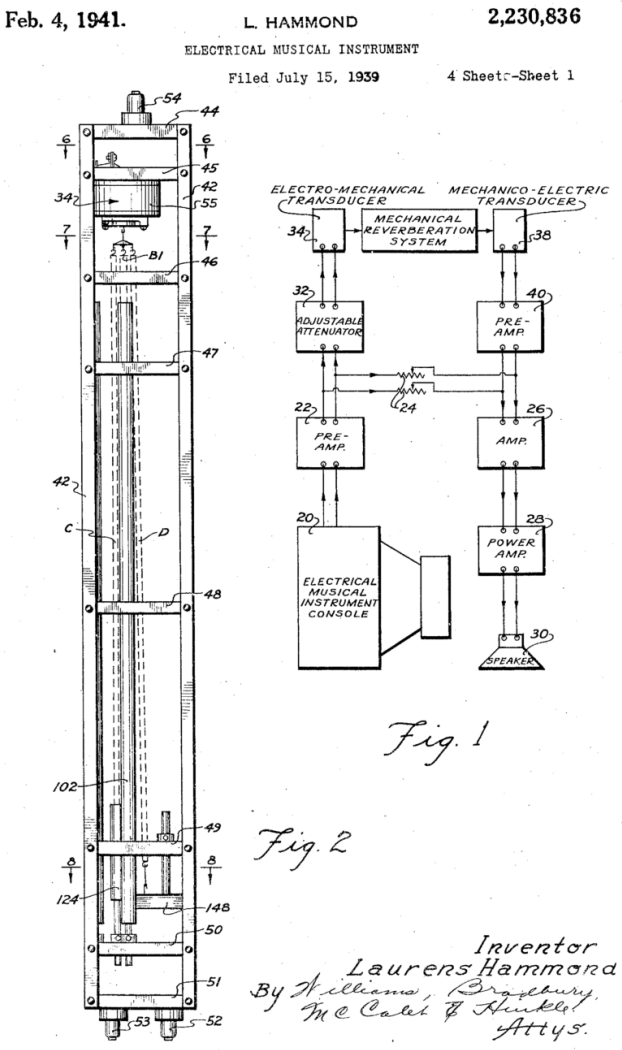
Physikalische Modellbildung eines Federhalls

Projektbericht - Audiotechnik - WS 2020



Jan Abel (876662)[1], Pierre-Hugues Hadacek (922337)[2], Steven Thiele (923364)[2], and [3]

[1]Technische Informatik - Embedded Systems, Berliner Hochschule für Technik
[2]Elektrotechnik - Kommunikationstechnik, Berliner Hochschule für Technik

29. März 2021

Inhaltsverzeichnis

1 Einleitung

In diesem Projekt wurde die Synthese eines Federhalls (engl. spring reverb) in Matlab durchgeführt. Federhallgeräte werden zur künstlichen Nachhall Erzeugung verwendet. Ziel war es ein Eingangssignal mithilfe unseres Modells eines Federhallgeräts mit einem Nachhall zu versehen.

Da die Simulation deutlich komplexer war als Anfangs angenommen, ging der Implementierung eine umfangreiche Literaturrecherche voraus. Besonders die Veröffentlichungen zur Dissertation *Dispersive Systems in Musical Audio Signal Processing* (J. D. Parker 2013) war hier eine große Orientierung. Während in diesen der Federhall mit Hilfe von Finiten Differenzen Methoden oder auch Allpass-Strukturen simuliert wurde, haben wir uns in diesem Projekt für die physikalische Modellierung mittels Modal-Synthese entschieden. Weitere wichtige Quellen war das Buch *Numerical Sound Synthesis* von (Bilbao 2013) und die Webseite von Prof. J. O. Smith (Smith 2021).

Ursprünglich war geplant das Projekt als Plug-In im VST-Format umzusetzen, um es in einer professionellen Audioworkstation (DAW) nutzen zu können. Das Format VST (Virtual Studio Technology) wurde 1991 von Steinberg in der Software Cubase eingeführt. Es gilt als die Lingua Franca der Audio-API im professionellen Audio-Bereich und wird als Standard-Plug-in-Format in den meisten professionellen DAWs verwendet (bemerkenswerte Ausnahme ist Pro-Tools, das nur sein natives aax-Format unterstützt). Allerdings handelt es sich nicht um eine offene Technologie und für die kommerzielle Entwicklung fallen Lizenzgebühren an. Matlab hat eine eingebaute Funktion, um ein Audioprojekt in ein VST-Plug-in zu wandeln.

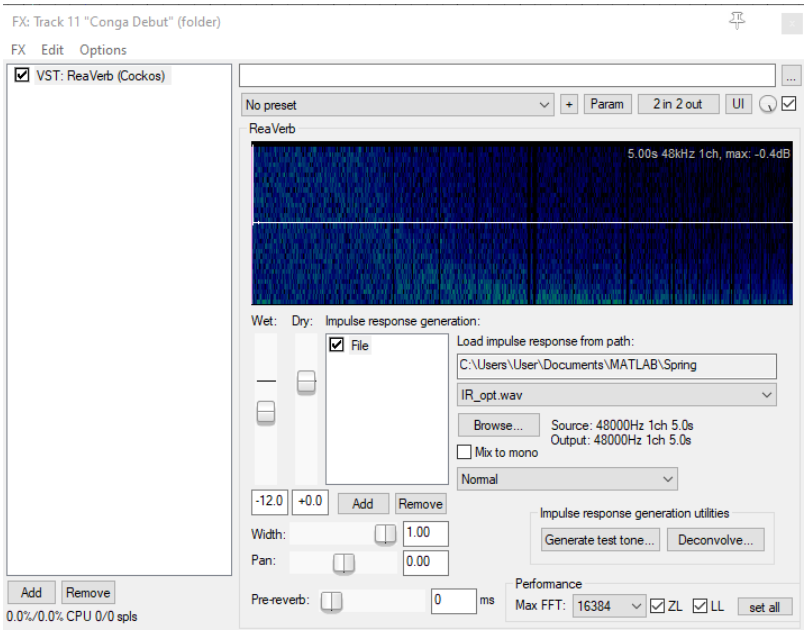


Abbildung 1: ReaVerb mit Impulsantwort der Modellierung

Aus zeitlichen Gründen haben wir uns aber dazu entschieden stattdessen mit unseren Script Impulsantworten zu erzeugen. Diese können dann mit einem beliebigen Convolution-Plug-in in Digitalen Audio Workstations (DAW) benutzt werden. Es gibt mehrere auf dem Markt. Beispielsweise ReaVerb, den eingebauten Faltungprozessor von REAPER. Obwohl es sich um ein kommerzielles Programm handelt, hat es eine voll funktionsfähige 60-tägige Testphase, die nicht eingeschränkt ist. Der Leser, der die erzeugten Impuls Antworten testen möchte, kann die Software auch über diesen Zeitraum hinaus frei nutzen.

2 Grundlagen und Theorie

2.1 Nachhall

Ein Hörer, der in einiger Entfernung von einer Schallquelle steht, nimmt Schall wahr, der tatsächlich eine Kombination aus direktem und indirektem Schall ist, der von Wänden oder Objekten reflektiert wird. Die Reflexionen werden als Nachhall bezeichnet. Nachhall kann den wahrgenommenen Klang einer Quelle deutlich verbessern. Die ersten Reflexionen prägen die Vorstellung von der Raumgröße. Die darauf folgenden Reflexionen vermitteln die Lebendigkeit eines Raums.

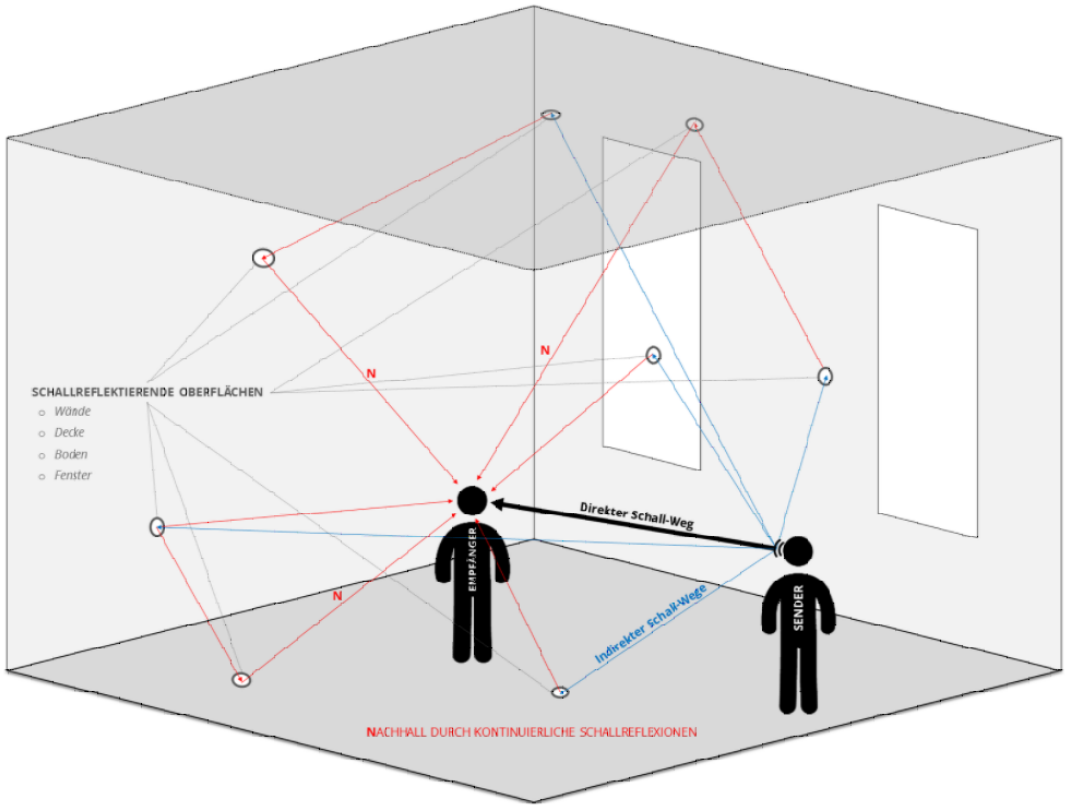


Abbildung 2: Nachhall

Stellt man sich vor, man befindet sich in einer großen Halle und klatscht einmal in die Hände. Die Zeitdauer, die für das Eintreffen der allerersten Reflexionen erforderlich ist, wird als Verzögerungszeit bezeichnet und hängt vom Raumvolumen (oder dem Abstand der reflektierenden Oberflächen vom Hörer) ab. Die Anzahl und Dichte der Reflexionen nimmt mit der Zeit schnell zu und sie werden unübersichtlich, während gleichzeitig der Pegel abnimmt, bis sie nicht mehr hörbar sind. Die Zeitdauer, die ein Schall benötigt, um seinen Pegel um 60 dB zu verringern, wird als Abklingzeit bezeichnet und hängt mit der größe des Raums und den akustischen Eigenschaften der reflektierenden Oberflächen im Hörbereich zusammen. Beispielsweise reflektieren gegossene Betonwände mehr (weniger absorbierend) akustische Energie als Trockenbauwände. Dieses Prinzip ist auch auf Festkörper übertragbar.

2.2 Federhall

Ein Federhall auch Hallspirale genannt ist ein elektro-akustisches Effektgerät mit dem sich ein künstlicher Nachhall erzeugen lässt. Das Gerät besteht aus folgenden Baugruppen, die zur Erzeugung des Federhall-Effekts verwendet werden.

Einen Eingangs- und Ausgangswandler bestehend aus einer Spule, die um eine magnetische Laminierung zentriert ist, und kleinen zylindrischen Magneten, die im Luftspalt der Laminierung zentriert sind.

Die Übertragungsfedern sind auf einem inneren Aluminiumkanal montiert, der über vier kleine Stützfedern mit einem äußeren Stahlgehäuse verbunden sind.

Ein an die Eingangswandlerspule angelegtes elektrisches Signal erzeugt ein magnetisches Wechselfeld, das die Wandlermagnete bewegt. Die Magnete sind mechanisch mit Getriebefedern gekoppelt. Das Signal wird durch die Übertragungsfedern mit einer Verzögerung hin und her reflektiert, die durch den Durchmesser, die Drahtstärke und die Länge jeder Feder bestimmt wird. Die beweglichen Magnete des Ausgangswandlers erzeugen ein magnetisches Wechselfeld, das ein elektrisches Signal in der Ausgangswandlerspule induziert.

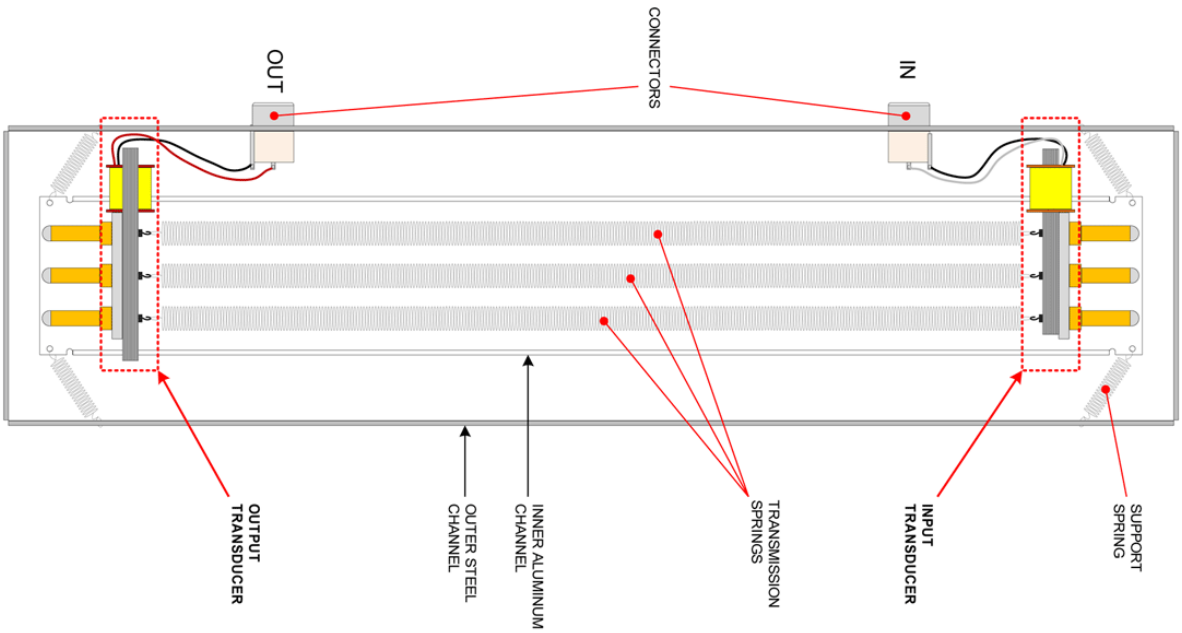


Abbildung 3: Mechanischer Aufbau eines Federhalls

Die Verwendung mehrerer Übertragungsfedern trägt zur Verbesserung der Hallcharakteristik bei. Ein Zuhörer in einem großen Saal mit natürlichem Nachhall steht normalerweise nicht in gleichem Abstand von jeder reflektierenden Oberfläche. Natürlich gibt es Reflexionen von verschiedenen Oberflächen mit unterschiedlichen Verzögerungszeiten. Die Verwendung mehrerer Übertragungsfedern mit unterschiedlichen Verzögerungszeiten dient dazu, eine natürlichere Umgebung zu simulieren und den Gesamtfrequenzgang zu verbessern, da die Reaktion einer Feder Hohlräume oder Löcher in der Reaktion der anderen Feder füllt.

Der Hallbehälter sollte so weit wie möglich von vibrierenden Oberflächen isoliert sein. Bei der Montage des Gehäuses muss vermieden werden, dass der äußere Kanal des Hallbehälters direkt an dem Gehäuse montiert wird. Dies kann wiederum mit Federn bewerkstelligt werden oder mit anderen Methoden, die für die mechanische Isolierung ausgelegt sind.

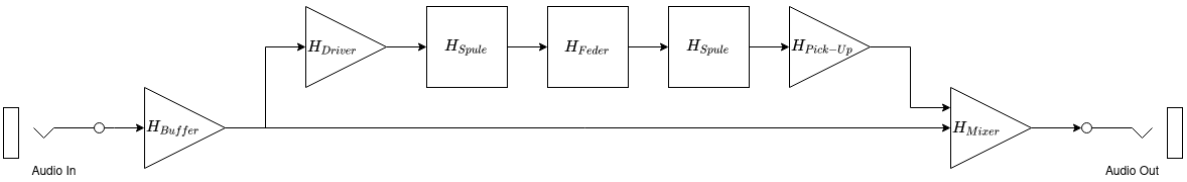


Abbildung 4: Schematischer Aufbau Federhallsystem

Laurens Hammond aus Illinois hat in den 1940er und 1950er Jahren die Verwendung künstlicher Nachhallgeräte durch seine Kirchenorgeln populär gemacht. Die frühen Hammond-Orgeln wurden an Kirchen verkauft, da viele Kirchen in den USA reflexionsarme Räume waren und die Hammond-Orgel musste ihren eigenen künstlichen Nachhall erzeugen.

Abbildung 5: Laurens Hammond

Federhallgeräte wurden in Gitarrenverstärker von E-Gitarren verbaut, was man in Popmusik der 1960er Jahre noch deutlich raushört und bis in die 1970er Jahre regelmäßig im Tonstudio neben Platten-Hall-Geräten oder auch Hallräumen zur Erzeugung von Nachhall einsetzte. Seit den 1980er Jahren wird künstlicher Nachhall in der Regel mit elektronischen Anordnungen erzeugt. Die Realisierung ist derart preiswert, dass auch einfache Mischpulte für Amateurmusiker oft damit ausgerüstet sind.

2.3 Physikalisches Modell

Um den charakteristischen Klang eines Federhalls zu synthetisieren, muss die Wellenausbreitung in der Feder beschrieben werden. Die Modellierung von Vibration in einer Feder ist aufgrund der geometrischen Eigenschaften des Helix relativ komplex. Dieses führt nach Wittrick oder auch (Sorokin 2009) zu einem Modell, welches aus einem Gleichungssystem mit 12 gekoppelten partiellen Differentialgleichung besteht.

$$
\begin{aligned}
\rho A \frac{\partial^2 \bar{u}}{\partial t^2} &= \frac{\partial Q_x}{\partial \bar{s}} + \frac{\cos^2\psi}{R} N_z - \frac{\sin\psi\cos\psi}{R} Q_y \\
\rho A \frac{\partial^2 \bar{v}}{\partial t^2} &= \frac{\partial Q_y}{\partial \bar{s}} + \frac{\sin\psi\cos\psi}{R} Q_x \\
\rho A \frac{\partial^2 \bar{w}}{\partial t^2} &= \frac{\partial N_z}{\partial \bar{s}} - \frac{\cos^2\psi}{R} Q_x \\
\rho I_x \frac{\partial^2 \alpha}{\partial t^2} &= \frac{\partial M_x}{\partial \bar{s}} + \frac{\cos^2\psi}{R} T_z - \frac{\sin\psi\cos\psi}{R} M_y - Q_y \\
\rho I_y \frac{\partial^2 \beta}{\partial t^2} &= \frac{\partial M_y}{\partial \bar{s}} + \frac{\sin\psi\cos\psi}{R} M_x + Q_x \\
\rho I_p \frac{\partial^2 \gamma}{\partial t^2} &= \frac{\partial T_z}{\partial \bar{s}} - \frac{\cos^2\psi}{R} M_x \\
\frac{M_x}{E I_x} &= \frac{\partial \alpha}{\partial \bar{s}} + \frac{\cos^2\psi}{R}\gamma - \frac{\sin\psi\cos\psi}{R}\beta \\
\frac{M_y}{E I_y} &= \frac{\partial \beta}{\partial \bar{s}} + \frac{\sin\psi\cos\psi}{R}\alpha \\
\frac{T_z}{G I_p} &= \frac{\partial \gamma}{\partial \bar{s}} - \frac{\cos^2\psi}{R}\alpha \\
\frac{Q_x}{\kappa G A} &= \frac{\partial \bar{u}}{\partial \bar{s}} - \frac{\sin\psi\cos\psi}{R}\bar{v} + \frac{\cos^2\psi}{R}\bar{w} - \beta \\
\frac{Q_y}{\kappa G A} &= \frac{\partial \bar{v}}{\partial \bar{s}} + \frac{\sin\psi\cos\psi}{R}\bar{u} + \alpha \\
\frac{N_z}{E A} &= \frac{\partial \bar{w}}{\partial \bar{s}} - \frac{\cos^2\psi}{R}\bar{u}
\end{aligned}
\tag{1}
$$

Dieses Modell beschreibt das Verhalten auch in einem hohen Frequenzbereich (MHz). Da dieses für eine Audioanwendung überflüssig und viel zu komplex ist, kann nach (Fletcher, Tarnopolskaya, and Hoog 2001) unter der Annahme, dass der Winkel Ψ des Helix sehr klein ist, was bei gewöhnlichen Federhallsystemen der Fall ist, das Modell in ein System bestehend aus nur zwei gekoppelten, partiellen Differentialgleichungen vereinfacht werden.

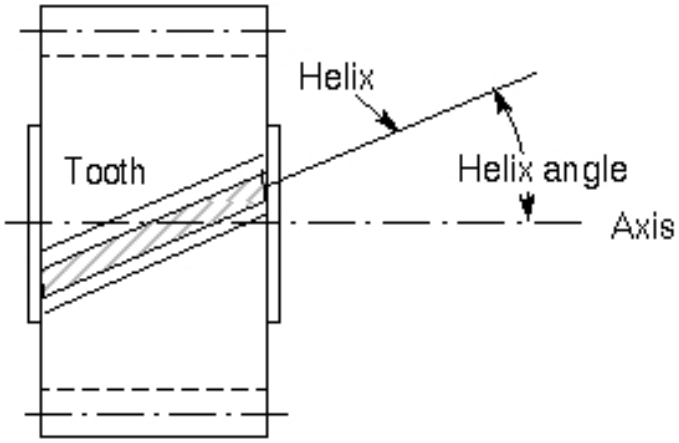


Abbildung 6: Winkel Helix Ψ

Dieses vereinfachte Modell erhält weiterhin das charakteristische Schwingungsverhalten im hörbaren Frequenzbereich und ist deshalb gut geeignet um das klangliche Verhalten des Federhalls zu beschreiben.

$$\begin{aligned} \frac{\partial^2 u}{\partial t^2} &= \frac{E}{\rho}\left(\frac{\partial^2 u}{\partial s^2} - \alpha\frac{\partial \upsilon}{\partial s}\right) \\ \frac{\partial^2 \upsilon}{\partial t^2} &= -\frac{EI}{\rho A}\left(\frac{\partial^4 \upsilon}{\partial s^4} + 2\alpha^2\frac{\partial^2 \upsilon}{\partial s^2} + \alpha^4\upsilon\right) + \frac{E\alpha}{\rho}\left(\frac{\partial u}{\partial s} - \alpha v\right) \end{aligned} \tag{2}$$

Durch eine weitere Annahme kann das Modell weiter vereinfacht werden. Der Parameter α beschreibt die Krümmung des Helix. Wird $\alpha = 0$ gesetzt werden die Gleichungen entkoppelt. Und es ergeben sich zwei unabhängige Differentialgleichungen.

$$\begin{aligned} \frac{\partial^2 u}{\partial t^2} &= \frac{E}{\rho}\frac{\partial^2 u}{\partial s^2} \\ \frac{\partial^2 \upsilon}{\partial t^2} &= -\frac{EI}{\rho A}\frac{\partial^4 \upsilon}{\partial s^4} \end{aligned} \tag{3}$$

Die erste der beiden partiellen Differentialgleichung ist die Wellengleichung in einer Dimension auch D'Alembert-Gleichung genannt. Sie beschreibt eine Mischung aus longitudinaler und torsionaler Wellenausbreitung in Richtung u.

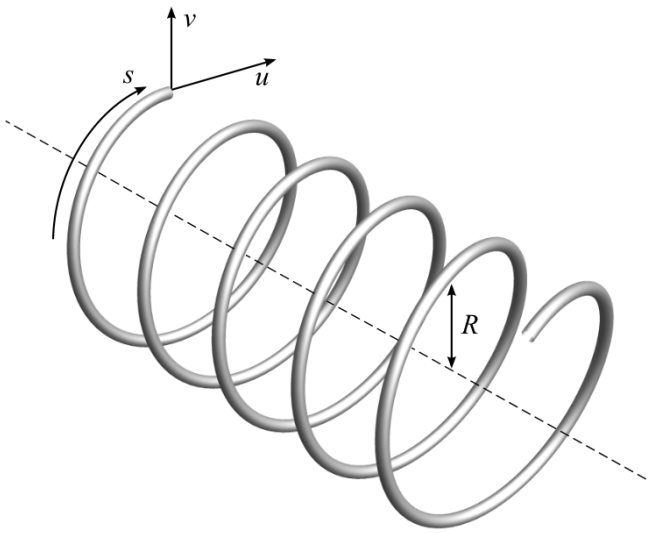


Figure 3: *Section of helical spring showing the direction of the displacements u and v, both of which are dependent on the helix arc-length coordinate, s. Also shown is the helix radius, R.*

Abbildung 7: Feder Helix im Koordinatensystem

Die zweite Differentialgleichung beschreibt die Wellenausbreitung in einem idealen Stab. In der Literatur wird diese partielle DGL auch Euler-Bernoulli-Balkentheorie genannt. Sie beschreibt die Transversal-Wellen in Richtung v. Der unterschiedliche Grad der Ableitung auf der linken und rechten Seite der PDG weist auf ein nicht-lineares Dispersionverhalten hin.

2.4 Dispersion

Die Dispersion beschreibt im allgemeinen die Abhängigkeit der Ausbreitungsgeschwindigkeit einer Welle von deren Frequenz bzw. Wellenlänge. Damit unterscheidet sich im Falle von Dispersion die Phasengeschwindigkeit v_p von der Gruppengeschwindigkeit v_g.

Die Phasengeschwindigkeit ist das Verhältnis der Kreisfrequenz ω zur Kreiswellenzahl k und beschreibt die Ausbreitungsgeschwindigkeit gleicher Phasen in einer Welle.

Die Gruppengeschwindigkeit ist die Geschwindigkeit mit der sich die Einhüllende (*engl. Envelope*) eines Wellenpaketes ausbreitet.

$$\begin{aligned} v_p(\omega) &= \frac{\lambda}{T} = \frac{\omega}{k} \\ v_g(\omega) &= \frac{\partial \omega}{\partial k} \end{aligned} \tag{4}$$

Um dieses besser zu verstehen hilft es die Dispersionsrelation zu betrachten. Diese beschreibt den Zusammenhang zwischen der Kreisfrequenz und der Kreiswellenzahl. Die Dispersionsrelationen der beiden PDG werden im folgenden hergeleitet um den Unterschied zwischen linearen und nicht-linearen Verhältnis zu verdeutlichen. Zunächst wird die 1D-Wellengleichung betrachtet.

$$\frac{\partial^2 u}{\partial t^2} = \frac{E}{\rho}\frac{\partial^2 u}{\partial x^2} \tag{5}$$

Es wird folgende Lösung eingesetzt.

$$u(x,t) = A\cos(kx - \omega t) \tag{6}$$

Differenziert und eingesetzt ergibt dieses folgende Gleichung.

$$\begin{aligned} -\omega^2 A\cos(kx-\omega t) &= -\frac{E}{\rho}k^2 A\cos(kx-\omega t) \\ \omega^2 &= \frac{E}{\rho}k^2 \\ \omega &= \frac{E}{\rho}k \end{aligned} \tag{7}$$

Es ergibt sich also ein linearer Zusammenhang zwischen Kreisfrequenz und Kreiswellenzahl.

Nun wird die Wellenausbreitung in einem idealen dünnen Stab betrachtet. Die geometrischen und materialabhängigen Konstanten werden zu einem gemeinsamen Parameter κ zusammengefasst. Dieser Parameter wird sowohl hier als auch später in der Umsetzung in Matlab zur Übersichtlichkeit benutzt.

$$\begin{aligned} \frac{\partial^2 u}{\partial t^2} &= -\frac{EI}{\rho A}\frac{\partial^4 u}{\partial x^4} \\ \frac{\partial^2 u}{\partial t^2} &= -\kappa^2\frac{\partial^4 u}{\partial x^4} \end{aligned} \tag{8}$$

Es wird wieder folgende Lösung eingesetzt.

$$u(x,t) = A\cos(kx - \omega t) \tag{9}$$

Differenziert und eingesetzt ergibt sich folgender Ausdruck.

$$\begin{aligned} -\omega^2 A\cos(kx-\omega t) &= -\kappa^2 k^4 A\cos(kx-\omega t) \\ \omega^2 &= \kappa^2 k^4 \\ \omega &= \kappa k^2 \end{aligned} \tag{10}$$

Es ergibt sich kein linearer Zusammenhang wie zuvor, sondern ein Quadratischer. Dieses ist auch in der folgenden Grafik zu sehen.

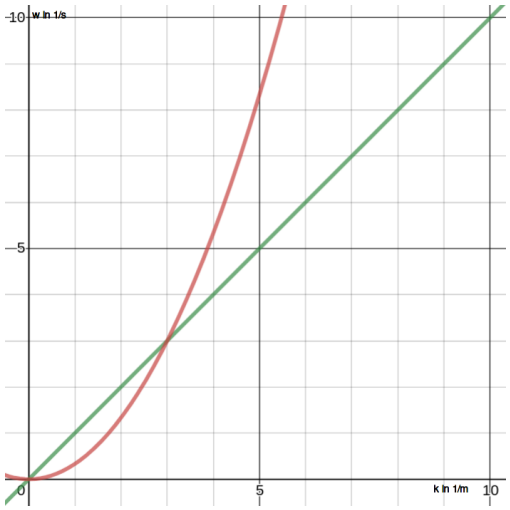


Abbildung 8: Dispersionsrelation (Grün: D'Alembert-Gleichung, Rot: Euler-Bernoulli)

Die akustische Dispersion der Transversalwellen in Richtung v sind somit eine wesentliche klanglich-charakteristische Eigenschaft des Federhalls.

2.5 Modal - Synthese

Die Modalsynthese wird zur Untersuchung von komplexen schwingungsfähigen Systemen in Maschinen- und Anlagenbau eingesetzt und fand darüber hinaus auch den Weg in die Akustik. Das System wird über die Eigenfrequenzen f_n der Moden beschrieben. Diese sind sowohl von den geometrischen Eigenschaften als auch von Eigenschaften des Materials abhängig. Die Beschreibung erfolgt über die Addition von exponentiell abklingenden Sinusfunktionen mit komplexen Amplituden.

$$s(t) = \sum_{n=1}^{N} A_n e^{-\frac{t}{\tau_n}} \sin(2\pi f_n t) \tag{11}$$

Die Gleichung setzt sich zusammen aus den Frequenzen f_n der Moden, den komplexen Amplituden A_n, und die über die Dämpfung definierte Abklingzeiten τ_n.

Diese Parameter können erstens durch das zugrunde liegende, physikalische Modell in Form der partiellen Differentialgleichung, inklusive der Randwert Bedingungen beschrieben. Zweitens durch die Bedingung der Anregung (engl. Ex) des Systems (Ort, Art) und die Anfangswertbedingungen des Systems (Bilbao 2013).

Vernachlässigt man die Induktivitäten und die magnetischen Perlen am Ende Feder, kann bei einem Federhall vereinfacht das Audio-Signal selbst als Funktion für die Anregungsbedingung betrachtet werden.

Aus der Euler-Bernoulli Gleichung, kann unter der Beachtung der einfachsten Randwertbedingung das beide Enden des Stabes bzw. hier der Feder einfach aufgehängt sind (engl. Simply Supported - S) die Funktion $f_r(n)$ nach (Bilbao 2013) zur Berechnung der Eigenfrequenzen hergeleitet werden.

$$\begin{aligned} f_r(n) &= \frac{\pi}{2L^2}\kappa n^2, n \in \mathbb{N} \\ w_r(n) &= 2\pi f_r(n) \end{aligned} \tag{12}$$

2.6 Hochpass Filter 2. Ordnung

Um die Modalsynthese zu Implementieren ist eine Parallelschaltung aus Hochpassfiltern 2. Ordnung geeignet. Dieses LTI-System besitzt sowohl eine Resonanzfrequenz ω_k, als auch einen Dämpfungsfaktor D und wird durch folgende Übertragungsfunktion beschrieben.

$$\begin{aligned} H_{HP}(s) &= \frac{s^2}{s^2 + 2D\omega_k s + \omega_k^2} \\ H_{Feder}(s) &= \sum_{n=1}^{N} H_{HP_n}(s) \end{aligned} \tag{13}$$

Die Impulsantwort des Hochpass Filter 2. Ordnung entspricht den mathematischen Anforderungen der Modal-Synthese und ist im folgenden zu sehen.

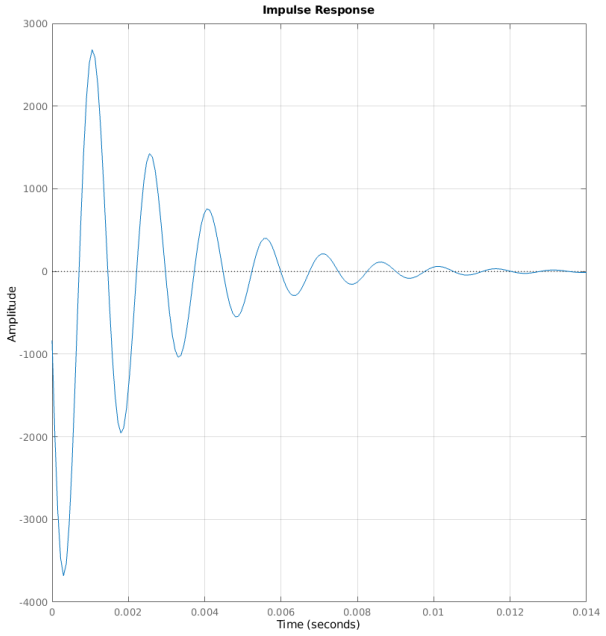


Abbildung 9: Impuls Antwort HP 2. Ordnung ($\omega_k = 2\pi 666\text{s}^{-1}$, $D = 0.1$)

Das Gesamtsystem $H_{Modell}(s)$ ist die Summe der Filter und wird durch folgendes Blockschaltbild dargestellt.

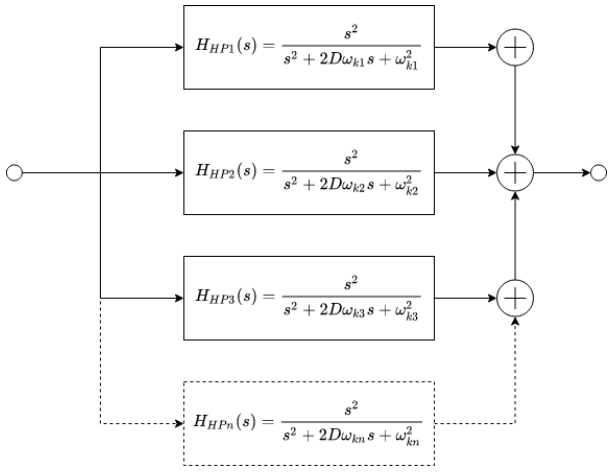


Abbildung 10: Blockschaltbild der Implementierung

2.7 Bilineare Transformation

Die Bilineare Transformation ist, auch als Tustin-Methode bezeichnet, ist nach (Smith 2021) eine Transformation um vom zeitkontinuierlichen zur zeitdiskreten Darstellung von Systemen zu wandeln. Sie ist definiert durch folgende geschickt gewählte Variablen Substitution.

$$s = c\frac{1 - z^{-1}}{1 + z^{-1}}, c > 0, c = \frac{2}{T_s} \tag{14}$$

Es lässt sich zeigen das die Bilineare Transformation die komplette s-Ebene einmal auf den Einheitskreis abbildet. Dies hat zum Vorteil das dieses Verfahren gegenüber z.B der Impulsinvarianzmethode keine Alias-Effekte im diskreten System zur Folge hat. Der Nachteil ist allerdings eine Frequenzverzerrung, da der kontinuierliche, unendliche Frequenzbereich auf den endlichen Bereich des Einheitskreises abgebildet wird.

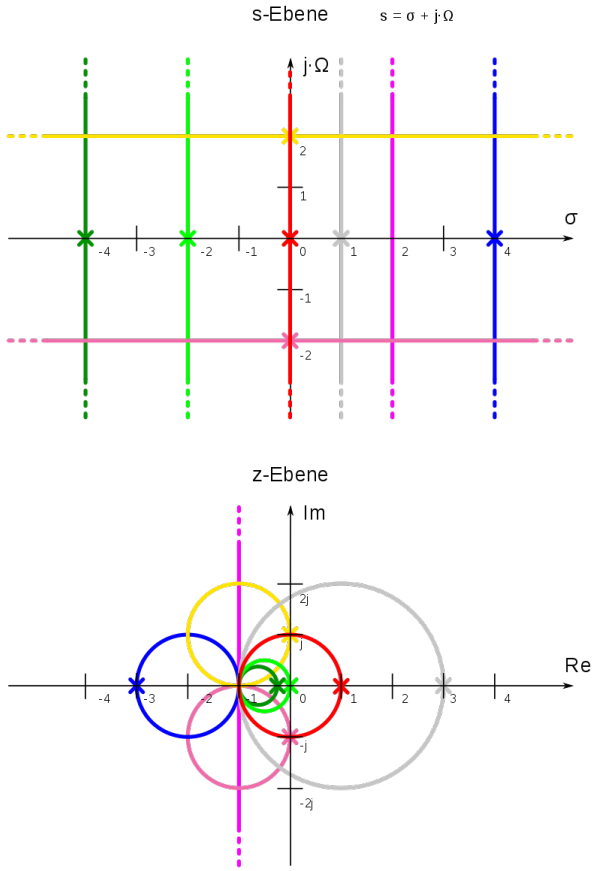


Abbildung 11: Zuordnung Bilinear Transformation

Wenn also die Substitution auf ein analoges System H_A in der s-Ebene angewandt wird, erhält man ein angenähertes digitales System H_D im Z-Bereich.

$$H_D = H_A\left(\frac{2}{T_s}\frac{1-z^{-1}}{1+z^{-1}}\right) \tag{15}$$

2.8 Digitales Biquad-Filter

Ein digitales Biquad-Filter ist ein linearer IIR-Filter 2. Ordnung mit jeweils zwei Pol und Nullstellen. Er ist durch folgende normalisierte Übertragungsfunktion gegeben.

$$H_{BQF} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{16}$$

Da IIR-Filter höherer Ordnung sehr schnell instabil werden können, werden in der Umsetzung in der Umsetzung nur Biquad-Filter eingesetzt.

2.9 Faltung

Eine Faltung ist ein Integral, das den Überlappungsgrad einer Funktion g bei der Verschiebung über eine andere Funktion f ausdrückt. Es “verschmilzt” also eine Funktion mit einer anderen. Die Faltung

ermöglicht es uns, den Ausgang eines LTI-Systems zu einem beliebigen Eingangssignal zu bestimmen.

LTI steht für *linear time invariant* System. Alle Systeme, die sich über gewöhnliche lineare DGL mit konstanten Koeffizienten beschreiben lassen, lassen sich als LTI-Systeme bezeichnen. Über die Herleitung der komplexen Exponentialfunktion als Eigenfunktion von LTI-Systemen kann gezeigt werden, dass jedes LTI-System auch mit eine Übertragungsfunktion beschrieben werden kann.

Das Modell der Feder kann als LTI System beschrieben werden. Die Übertragungsfunktion kann durch Anregung mit einer Dirac-Stoss berechnet werden.

Das Faltungsprodukt ist folgendermaßen definiert:

$$y(t) = (x * h)(t) = \int x(\tau) * h(t - \tau) d\tau$$

h(t) steht für die Impulsantwort des Systems, y(t) für den Ausgang und x(t) für den Eingang des Systems. Diese Formel beschreibt die zentrale Eingangs- und Ausgangsbeziehung von LTI-Systemen im Zeitbereich.

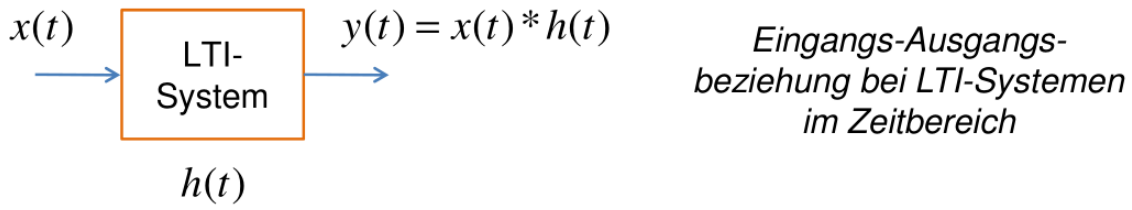


Abbildung 12: Eingangs-Ausgangs-beziehung bei LTI-Systemen im Zeitbereich [Jakob]

3 Umsetzung

Für die Umsetzung des Algorithmus wurde die numerische Software MATLAB genutzt. Auch unter der freien Software GNU-Octave sollten die Scripte mit Einschränkungen funktionieren. Dieses wurde allerdings nicht getestet. Es wurden zwei Matlab-Scripte mit jeweils eigenen Funktionen geschrieben.

Die erste Datei beinhaltet die Federmodellierung. In dieser können die physikalischen und geometrischen Eigenschaften einer Feder eingeben werden. Aus diesen werden dann die Eigenmoden berechnet. Das Script erstellt anschließend eine Impulsantwort, die dann zur Durchführung einer Faltung mit einer Audio-Eingangsdatei verwendet werden kann.

Die Faltung wird im zweiten Skript durchgeführt. Das zweite Skript ist optional und wurde nur zu Präsentationszwecken und für Tests geschrieben, um die Tonausgabe des modellierten Systems zu überprüfen.

3.1 Script 1: Synthese einer Impulsantwort

3.1.1 Abtastrate - Abtastzeit

Zunächst wird sowohl die Abtastrate f_s, mit einer in der Audiosignalverarbeitung Standard Abtastrate von 48 kHz, als auch deren Kehrwert die Abtastzeit T_s definiert.

```
%%  TEIL 1
%%% Sonstiges

clear all; home; close all; clc;

Fs = 48000;
Ts = 1/Fs;
```

3.1.2 Physikalische und geometrische Eigenschaften der Feder

Im folgenden Abschnitt wird ein Struktur *spring_parameter* erstellt. In dieser werden alle Eigenschaften einer Feder eingetragen. Dieses sind zum einen geometrische Eigenschaften wie u.a. die Länge des Helix, Durchmesser des Drahtes und Helix oder auch die Anzahl der Windungen. Aus diesen wird am Ende die Gesamtlänge des abgewickelten Drahtes berechnet.

$$L = \sqrt{\left(2\pi N \frac{d_{Helix}}{2}\right)^2 + (l_{Helix})^2} \tag{17}$$

Auch werden materialabhängige Eigenschaften wie Dichte und das Elastizitätsmodul definiert. Die meisten der Parameter werden am Ende zur Übersicht in der Variable κ (*siehe Dispersion*) zusammengefasst.

Abbildung 13: Olson X-82 - Spring Reverb [Bilbao, Parker]

Die Werte für die Modellierung enthalten die Parameter der Feder *Olson X-82*. Deren Eigenschaften konnten wir aus dem Anhang zum Paper *Spring Reverberation: A Physical Perspective* (J. Parker and Bilbao 2009) übernehmen.

```
%%  TEIL 2
%%% Physikalische Eigenschaften | Physical Parameters

% Hier werden die Eigenschaften eines Feders eingetragen |
% The properties of a given spring are entered here.
% Model: Olson X-82

%Laenge des Helix der Feder | Length of Helix[m]
spring_parameter.h_length = 0.065;

%Durchmesser des Helix der Feder | Helix  Diameter[m]
spring_parameter.h_diam = 0.0054;

%Durchmesser des Drahts | Diameter of the wire[m]
spring_parameter.f_diam = 0.00035;

%Anzahl der Windungen | Number of Turns of the Spring[1]
spring_parameter.turns = 148;

%Flaeche Querschnitt des Drahts | Area of round Wire[m^2]
spring_parameter.A = pi*(spring_parameter.f_diam/2)^2;

%Dichte | Density of Steel[kg*m^-3]
spring_parameter.rho = 7800;

%Elastizitaetsmodul | Young's Modulus of Steel[N*m^-2]
spring_parameter.E = 2^11;

%Flaechentraegheitsmoment | Second Moment of Area[m^4]
spring_parameter.I = (pi*(spring_parameter.h_diam/2)^4)/4;

%Laenge des gesamten Drahts | Length of the whole wire[m]
spring_parameter.L = sqrt((2*pi*spring_parameter.turns*...
    spring_parameter.h_diam/2)^2+spring_parameter.h_length^2);
```

```
% spring_parameter.L = 10

% Alle Eigenschaften in einer Variable gesammelt |
% All Parameter summarized in One Variable 'kappa'[m^2 * s^-1]
spring_parameter.kappa = sqrt((spring_parameter.E * spring_parameter.I)...
/(spring_parameter.rho * spring_parameter.A));
```

3.1.3 Resonanzfrequenzen

In diesem Teil werden die Resonanzfrequenzen der Modenformen berechnet und in dem Vektor $f_{r_{all}}$ gespeichert. In unserem konkreten Beispiel waren es 2744 Modenformen im hörbaren Bereich (bis 16 kHz). Anschließend wird, um Rechenaufwand zu sparen, nur ein Teil der Resonanzfrequenzen verwendet. Hier sind es 204 Stück.

Im Vektor $fr_{Colundi}$ wurden zu Testzwecken nur 10 Resonanzfrequenzen gespeichert.

Die Anzahl der Resonanzfrequenzen hat große Auswirkung auf den Klang und die Qualität des Hall Effektes und wird später in der Diskussion noch näher erläutert.

```
%%  TEIL 3
%%% Resonanzfrequenzen | Resonance Frequencies

%Vektor der Modenformen | Linearly spaced vector of modes
n = linspace(1,2744, 2744);

%Vektor der Resonanzfrequenzen | Vector of resonance frequencies
fr_all = 0.5*pi*spring_parameter.kappa*spring_parameter.L^(-2)*n.^2;
fr = fr_all(1:24:2454);


%Vector mit 10 Colundi Frequenzen
fr_colundi = zeros(1,10);
fr_colundi(1,1)= fr_all(1,27);
fr_colundi(1,2)= fr_all(1,43);
fr_colundi(1,3)= fr_all(1,75);
fr_colundi(1,4)= fr_all(1,112);
fr_colundi(1,5)= fr_all(1,158);
fr_colundi(1,6)= fr_all(1,295);
fr_colundi(1,7)= fr_all(1,473);
fr_colundi(1,8)= fr_all(1,610);
fr_colundi(1,9)= fr_all(1,1028);
fr_colundi(1,10)= fr_all(1,1226);
```

3.1.4 Funktion zur Erzeugung von Übertragungsfunktion

Mit der Funktion *create_TF*, die sich Matlab üblich am Ende des Skripts befindet, wird eine Übertragungsfunktion eines Hochpasses 2. Ordnung erstellt.

Die Funktion benötigt als Eingänge die Resonanzfrequenz des Hochpasses und den Dämpfungsfaktor. Die Funktion berechnet das von den beiden Variablen abhängige Nennerpolynom und gibt abschließend die Übertragungsfunktion als *transfer function model* zurück.

```
%%  TEIL 6
%%% Algorithmus zur Erzeugung einer Uebertragungsfunktion | Algorithm to create transfer function

```

```
function[S_HP2] = create_TF(resonance_freq,damping_factor)

%     arguments
%         resonance_freq double
%         damping_factor double
%     end

    w_k=2*pi*resonance_freq;
    d=damping_factor;

    num=[1 0 0];
    den = [1 2*w_k*d w_k^2];
    S_HP2=tf(num, den);
end
```

3.1.5 Erstellen der Übertragungsfunktionen und Impulsantworten

In diesem Abschnitt wird zunächst der Dämpfungsfaktor fest mit einem Wert für alle Frequenzen definiert. Die Dämpfung hat einen wesentlich Einfluss auf den klanglichen Charakter des Modells.

In der ersten Schleife werden dann mit der Funktion *create_TF* die kontinuierlichen Hochpass Übertragungsfunktionen in Abhängigkeit der berechneten Resonanzfrequenzen erstellt. Anschließend werden sie noch mit Hilfe der bilinearen Transformation mit der Funktion *c2d* in den diskreten Bereich gewandelt. Sowohl die kontinuierliche als auch die diskrete Übertragungsfunktionen werden jeweils zur Übersicht in einer Struktur gespeichert.

In der zweiten Schleife wird über die Funktion *impulse* die Impuls-Antworten der einzelnen Pässe generiert und in Strukturen gespeichert. Anschließend werden diese summiert und auf 1 normiert.

```
%%  TEIL 4
%%% Uebertragungsfunktionen und Impulsantworten | Transfer Functions and Impulse Responses

%Hier werden die Ubertragungsfunktionen und Impulsantworten berechnet

%Daempfung | Damping factor
D = 0.001;   % Beliebig

%Erzeugung der Uebertragungsfunktionen | Generation of transfer functions

%Optimales Design
for k = 1:length(fr)
    name = strcat('G',num2str(k));
    HP2_Order_continous.(name) = create_TF(fr(1,k), D);                          % Continuous
    HP2_Order_discrete.(name) = HP2_Order_continous.(name);
    HP2_Order_discrete.(name) = c2d(HP2_Order_discrete.(name),Ts, 'Tustin');     % Discrete
end


%10 Colundi Frequencies Design
% for k=1:10
%     name = strcat('G',num2str(k));
%     HP2_Order_continous.(name) = create_TF(fr_colundi(1,k), D);
%     HP2_Order_discrete.(name) = HP2_Order_continous.(name);
%     HP2_Order_discrete.(name)= c2d(HP2_Order_discrete.(name),Ts, 'Tustin');
```

```
% end

%Erzeugung der Impulsantworten | Generation of impulse responses
for k = 1:length(fr)
    name = strcat('G',num2str(k));
    ir_continous.(name)=impulse(HP2_Order_continous.(name),t);
    ir_discrete.(name)=impulse(HP2_Order_discrete.(name),t);
end

%Addieren aller Impulsantworten | Sum of all impulse responses
ir_continous_sum = zeros(length(ir_continous.G1),1);
ir_discrete_sum = zeros(length(ir_discrete.G1),1);

for k = 1:length(fr)
    name = strcat('G',num2str(k));
    ir_continous_sum = ir_continous_sum+ir_continous.(name);
    ir_discrete_sum = ir_discrete_sum+ir_discrete.(name);
end

%Normierung
ir_continous_sum_norm = ir_continous_sum/norm(ir_continous_sum);
ir_discrete_sum_norm = ir_discrete_sum/norm(ir_discrete_sum);
```

3.1.6 Speicherung im WAV-Format

Hier wird die zuvor erstellte Impulsantwort über die Funktion *audiorwrite* in eine WAV Datei im Workspace-Ordner gespeichert.

```
%%  TEIL 5
%%% Speicherung als .wav Datei | Export of summed impulse response as .wav file

%Die Impulsantwort wird als .wav datei exportiert zur weiteren Benutzung |
audiowrite('IR_discrete.wav', ir_discrete_sum_norm, Fs);
```

3.1.7 Grafische Darstellung der Impulsantworten

Im letzten Teil des Scripts werden die unnormierten und normierten Impulsantworten der kontinuierlichen und diskreten Systeme geplottet.

Anschließend wird mittels eine Schnellen Fourier Transformation ein Bodediagramm des Gesamtsystems erzeugt.

```
%%  TEIL 7
%%% Grafische Darstellung der Impulsantworten

% Hier werden die Impulsantwort und Bode-Diagramm des Systems dargestellt |
% Impulse response und Bode diagram are being plotted here

%Impulsantwort
figure(2)

subplot(2,2,1)
plot( t , ir_continous_sum);
axis([0 1 -1*10^4 10^4])
```

```
grid on;
title('IR - Continous')

subplot(2,2,2)
plot( t ,ir_continous_sum_norm);
grid on;
axis([0 1 -0.25 0.25])
title('IR - Continous (normalized)')


subplot(2,2,3)
plot( t , ir_discrete_sum);
axis([0 1 -1*10^4 10^4])
grid on;
title('IR - Discrete')

subplot(2,2,4)
plot( t , ir_discrete_sum_norm);
grid on;
axis([0 1 -0.25 0.25])
title('IR - Discrete (normalized)')

%Bode-diagramm
fft_IR_discrete = fft(ir_discrete_sum)*Ts;

fmax = 1/(2.*Ts);
df=1/((sample_N-1)*Ts);
f=-fmax:df:fmax;

Sshift = fftshift(fft_IR_discrete);
Skomp = transpose(Sshift).*exp(-j*2*pi*f*(Ts/2));

mag = 20*log10(abs(Skomp));
phase = rad2deg(angle(Skomp));

figure (3)

subplot(2,1,1)
semilogx(f,mag)
grid on
title('Amplitude')
xlabel('f in Hz')
ylabel('|Mag| (dB)')

subplot(2,1,2)
semilogx(f,phase)
title('Phase')
xlabel('f in Hz')
ylabel('Phase in degree')
```

3.2 Script 2: Faltung der Impulsantwort und Audioausgabe

Das zweite Script wurde hauptsächlich zur auditiven Beurteilung des Modells und zu Präsentationszwecken erstellt. Hier findet eine Faltung der generierten Impulsantworten mit einem Audio-Sample im WAV Format statt.

3.2.1 Laden des Audiosamples und grafische Darstellung

Zunächst wird über die Funktion *audioread* die Daten eines Audiosamples und die Abtastrate F_s eingelesen. Anschließend wird ein Vektor für die Samples und Einer für die Zeit erstellt. Danach erfolgt eine grafische Ausgabe.

```
%%  TEIL 2
%%% Importierung und grafische Darstellung der Audiodatei
%%% Import and graphic display of audio file

[in,Fs]=audioread('hier_audio_eingeben.wav');
in_left = in(:,1);
in_right = in(:,2);

t=[0:length(in)-1];      %Sample Axis
t2=[0:length(in)-1]*Ts; %Time Axis

figure(1)
plot(t,in);
xlabel('Sample [n]')
ylabel('Amplitude')
title('Waveform of selected sample [Sample Axis]')

plot(t2,in);
xlabel('Time [s]')
ylabel('Amplitude')
title('Waveform of selected sample [Time Axis]')
```

3.2.2 Laden der Impulsantwort

Im nächsten Schritt wird die Impulsantwort gelesen. Dieses funktioniert genauso wie beim Einlesen des Samples. Dieses kann auch in Stereo umgesetzt werden.

```
%%  TEIL 3
%%% Importierung der Impulsantwort | Import of impulse response

[h,Fs]=audioread('hier_IR_eingeben.wav');

% if stereo IR:
h_left = h(:,1);
h_right = h(:,1);
```

3.2.3 Faltung

Nun wird das Audiosample mit der Impulsantwort über die Funktion *conv* gefaltet. Dieses erfolgt sowohl für den rechten und linken Kanal eines Stereosignals. Anschließend wird über die Funktion *audiowrite* das gefaltete Signal in eine externe WAV-Datei geschrieben

```
%%  TEIL 4
%%% Faltung | Convolution

% out = conv(in,h);

%%% Stereo Faltung

out_left = conv(h_left,in_left);
out_right = conv(in_right,h_right);

out = [out_left,out_right];
audiowrite('hier_name_eingeben.wav',out,Fs)
```

3.2.4 Audioausgabe

Dieser auskommentierte Teil des Code wurde Anfangs zur Audioausgabe in Matlab benutzt. Es stellte sich später aber heraus, dass die Ausgabe und Beurteilung über einen externen Audioplayer komfortabler ist.

```
%%  TEIL 5
%%% Tonausgabe | Audio output

%Play original audio
% sound(in,Fs);

%Play processed sound
% sound(out,Fs);
```

3.2.5 Grafische Darstellung des gefalteten Signals

Abschließend wird das gefaltete Sample über die Funktion *plot* dargestellt,

```
%%  TEIL 6
%%% Grafische Darstellung der bearbeiteten Audiodatei | Grafic display of processed audio

figure(1)

plot(length(out),out);
xlabel('Sample [n]')
ylabel('Amplitude')
title('Waveform of convoluted sample [Sample Axis]')
hold on

plot(t2,out);
xlabel('Time [s]')
ylabel('Amplitude')
title('Waveform of convoluted sample [Time Axis]'))
```

4 Diskussion

Um das Verhalten des Modells zu testen waren Hörtest unabdingbar. Da wir zunächst das Modell nur mit 10 Resonanzfrequenzen implementiert hatten, waren die Ergebnisse sehr enttäuschend. Das Design schien eher einem Ringmodulator als einem Federhall zu ähneln. Es gab keinen halligen Charakter. Angesichts der starken mathematischen Vereinfachungen unserem Modells haben wir zwar nicht erwartet perfekte Ergebnisse zu erzielen aber dieses Resultat war sehr ungenügend.

Nach einem deutlich längeren Prozess, in dem wir viele Variablen des Modells immer wieder verändert hatten, erhielten wir deutlich bessere Ergebnisse.

Ein sehr wichtiger Faktor ist die Anzahl der benutzten Eigenmoden. Durch deren Verzwanzigfachung auf eine Anzahl von über 200 Moden, ließen sich deutlich realistischere Ergebnisse erzielen.

Auch haben wir festgestellt, dass es sehr wichtig ist niedrige Eigenmoden, die nicht im hörbaren Bereich, also kleiner als 20Hz sind, zu berücksichtigen. Dadurch erhält man eine deutlich längere Nachhallzeit. Dieses ist auf deren größer Periodendauer zurückzuführen, auf die sich die höhren Frequenzen aufmodulieren.

Weitere wichtige Variablen die große Auswirkung auf den Klang hatten, waren zum einen die Gesamtlänge der abgewickelten Feder und zum anderen der Dämpfungsfaktor. Mittels einer sehr geringen Dämpfung (D = 0.001) und einer großen Länge (L > 10m), konnten wir die Ergebnisse deutlich steigern bzw. den Effekt auch überhöhen. Wir erhielten teilweise *laserartige* Sounds, in denen sehr gut die akustische Dispersion zu hören ist.

Auch die Plots der Impulsantworten die wir generierten, lassen sich quantitativ mit realen Messungen von Federhallgeräten vergleichen und zeigen ähnlichen Charakter.

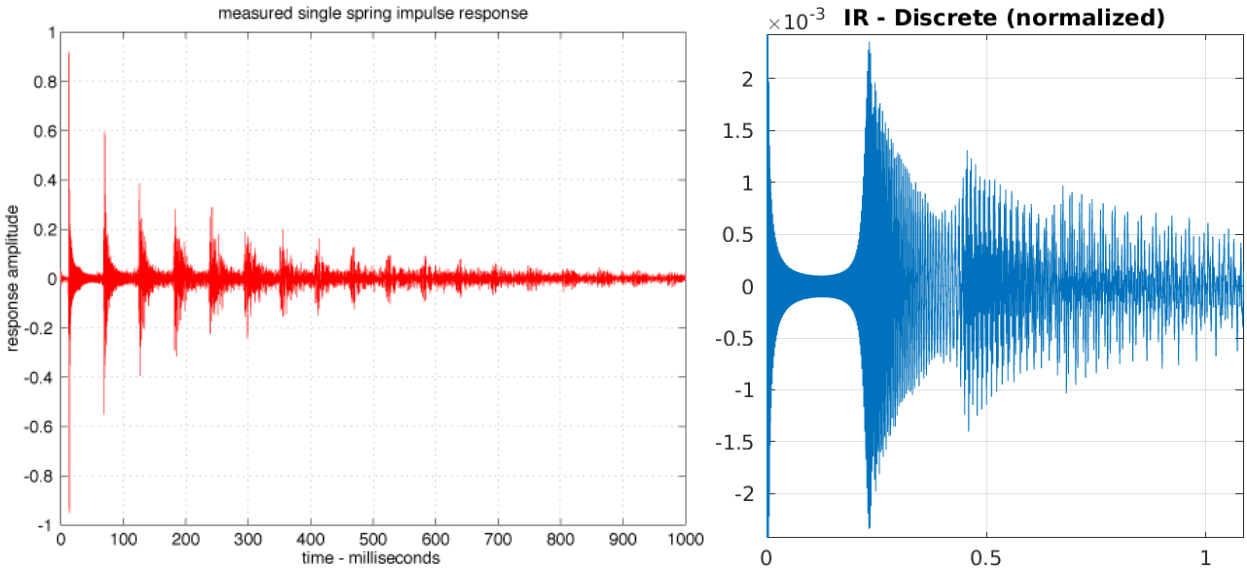


Abbildung 14: Impulsantwort einer Messung (rot) und des Modells (blau)

Mittels einer Schnellen Fourier Transformation in Matlab konnten wir außerdem ein Bodediagramm des Modells erstellen. In diesem lässt sich schön das fast schon chaotische Verhalten erkennen, welches sich auch an realen Federhall Systemen beobachten lässt. In den jeweiligen Resonanzfrequenzen kommt es auch auf Grund der geringen Dämpfung zu sehr starken Verstärkungen. In Realen Systemen sind dieses Teilweise wenige Hertz an Unterschied um dieses Verhalten zu provozieren.

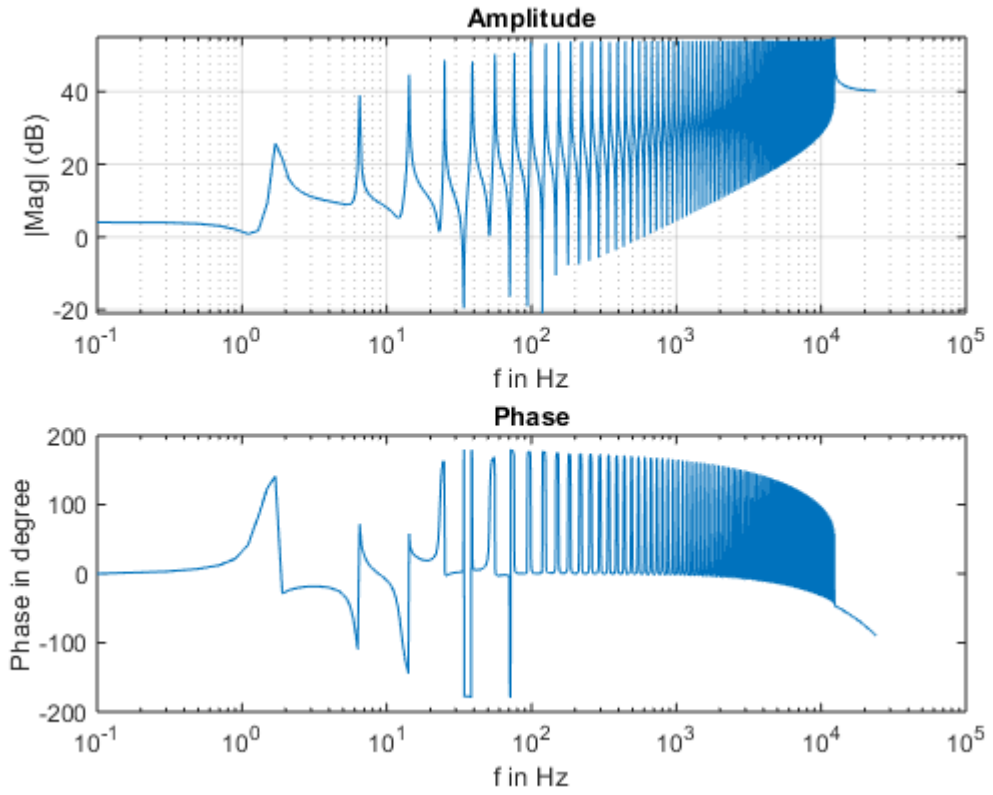


Abbildung 15: Bode-Diagramm des Models

Allerdings sollte beachtet werden, dass ein realer Federhall sich noch deutlich anders verhält als unser Modell. Unser Modell in dieser Implementierung ist daher eher mit einem sehr langen Draht zu vergleichen.

4.1 Fehler in der Parameterliste

Erst weit nach der Implemtierung und Testen des Algorthmus ist ein Fehler innerhalb der genutzten Parameter aufgefallen. Das Elastizitätsmodul von Stahl hatte in unseren Tests einen Betrag von $2^{11} = 2,048kPa$. Der eigentlich Wert von Stahl beträgt aber $2 \cdot 10^{11} = 200GPa$. Dieses ist eine Differenz von 8 Zehnerpotenzen. Damit entspricht das von uns modellierte Material noch nicht einmal den Werten von Gummi oder Kautschauk (Wikipedia 2021).

Diesen vermutliche Tippfehler hatten wir aus den Paper *Spring Reverberation: A Physical Perspective* (J. Parker and Bilbao 2009) übernommen.

Da der Faktor sehr groß ist, sind die Auswirkungen natürlich enorm. Mit dem korrekten Wert würden in den hörbaren Frequenzbereich viel weniger Eigenmoden fallen.

Um zumindest sound-technisch auf ähnliche bzw. gleiche Ergebnisse zu kommen, könnte man aber die Länge L ,des abgewickelten Drahtes vergrößern. Da dieser quadratisch in den Nenner der Funktion zur Berechnung der Eigenmoden eingeht, das Elastizitätsmodul jedoch nur linear in den Zähler.

4.2 Zukünftige Verbesserungen

Verbessern lässt sich das Modell an vielen Enden. Hier seien nur einige, unserer Meinung nach Wichtige, genannt.

Zu einem könnte die Anzahl der Modenformen weiter erhöhen. Dieses ist natürlich auch mit Blick auf eine Echtzeit-Implementierung sehr rechenaufwendig. Vielleicht wäre in dem Zusammenhang auch ein Ansatz sich über ein kluge Verteilung der ausgewählten Resonanzfrequenzen Gedanken zu machen.

Ein weiterer wichtiger Punkt wäre, da zurzeit nur Transversalwellen der Feder modelliert werde auch die Betrachtung Longitudinal- bzw. Torsionswellen.

Auch sollte das Modell der partiellen Differentialgleichung durch ein Dämpfungsmodell erweitert werden. So würde man eine frequenzabhängige Dämpfung erhalten. Dieses ist deutlich realistischer als die konstante Dämpfung über den gesamten Frequenzbereich die momentan Implementiert ist (Bilbao 2013).

Auch wäre es Interessant die elektrische Beschaltung der Feder, bestehend aus den Verstärkern und den Spulen, zu modellieren. Diese Beschaltung hat in realen Systemen ein Bandpassverhalten zur Folge. Dieses ist in unserem Modell bisher nicht zu sehen. Auch könnten man die Nicht-Linearitäten der Verstärker oder der Spule betrachten.

Außerdem fehlt dem Modell für eine praktische Nutzung bisher jegliche Funktionalität. Wichtig wäre es zum Beispiel zunächst einen Mixer zu implementieren, mit dem sich das Verhältnis zwischen Originalsignal (Dry) und Effektsignal (Wet) einstellen lässt.

5 Zusammenfassung

In dieser Projektarbeit wurde erfolgreich die Modal-Synthese eines Federhall durchgeführt. Es wurde ein einfaches aber geeignetes physikalisches Modell ermittelt, dass viele geometrischen und materialabhängigen Variablen einer Feder beachtet. Aus diesem wurden dann die Eigenfrequenzen des Systems berechnet und durch Aufsummierung vieler digitaler Biquad-IIR-Filter wurde das System anschließend implementiert. Durch das Experimentieren mit dem vom System gegebenen Variablen war es uns möglich ein zufriedenstellendes Ergebnis zu erreichen.

Zeitaufwändig war es in diesem Projekt sich in die vorhandene Literatur einzulesen und anschließend die mathematischen und physikalischen Grundlagen zu erarbeiten. Gerade die mechanische Modellierung waren für alle beteiligten Neuland und kosteten deshalb viel Zeit. Auch bei der Implementierung der digitalen Signalverarbeitung gab es die ein oder ander Schwierigkeit. Zum Beispiel haben wir zunächst versucht mit der *tf* Funktion in Matlab eine Gesamtübertragungsfunktion des Systems zu erzeugen. Dieses führte allerdings zu Koeffizienten die im double Format nicht mehr darstellbar waren.

Gerne hätten wir das Modell noch verbessert und es in Echtzeit als Plug-in oder auf einen Embedded System umgesetzt. Aufgrund der Komplexität fehlte dafür aber leider die Zeit. Für eine schnelle Umsetzung des Algorithmus in C/C++ könnte als Domänenspezifische Sprache die Programmiersprache Faust verwendet werden. Faust ist eine funktionale Programmiersprache für die Echtzeit Signalverarbeitung von Audio-Signalen. Das Modell könnte relativ einfach umgesetzt werden und der Faust-Compiler erzeugt daraus gut optimierten C/C++ Code. Dieser lässt sich dann mittels sogenannter *architecture files* auf die gewünschte Zielplattform (u.a. VST, Android, Pure Data, Raspberry Pi) portieren.

6 Quellen

Abbildungsverzeichnis

---

0. Titelbild: Patent 2230836A - Electrical Musical Instrument, (Hammond 1939)
1. Screenshot vom Programm ReaVerb
2. https://www.nachhallsanierung.de/nachhall-verstehen/nachhall-reduzieren/
3. https://www.amazona.de/wp-content/uploads/2014/07/reverbtank_1.gif
4. Erstellt mit draw.io
5. https://mysoundbook.eu/laurens-hammond-der-erfinder-der-hammond-orgel/
6. https://en.wikipedia.org/wiki/Helix_angle#/media/File:Helix_angle.jpg
7. Spring Reverberation: A Physical Perspective, (J. Parker and Bilbao 2009)
8. Erstellt mit Desmos
9. Erstellt mit Matlab
10. Erstellt mit draw.io
11. https://de.wikipedia.org/wiki/Bilineare_Transformation_(Signalverarbeitung)
12. Signale und Systeme – Einheit 11: LTI-Systeme im Zeitbereich (Jakob 2020)
13. Dispersive Systems in Musical Audio Signal Processing (J. D. Parker 2013)
14. Rot: https://www.uaudio.com/webzine/2006/april/index2.html | Blau: erstellt mit Matlab
15. erstellt mit Matlab

6.1 Code

6.1.1 Script 1: Synthese einer Impulsantwort - spring_ir.m

```
%%%% ENTWURF EINES FEDERHALLS - DESIGN OF A SPRING REVERBERATOR %%%%

%%% Beuth Hochschule fuer Technik Berlin
%%% Audiotechnik Uebung - WiSe 20/21
%%% Gruppe 7: Jan Abel, Pierre-Hugues Hadacek, Steven Thiele
%%% Dozent: Herr Prof. Andre Jakob

%%  TEIL 1
%%% Sonstiges

clear all; home; close all; clc;

Fs = 48000;
Ts = 1/Fs;

%dummy variable
sample_N = 240000;
t = [0:Ts:(sample_N-1)*Ts];

%%  TEIL 2
%%% Physikalische Eigenschaften | Physical Parameters

% Hier werden die Eigenschaften eines Feders eingetragen |
% The properties of a given spring are entered here.
% Model: Olson X-82

%Laenge des Helix der Feder | Length of Helix[m]
spring_parameter.h_length = 0.065;

%Durchmesser des Helix der Feder | Helix  Diameter[m]
spring_parameter.h_diam = 0.0054;

%Durchmesser des Drahts | Diameter of the wire[m]
spring_parameter.f_diam = 0.00035;

%Anzahl der Windungen | Number of Turns of the Spring[1]
spring_parameter.turns = 148;

%Flaeche Querschnitt des Drahts | Area of round Wire[m^2]
spring_parameter.A = pi*(spring_parameter.f_diam/2)^2;

%Dichte | Density of Steel[kg*m^-3]
spring_parameter.rho = 7800;

%Elastizitaetsmodul | Young's Modulus of Steel[N*m^-2]
spring_parameter.E = 2^11;

%Flaechentraegheitsmoment | Second Moment of Area[m^4]
spring_parameter.I = (pi*(spring_parameter.h_diam/2)^4)/4;
```

```
50
51  %Laenge des gesamten Drahts | Length of the whole wire[m]
52  spring_parameter.L = sqrt((2*pi*spring_parameter.turns*...
53      spring_parameter.h_diam/2)^2+spring_parameter.h_length^2);
54  % spring_parameter.L = 10
55
56  % Alle Eigenschaften in einer Variable gesammelt |
57  % All Parameter summarized in One Variable 'kappa'[m^2 * s^-1]
58  spring_parameter.kappa = sqrt((spring_parameter.E * spring_parameter.I)...
59  /(spring_parameter.rho * spring_parameter.A));
60
61  %%  TEIL 3
62  %%% Resonanzfrequenzen | Resonance Frequencies
63
64  %Vektor der Modenformen | Linearly spaced vector of modes
65  n = linspace(1,2744, 2744);
66
67  %Vektor der Resonanzfrequenzen | Vector of resonance frequencies
68  fr_all = 0.5*pi*spring_parameter.kappa*spring_parameter.L^(-2)*n.^2;
69  fr = fr_all(1:24:2454);
70
71
72  %Vector mit 10 Colundi Frequenzen
73  fr_colundi = zeros(1,10);
74  fr_colundi(1,1)= fr_all(1,27);
75  fr_colundi(1,2)= fr_all(1,43);
76  fr_colundi(1,3)= fr_all(1,75);
77  fr_colundi(1,4)= fr_all(1,112);
78  fr_colundi(1,5)= fr_all(1,158);
79  fr_colundi(1,6)= fr_all(1,295);
80  fr_colundi(1,7)= fr_all(1,473);
81  fr_colundi(1,8)= fr_all(1,610);
82  fr_colundi(1,9)= fr_all(1,1028);
83  fr_colundi(1,10)= fr_all(1,1226);
84
85  %%  TEIL 4
86  %%% Uebertragungsfunktionen und Impulsantworten | Transfer Functions and Impulse Responses
87
88  %Hier werden die Ubertragungsfunktionen und Impulsantworten berechnet
89
90  %Daempfung | Damping factor
91  D = 0.001;   % Beliebig
92
93  %Erzeugung der Uebertragungsfunktionen | Generation of transfer functions
94
95  %Optimales Design
96  for k = 1:length(fr)
97      name = strcat('G',num2str(k));
98      HP2_Order_continous.(name) = create_TF(fr(1,k), D);                          % Continuous
99      HP2_Order_discrete.(name) = HP2_Order_continous.(name);
100     HP2_Order_discrete.(name) = c2d(HP2_Order_discrete.(name),Ts, 'Tustin');     % Discrete
101 end
```

```
102
103
104 %10 Colundi Frequencies Design
105 % for k=1:10
106 %     name = strcat('G',num2str(k));
107 %     HP2_Order_continous.(name) = create_TF(fr_colundi(1,k), D);
108 %     HP2_Order_discrete.(name) = HP2_Order_continous.(name);
109 %     HP2_Order_discrete.(name)= c2d(HP2_Order_discrete.(name),Ts, 'Tustin');
110 % end
111
112 %Erzeugung der Impulsantworten | Generation of impulse responses
113 for k = 1:length(fr)
114     name = strcat('G',num2str(k));
115     ir_continous.(name)=impulse(HP2_Order_continous.(name),t);
116     ir_discrete.(name)=impulse(HP2_Order_discrete.(name),t);
117 end
118
119 %Addieren aller Impulsantworten | Sum of all impulse responses
120 ir_continous_sum = zeros(length(ir_continous.G1),1);
121 ir_discrete_sum = zeros(length(ir_discrete.G1),1);
122
123 for k = 1:length(fr)
124     name = strcat('G',num2str(k));
125     ir_continous_sum = ir_continous_sum+ir_continous.(name);
126     ir_discrete_sum = ir_discrete_sum+ir_discrete.(name);
127 end
128
129 %Normierung
130 ir_continous_sum_norm = ir_continous_sum/norm(ir_continous_sum);
131 ir_discrete_sum_norm = ir_discrete_sum/norm(ir_discrete_sum);
132
133 %%  TEIL 5
134 %%% Speicherung als .wav Datei | Export of summed impulse response as .wav file
135
136 %Die Impulsantwort wird als .wav datei exportiert zur weiteren Benutzung |
137 audiowrite('IR_discrete.wav', ir_discrete_sum_norm, Fs);
138
139
140 %%  TEIL 7
141 %%% Grafische Darstellung der Impulsantworten
142
143 % Hier werden die Impulsantwort und Bode-Diagramm des Systems dargestellt |
144 % Impulse response und Bode diagram are being plotted here
145
146 %Impulsantwort
147 figure(2)
148
149 subplot(2,2,1)
150 plot( t , ir_continous_sum);
151 axis([0 1 -1*10^4 10^4])
152 grid on;
153 title('IR - Continous')
```

```
154
155 subplot(2,2,2)
156 plot( t ,ir_continous_sum_norm);
157 grid on;
158 axis([0 1 -0.25 0.25])
159 title('IR - Continous (normalized)')
160
161
162 subplot(2,2,3)
163 plot( t , ir_discrete_sum);
164 axis([0 1 -1*10^4 10^4])
165 grid on;
166 title('IR - Discrete')
167
168 subplot(2,2,4)
169 plot( t , ir_discrete_sum_norm);
170 grid on;
171 axis([0 1 -0.25 0.25])
172 title('IR - Discrete (normalized)')
173
174 %Bode-diagramm
175 fft_IR_discrete = fft(ir_discrete_sum)*Ts;
176
177 fmax = 1/(2.*Ts);
178 df=1/((sample_N-1)*Ts);
179 f=-fmax:df:fmax;
180
181 Sshift = fftshift(fft_IR_discrete);
182 Skomp = transpose(Sshift).*exp(-j*2*pi*f*(Ts/2));
183
184 mag = 20*log10(abs(Skomp));
185 phase = rad2deg(angle(Skomp));
186
187 figure (3)
188
189 subplot(2,1,1)
190 semilogx(f,mag)
191 grid on
192 title('Amplitude')
193 xlabel('f in Hz')
194 ylabel('|Mag| (dB)')
195
196 subplot(2,1,2)
197 semilogx(f,phase)
198 title('Phase')
199 xlabel('f in Hz')
200 ylabel('Phase in degree')
201
202 %%  TEIL 6
203 %%% Algorithmus zur Erzeugung einer Uebertragungsfunktion | Algorithm to create transfer function
204
205 function[S_HP2] = create_TF(resonance_freq,damping_factor)
```

```
206
207 %     arguments
208 %         resonance_freq double
209 %         damping_factor double
210 %     end
211
212     w_k=2*pi*resonance_freq;
213     d=damping_factor;
214
215     num=[1 0 0];
216     den = [1 2*w_k*d w_k^2];
217     S_HP2=tf(num, den);
218 end
```

6.1.2 Script 2: Faltung der Impulsantwort und Audioausgabe - spring_conv.m

```
1  %%%% ENTWURF EINES FEDERHALLS - DESIGN OF A SPRING REVERBERATOR %%%%
2
3  %%% Beuth Hochschule fÜr Technik Berlin
4  %%% Audiotechnik Übung - WiSe 20/21
5  %%% Gruppe 7: Jan Abel, Pierre-Hugues Hadacek, Steven Thiele
6  %%% Dozent: Herr Prof. Andre Jakob
7
8  %%  TEIL 1
9
10 clear all; home; close all; clc;
11
12 %%  TEIL 2
13 %%% Importierung und grafische Darstellung der Audiodatei
14 %%% Import and graphic display of audio file
15
16 [in,Fs]=audioread('hier_audio_eingeben.wav');
17 in_left = in(:,1);
18 in_right = in(:,2);
19
20 t=[0:length(in)-1];      %Sample Axis
21 t2=[0:length(in)-1]*Ts; %Time Axis
22
23 figure(1)
24 plot(t,in);
25 xlabel('Sample [n]')
26 ylabel('Amplitude')
27 title('Waveform of selected sample [Sample Axis]')
28
29 plot(t2,in);
30 xlabel('Time [s]')
31 ylabel('Amplitude')
32 title('Waveform of selected sample [Time Axis]')
33
34 %%  TEIL 3
35 %%% Importierung der Impulsantwort | Import of impulse response
36
```

```
[h,Fs]=audioread('hier_IR_eingeben.wav');

% if stereo IR:
h_left = h(:,1);
h_right = h(:,1);

%%  TEIL 4
%%% Faltung | Convolution

% out = conv(in,h);

%%% Stereo Faltung

out_left = conv(h_left,in_left);
out_right = conv(in_right,h_right);

out = [out_left,out_right];
audiowrite('hier_name_eingeben.wav',out,Fs)

%%  TEIL 5
%%% Tonausgabe | Audio output

%Play original audio
% sound(in,Fs);

%Play processed sound
% sound(out,Fs);

%%  TEIL 6
%%% Grafische Darstellung der bearbeiteten Audiodatei | Grafic display of processed audio

figure(1)

plot(length(out),out);
xlabel('Sample [n]')
ylabel('Amplitude')
title('Waveform of convoluted sample [Sample Axis]')
hold on

plot(t2,out);
xlabel('Time [s]')
ylabel('Amplitude')
title('Waveform of convoluted sample [Time Axis]'))
```

6.2 Literaturverzeichnis

Bilbao, Stefan. 2013. “NUMERICAL SIMULATION OF SPRING REVERBERATION,” 8.

Fletcher, N. H, T Tarnopolskaya, and F. R de Hoog. 2001. “Wave Propagation on Helices and Hyperhelices: A Fractal Regression.” *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 457 (2005): 33–43. https://doi.org/10.1098/rspa.2000.0654.

Hammond, Laurens. 1939. “Patent - US2230836A - Electrical Musical Instrument,” 13.

Jakob, Andre. 2020. “Signale und Systeme – Einheit 11: LTI-Systeme im Zeitbereich.”

Parker, Julian D. 2013. “Dispersive Systems in Musical Audio Signal Processing,” 54.

Parker, Julian, and Stefan Bilbao. 2009. “Spring Reverberation: A Physical Perspective,” 6.

Smith, Julius O. 2021. “JOS Home Page.” 2021. https://ccrma.stanford.edu/~jos/.

Sorokin, S. V. 2009. “Linear Dynamics of Elastic Helical Springs: Asymptotic Analysis of Wave Propagation.” *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 465 (2105): 1513–37. https://doi.org/10.1098/rspa.2008.0468.

Wikipedia. 2021. “Elastizitätsmodul - Wiki.” 2021. https://de.wikipedia.org/wiki/Elastizit%C3%A4tsmodul.