebnis der Faltung zu erhalten, müssen zu bestimmten Zeitpunkten die Teilergebnisse bereits berechnet worden sein. Andernfalls fehlen diese im Ergebnis [Wef14].

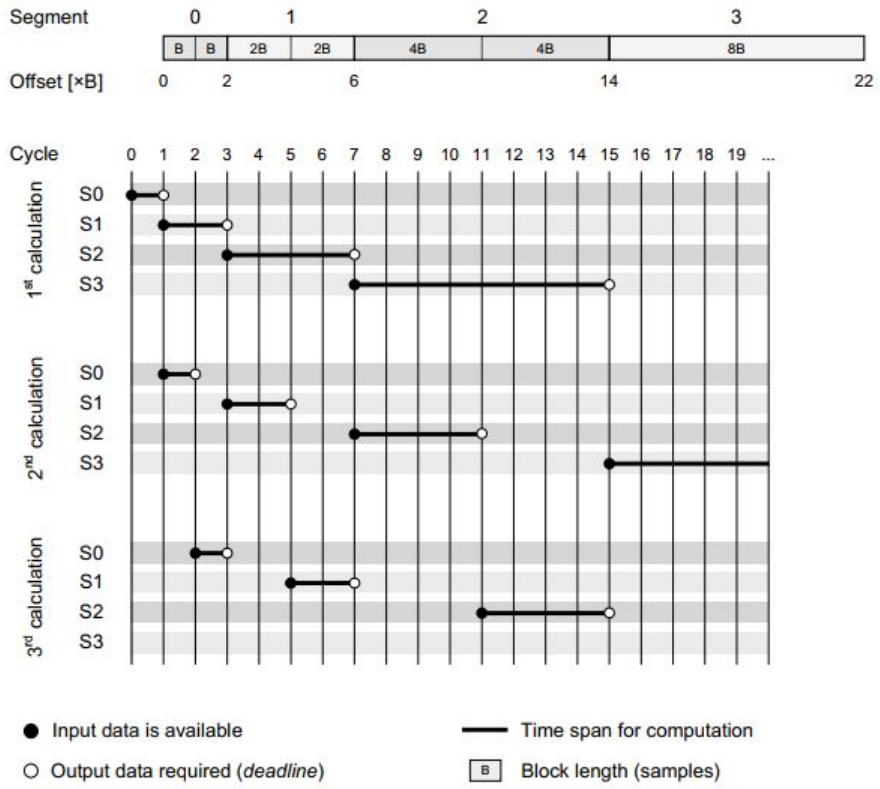

Abbildung 20 – Schedule der einzelnen Blockfaltungen. Quelle: [Wef14]

4 Analytischer Teil

Im zweiten Teil dieser Arbeit werden die Messergebnisse der 3 zuvor erwähnten Faltungsalgorithmen (Uniformly, Non-Uniformly und Sparse-FIR) verglichen und anschließend diskutiert. Sowohl in Xcode, als auch Visual Studio 2017 ist es möglich ein Leistungsprofiling für eine Applikation durchzuführen um Daten über die Leistung eines Programms zu erhalten. Man unterscheidet zwischen Sampling bzw. der Instrumentation.

4.1 Sampling

Beim Sampling werden Daten über den Work-Load einer Applikation gesammelt, während der Leistungsprofiler läuft. In Visual Studio ist es die Standardmethode, um Einblicke in die Performance zu erhalten. Der große Vorteil dieser Messmethode ist, dass sie besonders ressourcenschonend und zusätzlich wenig Einfluss auf die Ausführungszeit des zu untersuchenden Programms hat. Dabei wird in vorgegebenen Zeitintervallen der Prozessor unterbrochen und Informationen über den "Function Callstack"gesammelt. Bei einem Call Stack handelt es sich um eine dynamische Datenstruktur, die alle aktiven Subroutinen eines Computers bzw. Prozessors auflistet. Während dem Profiling wird bestimmt, ob der Prozessor Code in dem zu untersuchende Prozess ausführt. Falls dies nicht der Fall ist, so wird das Sample verworfen, andernfalls erhöht der Profiler den samplecount für jede Funktion auf dem Stack. Zu jedem Zeitpunkt wo ein Sample aufgenommen wird, wird nur eine Funktion am Callstack ausgeführt. Man unterscheidet zwischen Inklusiven und Exklusiven Werten. *Inklusive Werte* (engl. **inclusive samples)** sind die Gesamtzahl aller Werte die während direkter und indirekter Funktionsaufrufe gesammelt worden sind. *Exklusive Werte* (engl. **exclusive samples**) hingegen berücksichtigen nur die direkten Funktionsaufrufe.

4.2 Instrumentation

Bei dieser Messmethode werden zeitliche Informationen über die Ausführungszeit eines Programms bzw. einer Funktion gesammelt. Dabei wird zusätzlicher Code als binäre Datei in die entsprechenden Funktionsaufrufe eingefügt. Bei der Instrumentierung werden folgende Werte betrachtet: Die vergangenen *inklusiven Werten* (engl. **elapsed inclusive**) werden als jene Zeit interpretiert, die während der Ausführung in einer Funktion verbracht wurde. *Applikations inklusive Werte* (engl. **application inclusive**) **elapsed inclusive**, ignorieren hingegen Betriebssystemaufrufe. Für nähere Details sei auf folgende Webseite verwiesen [23]

23. https://docs.microsoft.com/en-us/visualstudio/profiling/cpu-usage?view=vs-2017

4.3 Diskussion

Als Testsysteme wurden ein aktueller Windows 10 Rechner (Intel i7 Prozessor) aus dem Jahr 2018 und ein IMac mit MacOS 10.12 aus dem Jahr 2011 (Intel i5 Prozessor) verwendet. Auf Windows wurde unter Visual Studio getestet, unter Mac OS in Xcode [24]. Schlussendlich wurden die Plugins auf unterschiedlichen Plattformen gemessen um herauszufinden, wie stark sich beispielsweise das Betriebssystem (Ressourcen Management, etc.) oder auch Compileroptimierungen bei der Ausführung der jeweiligen Algorithmen auswirkt. Auf beiden Plattformen wurden die Releasebuilds der einzelnen Plugins getestet. Diese weisen im Gegensatz zu den Debugbuilds bereits Optimierungen auf. Des Weiteren wurde der gleiche Audiotreiber (ASIO) [25] und das gleiche Audiointerface (Focusrite Scarlett 2i2) verwendet um den Vergleich zwischen den Plattformen so fair wie möglich zu gestalten.

Für jeden der folgenden Test Fälle wurden jeweils 5 Messungen durchgeführt, um aussagekräftigere Ergebnisse zu erzielen. Die Erwartungshaltung war, dass sich die Messergebnisse nicht allzu deutlich voneinander Unterscheiden und sich mit der Theorie aus den vorangegangen Kapiteln decken dürften. Dies hat sich in der Praxis auch bestätigt.

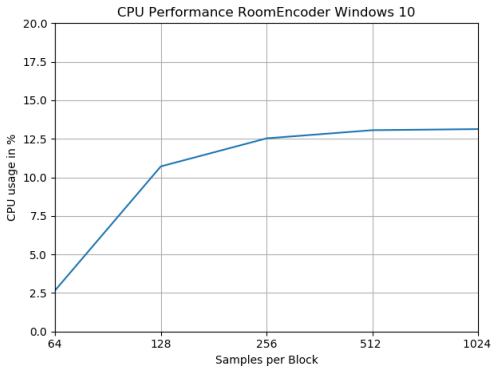

Abbildung 21 – Messerergebnis RoomEncoder ohne ASIO Treiber

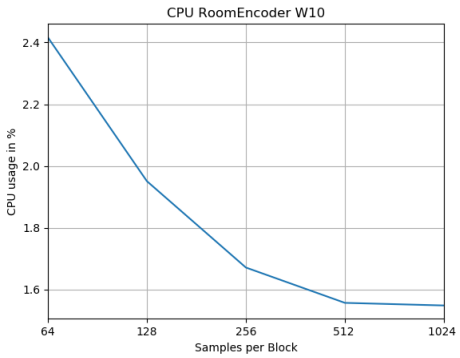

Abbildung 22 – Messergebnis RoomEncoder mit ASIO Treiber,

Bei allen Messungen wurde während des Profilings die Juce Klasse SStandalonePluginHolder"betrachtet, da diese für die Initialisierung eines Standalone Plugins bzw. Applikation verantwortlich ist [26]

24. Die ursprüngliche Idee war, auf beiden Plattformen unter Visual Studio zu testen. Somit wären die Ergebnisse besser vergleichbar gewesen. In der MacOS Version von VS ist es jedoch nicht möglich C/C++ Programme zu kompilieren, siehe hierfür https://bit.ly/2xnDDw1

25. Bevor der ASIO Treiber unter Windows 10 verwendet worden ist wurde zunächst mit dem Standardaudiotreiber getestet. Dies hatte jedoch erhebliche Dropouts und Verzerrungen im Klang zu Folge und wirkte sich ebenso nachteilig auf die Messungen aus, da diese in keinster Weise mehr nachvollziehbar waren (Siehe Abbildung 21).

26. siehe hierfür: https://docs.juce.com/master/classStandalonePluginHolder.html

4.3.1 Matrix Convolver vs. mcfx vs. RoomEncoder

Diese Messung dient dazu, die unterschiedlichen Plugins miteinander vergleichen zu können. Dazu wurde zunächst eine Impulsantwort des RoomEncoder Plugins generiert, indem eine .wav Datei mit einem Einheitsimpuls erzeugt worden ist. Anschließend wurde diese in einer DAW (Digital Audio Workstation) wie Reaper importiert und auf die entsprechende Audiospur der RoomEncoder als Effekt angewendet. Zum Schluss wurde diese Spur herausgerechnet, damit die resultierende Audiodatei in den mcfx bzw. MatrixConvolver hinein geladen werden konnte. Man sieht zum Einen sehr schön, dass die Plugins unter Windows 10 und MacOS den gleichen Verlauf aufweisen. Zum Anderen erkennt man, dass eine Verdopplung der Eingangsblöckgröße, oftmals eine Halbierung der CPU Last zur Folge hat. Dies lässt sich damit begründen, da weniger oft ein Callback aufgerufen werden muss und schlussendlich mehr Zeit bleibt die entsprechenden Eingangswerte in den Eingangsbuffer hineinzuladen. Auf klanglicher Ebene waren keine Unterschiede feststellbar. Man sieht außerdem sehr gut, wie viel effizienter die Non-Uniformly-Partitioned Convolution im Vergleich zu den beiden anderen Faltungsalgorithmen ist. Dieser Unterschied wird vor allem bei langen Impulsantworten sehr deutlich.

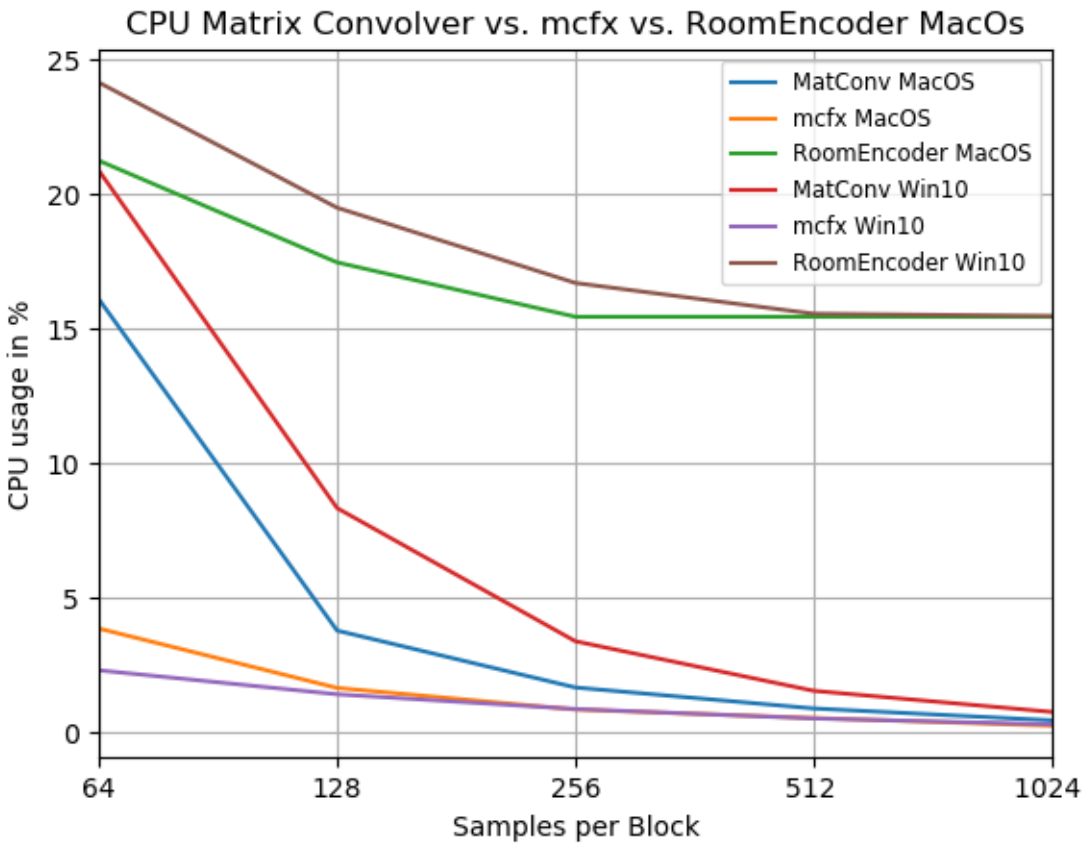

Abbildung 23 – Vergleich Matrix Convolver vs. mcfx vs. RoomEncoder unter Windows 10 und MacOS

4.3.2 Matrix Convolver vs. mcfx

Hier wurde als Impulsantwort normalverteiltes weißes Rauschen mit einer Länge von 65536 Samples verwendet. Der Grund hierfür war, dass die beiden Faltungs-Plugins so stark wie nur möglich ausgelastet werden sollten. Auch hier sieht man wieder, dass beide System den gleichen Verlauf aufweisen. Man erkennt, dass eine Verdopplung der Eingangsblockgröße eine erhebliche Senkung der CPU Last zu folge hatte. Je länger die zu transformierenden Folgen sind, desto mehr Gebrauch wird von der Effizienz der FFT gemacht. Vor allem durch das Scheduling kann der Non-Uniformly-Partitioned Convolution Algorithmus seine stärken ausspielen.

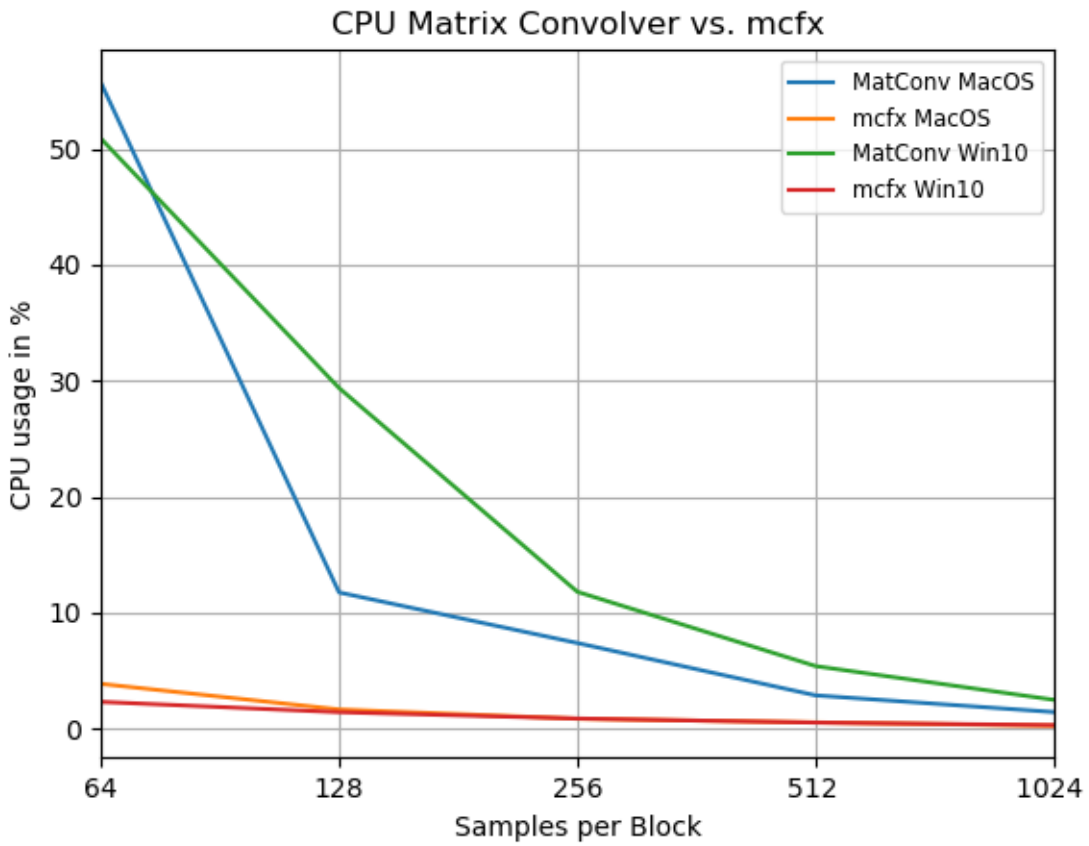

Abbildung 24 – Vergleich Matrix Convolver vs. mcfx unter Windows 10 und MacOS

4.3.3 Matrix Convolver Erweitert

Hierbei wurde versucht die einzelnen Partitionen künstlich zu verlängern um die gesamte Anzahl der zu berechnenden Partitionen vermeintlich zu halbieren. Das bedeutet, wenn die AudioCallBack Funktion alle 64 Werte aufgerufen wird, dann wird mit einer Eingangsfolge von 128 Werten gefalten, wobei die letzten 64 Werte Nullen sind. Das Ergebnis hat jedoch auf beiden Plattformen nicht die gewünschte Reduzierung der CPU Last gebracht.

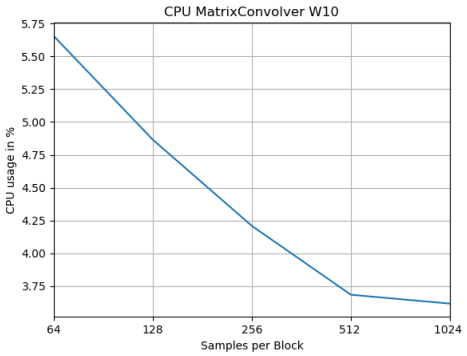

Abbildung 25 – Messergebnis Matrix Convolver unter W10

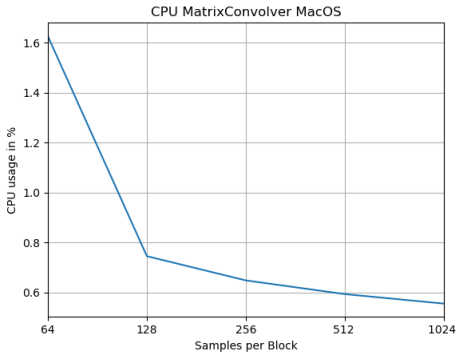

Abbildung 26 – Messergebnis Matrix Convolver unter MacOS

5 Schlussfolgerung

Aus den durchgeführten Messungen geht hervor, dass sich alle Faltungsalgorithmen bei kurzen Impulsantworten sehr ähnlich sind. Je länger die Filterlänge wird, desto eher sollte man auf Blocktransformationen in Kombinationen mit OLA bzw. OLS Verfahren zurückgreifen. Bei einem nicht partitionierten Filter können sich erhebliche Verzögerung zwischen dem Ein- und Ausgang ergeben. Für kürzere bis mittlere Impulsantworten eignet sich die Uniformly Partitioned Convolution sehr gut, da sie recht einfach zu implementieren, aber trotzdem sehr effizient ist. Möchte man also so wenig Latenz wie nur möglich haben, so empfiehlt es sich eine Non-Uniformly Partitioned Convolution einzusetzen. Dies ist mit einem deutlichen Implementierungsaufwand verbunden, da man einen möglichst effizienten Scheduler schreiben müsste. Falls man hauptsächlich mit statischen Quellen in einer musikalischen Szene arbeitet, so sind die Uniformly und Non-Uniformly dem FIR bzw. sparse-FIR zu bevorzugen, anderenfalls sollte man auf letzteren zurückgreifen, aufgrund der geringen Verzögerung zwischen Ein und Ausgang.

Literatur

[CLRS09] Cormen, Leierson, Rivest, and Stein, *Introductions to Algorithmns*, 3rd ed. MIT Press, 2009.

[Gar95] W. G. Gardner, "Efficient convolution without input-output delay," *Journal Audio Engineering Society*, vol. 43, no. 3, 1995.

[GW14] G. Graber and W. Weselak, *Raumkustik*, 2014.

[MC03] M. Moeser and L. Cremer, *Technische*, 5th ed. Springer, 2003.

[Moe15] M. Moeser, *Technische Akustik*, 10th ed. Springer, 2015.

[OS14] A. V. Oppenheim and R. W. Schafer, *Discrete Time Signal Processing*, 3rd ed. Pearson, 2014.

[Smi] J. O. Smith, *Physical Audio Signal Processing*. MIT Press.

[Wef14] F. Wefers, "Partitioned convolution algorithms for real-time auralization," PhD Thesis, RWTH Aachen University, RWTH Aachen University, 2014.

[Zoe] U. Zoelzer, *DAFX - Digital Audio Effects*. Wiley.

[ZZ93] E. Zwicker and M. Zollner, *Elektroakustik*, 3rd ed. Springer, 1993.