Dokumentation

der Seminararbeit aus

Algorithmen in Akustik und Computermusik 2 SE und UE
WS & SS 2007/2008

„Parametrierung von Raumimpulsantworten“

Christoph Frank 0430937
Benjamin Dietze 0473007

Abstract

Das zu erarbeitende Thema „Parametrierung von Raumimpulsantworten“ der VO und UE „Algorithmen 2“ des Instituts für elektronische Musik Graz IEM im Studienjahr 07/08 wird im Folgenden wie folgt dokumentiert.

Zum allgemeinen Verständnis wird auf die grundlegenden raumakustischen Größen, die zur Beschreibung von Raumimpulsantworten benötigt werden, eingegangen.

Anschließend gilt es die Definition von Impulsantworten und daraus folgend von Raumimpulsantworten – und was diese beschreiben – zu erörtern.

In weiterer Folge wird auf verschiedene Anregungssignale zur Impulsantwortmessung eingegangen und diese werden bezüglich ihrer Vor- und Nachteile, sowie die Implementierung in MATLAB beschrieben.

Die anschließende allgemeine Betrachtung der Definition der Nachhallzeit RT 60 sowie deren Ermittlung durch lineare Regression aus der sogenannten T_{20} bzw. T_{30} dient zum weiterfolgenden Verständnis der Berechnung der Nachhallzeit aus Raumimpulsantworten.

Daraufhin werden durch Energiebetrachtung der Raumimpulsantworten die allgemeine Energie-Abklingkurve – die sog. „EDC … Energy Decay Curve“ – sowie alle Energie-Abklingkurven über dem gemessenen Frequenzband Δf – die „EDR … Energy Decay Reliefs“ – beschrieben. In diesem Zuge wird ebenfalls auf allgemeine Probleme der Energiebetrachtung von Raumimpulsantworten eingegangen und die daraus resultierenden Probleme für die Ermittlung der raumakustischen Größen (bspw. Nachhallzeit RT60) erörtert.

Als Lösung der Probleme bei der Ermittlung von raumakustischen Größen aus der Energiebetrachtung von Raumimpulsantworten (EDC bzw. EDR) wird nun als Lösungsmöglichkeit die „Schröder Rückwärtsintegration“, deren Vor-/Nachteile und Implementierung in MATLAB besprochen.

Aus der Problematik der Auffindung optimaler Integrationsgrenzen für die „Schröder Rückwärtsintegration“ von Raumimpulsantworten wird auf die Implementierung spezieller Schätzalgorithmen (z.B. Lundeby Noise Floor Estimation), sowie deren Implementierung in MATLAB eingegangen.

Durch Vorteile der „Schröder Rückwärtsintegration“ wird folgend die Ermittlung der raumakustischen Parameter und das Erstellen von „Nachhallplateus“ beschrieben.

Als letzter Punkt dieser Dokumentation wird fertige Softwareimplementierung in MATLAB bzw. Pure Data erläutert.

Inhaltsverzeichnis

Raumakustische Größen

Im Bezug auf das zu erörternde Thema der „Parametrisierung von Raumimpulsantworten“ sind nur wenige raumakustischen Größen von Wichtigkeit, die zum Verständnis der Thematik beitragen.
Zu diesen raumakustischen Größen gehören der *Direktschall D*, die *frühen Reflexionen*, die *„Early Decay Time EDT“*, die *„Reverbation Time RT60“*, und der *„Noise Floor“*.

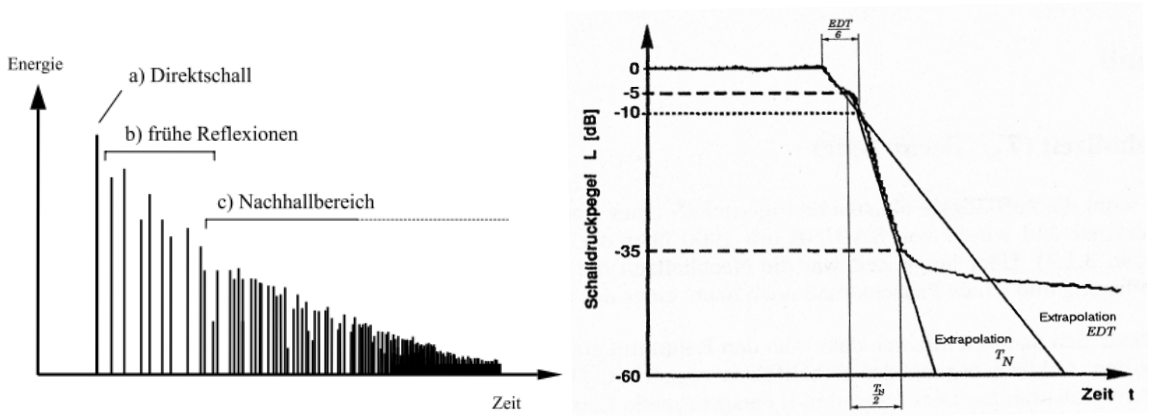


Abbildung 1: Reflexionsdiagramm(links) und Abklingkurve(rechts) eines Raumes

Reverbation Time RT60 (Nachhallzeit):

Die Nachhallzeit RT60 in der Raumakustik ist jene Zeit die nach Impulsanregung den Abfall des Gesamtschalldruckpegels von 0dB SPL bis -60dB SPL beschreibt[1].

Direktschall D:

Der Direktschall D beschreibt in der Raumakustik jenen abgestrahlten Schall einer im Raum befindlichen Schallquelle, der als erstes Hörereignis am Hör- bzw. Messort eintrifft (ohne Reflexionen). Der Direktschall D beschreibt damit maßgeblich die empfundene Herkunftsrichtung (Lokalisation) der Schallquelle im Raum (Haas- bzw. Präzedenz-Effekt). Damit kann der Direktschall (zumindest in großen Räumen) leicht aus dem Eintreffen der Schallereignisse am Hörort nachempfunden und somit bei der Betrachtung von Raumimpulsantwort detektiert werden [2].

Frühe Reflexionen:

Die frühen Reflexionen beschreiben in der Raumakustik jene reflektierten Schallereignisse einer im Raum befindlichen Schallquelle, die als erste Hörereignisse nach dem Direktschall D am Hör- bzw. Messort eintreffen (Reflexionen erster bzw. zweiter Ordnung). Die frühen Reflexionen sind dabei hauptverantwortlich für den Beitrag zur „empfundenen Räumlichkeit“ in einem Raum (vgl. EDT) [2].

Early Decay Time EDT (Anfangsnachhallzeit):

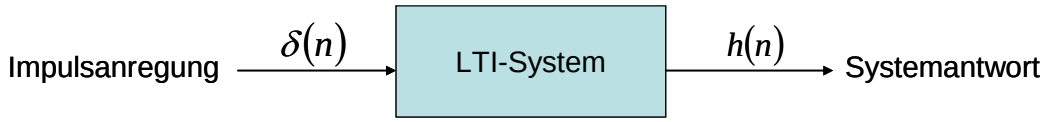
Da der Anfangsteil des Abklingvorgangs im Allgemeinen besser wahrgenommen wird als ein gesamter 60dB Abfall der Nachhallzeit (vgl. Definition RT60), dominiert dieser Anfangsteil auch den subjektiv empfundenen Nachhall. Die EDT wird aus dem ersten 10dB Abfall des Nachhalls durch Extrapolierung zum definierten 60dB Abfall der Nachhallzeit ermittelt [2].

Der Noise Floor NF:

Der Noise Floor NF beschreibt in der Raumakustik zunächst den stets vorhanden Grundgeräuschpegel im Raum. Dieser ist vor allem deshalb von Interesse, weil zur theoretischen Ermittlung der Nachhallzeit mindestens ein 60dB Abfall des Abklinkvorgangs bei Impulsanregung erreicht werden muss.
Weiters summiert sich zum Grundgeräuschpegel im Raum auch noch das Rauschen der gesamten Messkette (Mikrofon, Verstärker, …) und bringt damit weiteren Beitrag zum Noise Floor NF.
In der Praxis wird daher meist auf einen Schallpegelabfall bis -35dB bzw. bei extrem hohen Noise Floor auf einen Schallpegelabfall bis -25dB bei Impulsanregung wert gelegt und anschließend auf die Zeit des Gesamtnachhalls extrapoliert (vgl. T_{20} und T_{30}).

Impulsantworten

Aus raumakustischer Sicht ist die Impulsantwort die Darstellung des in einem Raum empfangenen Schalldrucks als Funktion der Zeit, bewirkt durch die Anregung des Raumes durch eine Diracsche Delta-Funktion (Impuls) [2],[3].



Formel 1:

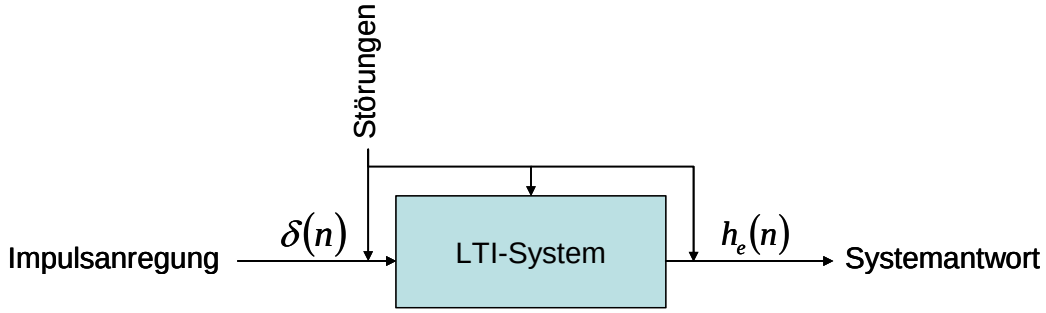
$$y(n) = \sum_{k=-\infty}^{\infty} x[k] h_k[n] \qquad \text{mit } h_k[n] = h_0[n-k]$$

Allgemein besagt die Signaltheorie, dass ein beliebiges lineares zeitinvariantes System (LTI System) vollständig durch seine Impulsantwort beschrieben werden kann.

Formel 2:

$$H(z) = \sum_{n=-\infty}^{\infty} h[n] z^{-n}$$

Dieser Sachverhalt gilt auch eingeschränkt für die Bestimmung von Raumimpulsantworten. Die Einschränkung von Raumimpulsantworten liegt nun genau darin, dass man mit einer einzigen Raumimpulsantwort niemals den ganzen Raum (das ganze System beschreiben kann).



Näherungsweise besteht daher die Möglichkeit über viele verschiedene Messpunkte im Raum zu mitteln und damit näher in die Richtung des realen Systems zu gelangen. Jedoch darf man auch niemals die Beeinflussung der Messeinrichtung auf die Messung und damit das System selbst außer Acht lassen.

Geeignete Anregungssignale

Am besten würde man Raumimpulsantworten durch Anregung des Raumes mit einem idealen Dirac-Impuls erhalten (Impuls, der unendlich schmal und unendlich hoch ist). Durch diese Vorgehensweise würde das Messsignal auch einer wirklichen Impulsantwort entsprechen.

Durch die Beschränktheit in der Dynamik (durch Grenzen der Elektronik und Mechanik) bei der Anregung des Systems (Lautsprecher oder Andere), lassen sich in der Realität keine idealen Dirac-Impulse erzeugen.

Daher liegt die Überlegung nahe, geeignete Anregungssignale zur Ermittlung von Raumimpulsantworten zu finden.

Nahezu ideale Raumimpulsantworten lassen sich nun in der Praxis mit Signalen wie *Exponentialsweep*, *MLS Pseudorauschen* oder ähnlichen Signalen erreichen. Die Anregung erfolgt dabei rein elektromechanisch (mit Lautsprechern).

Manuell lassen sich hingehen auch *Dirac-ähnliche Signale* erzeugen, die ebenfalls geeignete Signale zur Ermittlung von Raumimpulsantworten darstellen (Pistolenknall, Klatschen, Luftballon).

Es ist ausdrücklich darauf hinzuweisen, dass das in der „ÖNORM EN ISO 3382“ zur Messung von Nachhallzeit von Räumen beschriebene *Verfahren des abgeschalteten Rauschens* nicht zur Ermittlung von Raumimpulsantworten dienen kann. Dieser Sachverhalt wird nachfolgend erklärt.

Exponentieller Sweep:

Der exponentielle Sweep[4] ist ein Sinussignal, das über die Zeit exponentiell frequenzmoduliert wird.

Formel 3:

$$x(t) = \sin\left[A\left(e^{t/\tau} - 1\right)\right]$$

$$A = \frac{T\omega_1}{\ln\left(\omega_2/\omega_1\right)} \qquad \tau = \frac{T}{\ln\left(\omega_2/\omega_1\right)}$$

... ω_1 : Startfrequenz in Hz
... ω_2 : Stopfrequenz in Hz
... T : Signaldauer in Sekunden

Dabei ist zu jedem Zeitpunkt der Anregung des Systems durch den exponentiellen Sweep nur eine einzige definierte Frequenz vorhanden. Durch diese Tatsache lässt sich leicht viel Energie in das System einbringen, was dem Signalrauschabstand SNR und damit der Messung zu Gute kommt.

Die wichtige Tatsache, dass die exponentielle Frequenzmodulation des Sinussignals der logarithmischen Frequenzachse entspricht, kommt weiters der Anregung des Systems insofern zu Gute, dass tiefe Frequenzen tendenziell länger angeregt werden als hohe. Somit wird – bei geeigneter Parametrierung – ein eingeschwungener Zustand des Systems bei jeder Modulationsfrequenz sichergestellt (tiefe Frequenzen brauchen länger zum Einschwingen, als hohe Frequenzen).

Aus Sicht der Signaltheorie ergibt sich weiters der Vorteil, dass der inverse exponentielle Sweep dem inversen System entspricht und es gilt daher folgender Sachverhalt

Formel 4:

$$Y(f) = H(f) \cdot X(f) \rightarrow Y(f) \cdot X^{-1}(f) = H(f) \cdot \underbrace{X(f) \cdot X^{-1}(f)}_{=1} \text{ [4]}.$$

Wie zu erkennen ist, lässt sich durch Faltung mit dem inversen Sweep die Impulsantwort ermitteln. Da das Anregungssignal leicht zu invertieren ist, stellt dies einen entscheidenden Vorteil in der Implementierung des exponentiellen Sweeps dar.

In MATLAB wurde eine Funktion zu Erzeugung des exponentiellen Sweeps und dessen Inversen erzeugt („expsweep.m“ und „invexpsweep.m“). Im Folgenden ist die Funktion „expsweep.m“ und ein Plot des Ergebnisses dieser dargestellt, die der Implementierung oben erwähnter Formeln entspricht:

```
function vec=expsweep(fstart,fend,T,fs)

% vec=expsweep(fstart,fend,T,fs)

%T=T/fs;
w1=2*pi*fstart;
w2=2*pi*fend;

A=T*w1/log(w2/w1);
tau=T/log(w2/w1);

t=0:1/fs:T-1/fs;
vec=sin(A*(exp(t/tau)-1))';
```

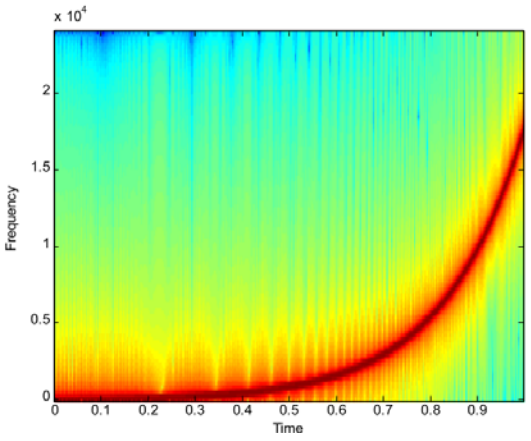


Abbildung 2: Zeit vs. Frequenz bei Sinussweeps [4]

Die Messung der Impulsantwort mit exponentiellem Sweep wurde wie folgt in der Funktion „impulse_exp.m“ realisiert.

```
%Messung der Impulsantwort mit 2s exponentiellem Sweep
function in = impulse_exp(fs,T);

%fs = Samplingfrequenz
%T = Dauer des gewählten Sweeps

if ~exist ('T')
    disp('Dauer des Exponentialssweeps 2s')
    T = 2;
end

%Erzeugung des exponentiellen Sweeps
sw=expsweep(20,20000,T,fs); %expsweep(fstart,fend,T,fs)
isw=invexpsweep(sw,20,20000,fs);
```

```
%Anregung mit exponentiellem Sweep
lag = 2*fs; %zu erwartende maximale Länge der Impulsantwort


%Messung der Impulsantwort
wavplay(sw, fs,'async');
y = wavrecord (length(sw)+lag, fs, 1);


%Daraus Berechnung der Impulsantwort
nges = length(sw)+length(isw)+lag;
in = real(ifft(fft(y,nges).*fft(isw,nges))) / length(sw);
in = in(length(sw):length(sw)+lag);

%Ausgabe der normierten Impulsantwort;
in = in./max(abs(in));

%Ausgabe des auf die Impulsantwort normierten Noisefloor
backnoise = backnoise./max(abs(in));
```

Zur geeigneten Anregung des Systems wird von einer Mindestlänge des Signals von zwei Sekunden ausgegangen.
Zunächst werden nach Aufruf der Funktion die gewünschten Anregungssignale („expsweep.m“ und „invexpsweep.m“) mit gewünschten Parametern erzeugt. Anschließend wird ein *lag* eingeführt, das der maximalen Länge der zu erwartenden Impulsantwort entspricht und zur weiteren Berechnung benötigt wird.
Die Gesamtlänge der Impulsantwort stellt sich nun aus der Länge des exponentiellen Sweeps selbst und dessen Inversen, sowie dem *lag* der zu erwartenden Maximallänge der Impulsantwort dar.
Die Impulsantwort ergibt sich nun durch Multiplikation der Fouriertransformierten (im Folgenden als FT bezeichnet) des aufgenommenen Signals Y mit der FT des inversen Sweeps im Frequenzbereich. Der Realteil der inversen FT und die Normierung mit der Länge des Anregungssignals ergibt nach geeignetem Abschneiden des Signals die gewünschte Impulsantwort. Durch das „Wegwerfen“ des Imaginärteils der inversen FT werden nichtlineare Verzerrungen eliminiert, da diese im Realteil nicht mehr enthalten sind.

Nachfolgend ist das Ergebnis einer guten Exponentialsweep-Anregung zu sehen, die mit unserer Implementierung realisiert wurde:

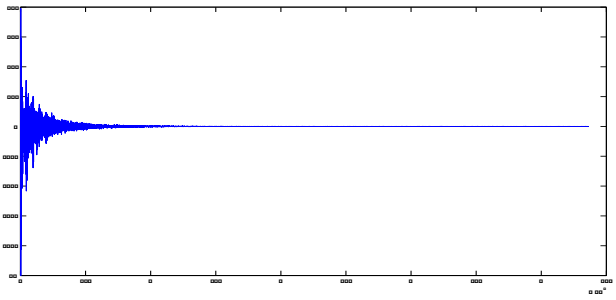


Abbildung 3: Antwort nach Anregung mit exponentiellem Sweep

MLS Anregung:

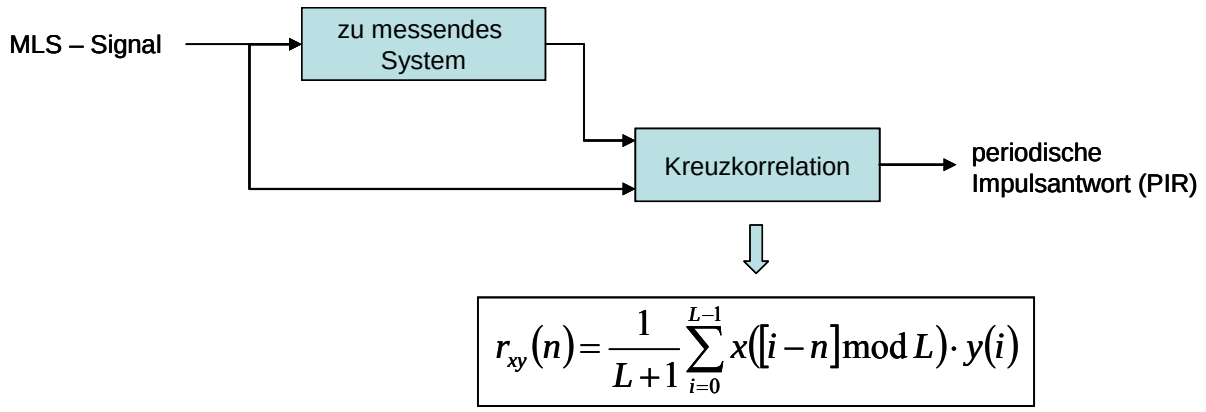
Die Idee, die hinter diesem Anregungssignal steht ist, dass nicht das Anregungssignal selbst, sondern die Autokorrelationsfunktion AKF des Anregungssignals einem idealen Dirac-Impuls entsprechen soll [4]:

Formel 5:

$$r_{xx}(n) = \delta(n) - \frac{1}{L+1}.$$

Dieser Sachverhalt lässt sich durch eine *binäre Pseudozufallsfolge* realisieren, die selbst nur Amplitudenwerte von 0 oder 1 beinhaltet und wurde bei uns durch einen öffentlich zur Verfügung gestellten MATLAB Patch von Christopher Brown [5] implementiert („mls.m“). Die Länge dieser Zufallsfolge entspricht dabei stets $2^N - 1$ Samples plus der zusätzlichen digitalen Stille (lag), die für die Messung der Impulsantwort dient. N beschreibt die Ordnung des MLS Signals.

Durch Anregung des Systems mit der MLS Pseudozufallsfolge lässt sich nun durch Kreuzkorrelation der Systemantwort mit dem MLS Signal selbst die Impulsantwort des Systems berechnen [4].



Das System wird dabei öfters mit dem MLS Signal angeregt und somit am Ausgang eine periodische Impulsantwort „PIR“ erzeugt [4].

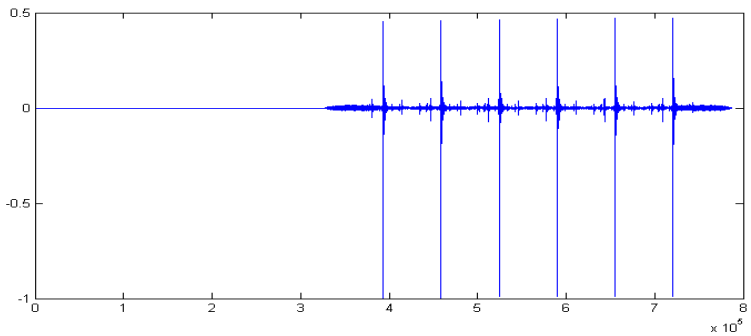


Abbildung 4: Periodische Impulsantwort

Durch geeignetes Abschneiden der periodischen Impulsantwort erhält man in weiterer Folge die Impulsantwort seines Systems.

Die weitere Implementierung des MLS Signals stellt sich als sehr einfach dar und wurde in der MATLAB Funktion „impulse_mls.m“ realisiert.

```
function in = impulse_mls(fs);

m = mls(16);

signal=[m;m;m;m;m;m];
wavplay(signal,fs,'async');
y = wavrecord(length(signal),fs,1);

hr=xcorr(y, m);

hr=hr(6*length(m):7*length(m)-1);

hr_norm = hr./max(abs(hr));

in=hr_norm;
```

In unserer Implementierung wurde ein MLS Signal mit Ordnung 16 erzeugt und der Raum mit 5 Wiederholungen dieses Signals beschickt um gleichzeitig die Systemantwort aufzunehmen.

Die Kreuzkorrelation der Systemantwort mit dem Anregungssignal ergibt die bereits erwähnte periodische Impulsantwort, die nach geeignetem Abschneiden und anschließender Normierung unserer gewünschten Impulsantwort entspricht.

Abgeschaltetes Rauschen:

Der bereits erwähnte Sachverhalt, dass das Verfahren mit abgeschaltetem Rauschen nicht zur Ermittlung der Raumimpulsantwort dienen kann [1], lies uns nicht davon abhalten einen Implementierungsversuch zu starten um aufzuzeigen, wo die Probleme dieser Vorgangsart liegen.

Zunächst ist grundlegend festzuhalten, dass eine Impulsantwort im Zeitbereich nur durch eine Impulsanregung zu realisieren ist. Da weißes Rauschen oder rosa Rauschen alles Anderem, nur keinem Impuls entspricht, lässt sich leicht verstehen, dass diese Signale nicht zur Ermittlung von Impulsantworten geeignet sind.

Jedoch kann dieses Verfahren für unseren Zweck ohne Verwendung von Impulsantworten zur Ermittlung der Nachhallzeiten eines Raumes verwendet werden und somit wurde diese Art der Nachhallzeitmessung erfolgreich durch die Hilfe unseres Seminarbetreuers DI Franz Zotter vom IEM in Pure Data implementiert (kurze Beschreibung folgt).

Warum nutzt man rosa Rauschen?
Die spektrale Rauschleistungsdichte von rosa Rauschen nimmt pro Oktave um 3dB ab. Daraus resultiert ein konstanter Schalldruckpegelverlauf über die logarithmische Frequenzachse und eine konstante Energieverteilung bei gleicher relativer Bandbreite (ähnlich unserem Gehör).

Rauschsignale sind im Allgemeinen nicht stationäre Signale [2]. Daher ist eine Mittelung über mehrere Messungen bei professioneller Messung von Nachhallzeiten unabdingbar.

Großer Nachteil der Anregung mit Rauschsignalen ist allerdings, dass nur begrenzt Energie in den Raum hineingebracht werden kann, da breitbandig angeregt wird und damit auch die Gesamtenergie auf das gesamte Anregungsband aufteilt ist.

Der Versuch der Ermittlung einer Raumimpulsantwort führt nach langem optimieren für einen geeigneten Schnittpunkt für den Zeitpunkt des Beginns des Abklingvorgangs in MATLAB zu folgendem grafischen Ergebnis:

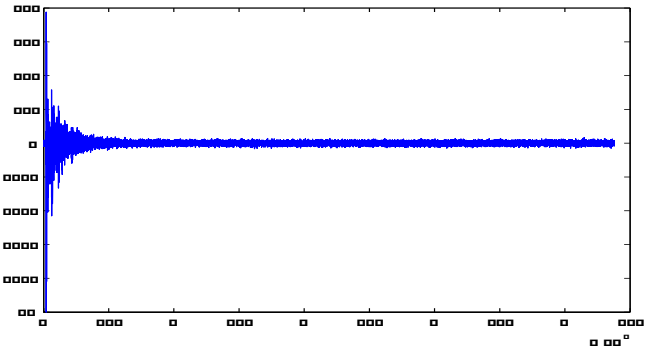


Abbildung 5: „Falsche“ Antwort nach Anregung mit abgeschaltetem Rauschen

Außer dem hohen Rauschanteil lässt dieser Plot ein recht zufriedenstellendes Ergebnis vermuten, jedoch lässt sich mit diesem Signal nichts anfangen. Diese Antwort beschreibt einen bestimmten definierten Zustand des System, der nur bestimmte dominante Frequenzen zum Momentanzeitpunkt darstellt, jedoch beschreibt dieser Plot keine Antwort des Systems auf einen Impuls.

Im Folgenden ist die MATLAB Funktion „impulse_switched_noise.m“ dieses Implementierungsversuchs ohne genauere Erklärung dargestellt.

```
%Messung der Impulsantwort mit abgeschaltetem Rauschen (2sek Einrauschen)
function in = impulse_switched_noise(fs);

% fs = Samplingfrequenz

H = zeros(2048,1);

for n = 1:1024
    H(n) = 1/n;
    H(2049-n) = H(n);
end

h_pink = ifft(H);

%Generierung von rosa Rauschen
noise = randn(2*fs,1);
noise_norm = noise./max(abs(noise));

pink = conv (noise_norm,real(h_pink(1:1024)));
pink = pink(1:(end-1023));
pink_norm = pink./max(abs(pink));
wavplay(pink_norm, fs,'async');
```

```
y = wavrecord (3*fs, fs, 1);
in = y(70000:end);

%Impulsantwort schneiden
in_2 = in.^2;

int=round(30e-3*fs); % Mittelungsintervall
int_count= round(length(in_2)/int);

mean_in_2 = zeros(int_count,1);
mean_in_2(1) = mean(in_2(1:int));

if int*int_count > length(in)
    int_count = int_count-1;
end

for i = 2:int_count
    mean_in_2(i) = mean(in_2((i-1)*int:i*int));
end

% wird zur Normalisierung von Mittelwert und Noise gebraucht
[maxmean, index_max]=max(mean_in_2);

% Mittelwert wird Normalisiert und in dB gewandelt
mean_in_2=mean_in_2./maxmean;
mean_in_2_dB=10*log10(sqrt(mean_in_2));

%Wieviel sind die letzten 10% des Signals?
noise_begin = (length(in)-round(length(in)*0.1))/int;

%Schneide die letzten Prozent aus mean_in_2_dB um Rauschen abzuschätzen
noise = mean_in_2_dB(noise_begin:end);
mean_noise = mean(noise);

%Suche Startwert und Endwert für Decayline
index_start = find(mean_in_2_dB >= -3);
index_end = find(mean_in_2_dB >= mean_noise+5); %Ende 5dB über Rauschen

x = index_start(end):1:index_end(end) ;
poly = polyfit(x,mean_in_2_dB(index_start(end):index_end(end))',1);

x=1:1:length(mean_in_2_dB);
y = poly(2)+x.*poly(1);

Cut = find(y <= 0);

in = in(Cut(1)*int:end);
in = in./max(abs(in));
```

Es sei erwähnt, dass exakte Nachhallzeiten nur durch Messung des Gesamtpegels unter Verwendung des Verfahrens des abgeschalteten Rauschens mit Messung von Absolutpegeln zu erreichen sind.

In unserer Seminararbeit wurde daher über Pure Data ein Rauschsignal in den Raum gesendet. Nach Abschalten der Rauschquelle wird die Zeit für einen Schalldruckpegelabfall von 20dB in Oktavbändern gemessen. Die resultierende Zeit wird mit drei multipliziert und man erhält so die Nachhallzeit für die verschiedenen Oktavbänder (gemäß ISO 3382) [1].

Vergleich der Anregungssignale:

Im Folgenden sind zur Übersicht die genutzten Anregungssignale gegenübergestellt .

- Exponentieller Sweep:
  - hohe SNR möglich
  - empfindlich gegenüber transienten Störungen bzw. Zeitvarianz
  - unempfindlich gegenüber Verzerrungen (Nichtlinearitäten)

- MLS:
  - unangenehmer Klang ;-)
  - sehr hohe SNR möglich
  - Empfindlichkeit: Verzerrungen

- Abgeschaltetes Rauschen:
  - nicht stationär (zufällige Phasenlage beim Abschalten)
  - zwar alle Frequenzen enthalten, aber nur in einem best. Zeitintervall
  - Energie teilt sich auf gesamte Bandbreite (daher geringe SNR)
  - *Kein Anregungssignal zur Impulsantwortmessung*

Weitere geeignete Anregungssignale:

Der Vollständigkeit halber sei erwähnt, dass weitere Möglichkeiten zur Messung von Raumimpulsantworten existieren.

„Time Delay Spectrometry“ TDS beschreibt einen zeitlich linearen Sinussweep [2], ähnelt also in gewisser Weise den bereits implementierten Anregungssignalen und war somit kein weiterer Bestandteil unserer Implementierung.

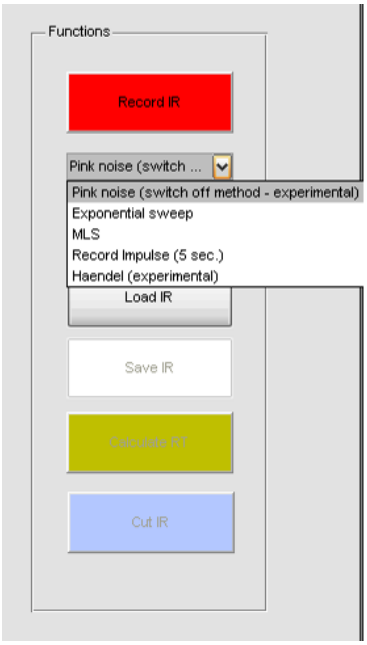


Abbildung 6: Teil der GUI in MATLAB

Ein Schuss einer Schreckschusspistole eignet sich äußerst gut zur Ermittlung von Raumimpulsantworten [2]. Die aufgenommene Systemantwort kann folgend sofort als Impulsantwort angenommen werden.

Genauso gut funktioniert dies auch bei kleineren Räumen mit einem platzenden Luftballon. In unserem MATLAB Programm wurde auch die Aufnahme beliebiger Signale implementiert, sodass wir auch beliebige Dirac-ähnliche Signale händisch erzeugen können.

Grundlegend gilt für die Messung von Raumimpulsantworten →
je mehr Energie in den Raum breitbandig (Dirac) eingebracht werden kann, desto besser wird das Ergebnis einer Raumimpulsantwort.

Das abgebildete Auswahlfenster ist Teil unserer grafischen Benutzeroberfläche „IR_gui.m“ im Ordner „Matlab“ der beigelegten Ordnerstruktur, von der aus sämtliche Funktionen unserer Implementierung bequem bedient werden können.

Gewinn einiger raumakustischer Größen aus Impulsantworten

Da wir nun wissen wie Impulsantworten gemessen werden und wo die vor und Nachteile der verschiedenen Methoden liegen, wollen wir uns in diesem Kapitel mit der Gewinnung einiger der am Anfang erwähnten raumakustischen Größen, aus der Impulsantwort beschäftigen.
Neben den theoretischen Grundlagen wird auch immer eine Implementierung in MATLAB gezeigt.

Nachhallzeit RT60 – Hallradius

Der wohl wichtigste Parameter in der Raumakustik ist die Nachhallzeit.
Die Nachhallzeit, im Englischen „Reverberation Time“, gibt an wie lange es dauert bis der Schalldruck im Raum nach einem Anregungs-Impuls um 60 dB abgeklungen ist – daher auch die englische Abkürzung RT60, welche in Folge verwendet wird.

ISO NORM 3382: „Zeit in Sekunden, die der Schalldruckpegel benötigt um sich um 60dB zu vermindern.“ [1]

Die RT60 kann sowohl messtechnisch aber auch rechnerisch bestimmt werden – wobei messtechnische Methoden gerade bei komplexen Raumstrukturen viel genauer und auch frequenzselektiv arbeiten können.

In der Literatur findet man die Abkürzung RT60 eher im messtechnischen Zusammenhang, bei rechnerischen Verfahren eher die Bezeichnung T_N.

Für die Berechnung gibt es die Formel für die Nachhallzeit nach Eyering [2]:

Formel 6:

$$T_N = 0.161 \frac{V}{-A \cdot \ln(1-\alpha)}$$

V… Raumvolumen
A… Absorptionsfläche
α… Absorptionskoeffizient

Handelt es sich dabei um niedrige Absorptionskoeffizienten, also z.B. Holz oder Steinwände, kann der Logarithmus ln(1 – α) durch Taylorreihenentwicklung mit – α angenähert werden und es ergibt sich die Nachhallzeit nach Sabine:

Formel 7:

$$T_N = 0.161 \frac{V}{\alpha A} \quad [2]$$

Die Genauigkeit dieser zwei Formeln hängt natürlich stark von den Parametern ab, weswegen bei komplexeren Strukturen und unterschiedlichen Wand/Deckenmaterialien ein Messen der Impulsantwort unerlässlich ist – für eine schnelle Abschätzung ist Sabine jedoch eine gute Wahl.

Einer der für die Aufnahmetechnik wichtigsten Parameter lässt sich direkt aus der Nachhallzeit bzw. dem Raumvolumen berechnen – der Hallradius.
Die Definition des Hallradius besagt: Am Hallradius herrscht ein Ausgleich zwischen Direktschall und Diffusschall – in den meisten Fällen die ideale Position für ein Hauptmikrofon:

Formel 8:

$$r_H = 0.057\sqrt{\frac{V}{T_N}}$$ [2]

T20 - T30

Das große Problem bei der Messung der RT60 ist der erforderliche Signal-Rauschabstand (SNR). Da ja ein 60dB Abfall festgestellt werden soll, muss die SNR mindestens diese 60 dB betragen [2].
Das heißt bei einem Ruhepegel von 60dB(A) müsste der Anregungsimpuls eine Lautstärke von mindestens von 120dB(A) haben – was mit einem Lautsprecher im Nahfeld noch möglich ist, wegen des 6dB Abfalls pro Entfernungsverdopplung jedoch als Diffusschall im Raum kaum verzerrungsfrei zu erreichen ist.

Abgesehen von einigen bereits erwähnten Verfahren die keinen Lautsprecher zu Raum-Anregung benötigen (z.B. Schreckschusspistole), sind 60dB SNR unter realen Bedingungen nicht zu erreichen.
Folglich wurden zwei normierte Verfahren entworfen – die T20 und die T30.
Diese benötigen wesentlich weniger SNR, führen aber dennoch zu gut an die Realität angenäherte Ergebnisse.
T20 und T30 basieren darauf, dass der Pegelabfall einer Raumimpulsantwort in der Regel exponentiell verläuft – d.h. wenn man ein Teilstück des Abklingvorganges kennt, kann man diesen extrapolieren bis die -60dB erreicht werden.
Betrachtet man die Impulsantwort in dB ist der Verlauf sogar linear – d.h. eine einfache Multiplikation der Teilnachhallzeiten ergibt die RT60:

T20: Zeit für Abfall von -5dB bis -25dB => Multiplikation mit 3
T30: Zeit für Abfall von -5dB bis -35dB => Multiplikation mit 2

Die Wahl des Verfahrens hängt nun von der verfügbaren SNR ab (25dB für T20, 35dB für T30) – genauere Ergebnisse bringt die T30.

EDC – EDR

Nimmt man die rohe Impulsantwort und versucht daraus Abfall-Zeiten zu lesen, wird man aufgrund der Welligkeit schnell daran scheitern – es müssen also Verfahren zur Glättung der Impulsantwort gefunden werden.

Die einfachste brauchbare Darstellung der Impulsantwort ist die „Energy Decay Curve“ EDC. Sie verwendet die quadratische Impulsantwort und bringt diese in die Dezibel-Skala [6],[7]:

Formel 9:

$$EDC = 10\log_{10}\left(h^2[n]\right)$$

```
figure ('Name', 'EDC')

edc = 10*log10(imp.^2);

plot (fax, edc);

xlabel('Time in s');
ylabel('Energy in dB');
```

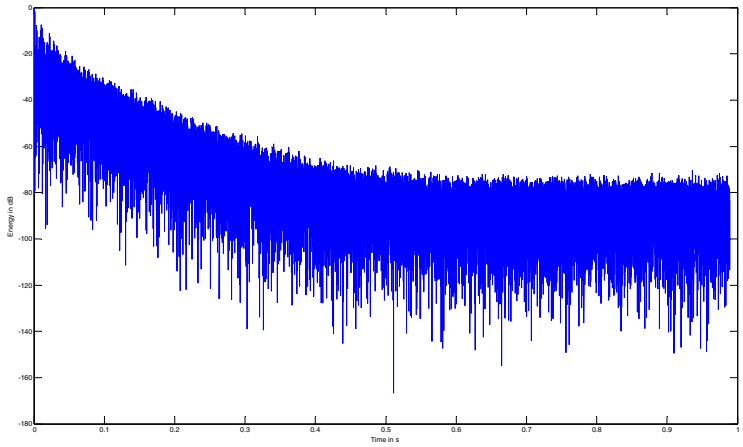


Abbildung 7: Energy Decay Curve EDC

Bei der EDC kann man schon den linearen Abfall auf der Dezibel-Skala erkennen, auch der Übergang in den Noisefloor ist ersichtlich. Die EDC gibt jedoch keine Aussage über das Verhalten im Frequenzbereich.

Hierfür wählt man in Folge das „Energy Decy Relief“ EDR. Diese Darstellung, auch bekannt als Wasserfall Diagramm, erstellt mehrere Kurzzeitspektren und stellt deren zeitlichen Verlauf dann dreidimensional dar.

In MATLAB kann diese mit Hilfe eines Spektrogramms implementiert werden. Hierbei werden die Stücke für die FFT-Analysen zusätzlich gefenstert (mittels eines Hanning-Fensters).

```
figure('Name', 'EDR')
window = (hann(1024).^4);

[S,F,T]=spectrogram(imp,window,round(0.75*length(window)),512,handles.fs);

EDR_spect = 20*log10(S);
mesh(T,F,EDR_spect);

xlabel('Time in s')
ylabel('Frequency')
zlabel('Energy in dB')
```

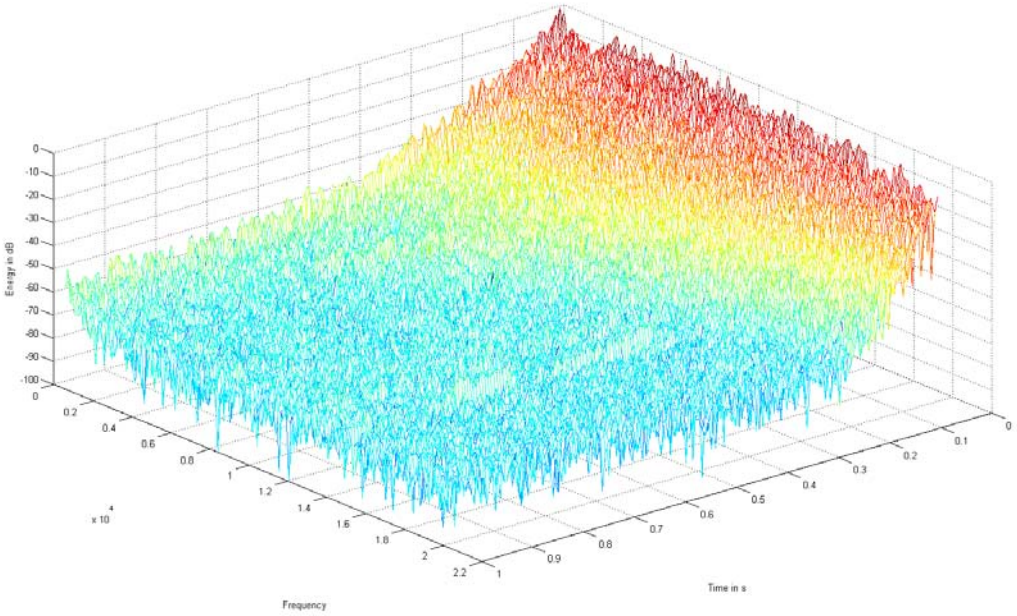


Abbildung 8: Energy Decay Relief EDR

Wie man sieht sind beide Darstellungen aufgrund ihrer Nullstellen jedoch noch wenig Brauchbar um Abfall-Zeiten zu bestimmen.

Schröder-Rückwärtsintegration

Als Methode für die Glättung der EDC wird die Schröder-Rückwärtsintegration verwendet, welche auf der EDC basiert und diese aufintegriert [1],[2],[3],[6],[7]:

Formel 10:

$$L[n] = 10\log_{10}\left(\frac{\int_n^N h^2[\eta]d\eta}{\int_0^N h^2[\eta]d\eta}\right)$$

Die Division im Nenner stellt lediglich eine Normierung auf 0dB dar.

In MATLAB gibt es mehrere Möglichkeiten die Integration durchzuführen. Die akkurateste ist mittels Trapezregel „cumtrapz“ – hierbei wird ein Vektor erstellt dessen Werte immer das Integral bis zum jeweiligen Sample beinhalten. Somit kann man sich eine aufwändige Schleife ersparen:

```
figure('Name', 'EDC-schroeder')

h_2=imp.^2;

int2 = cumtrapz(h_2);
int1 = fliplr(cumtrapz(fliplr(h_2')));

L = 10*log10(int1(1:N)./int2(N));

plot (fax,L);
ylabel('Energy in dB')
xlabel('time in seconds')
```

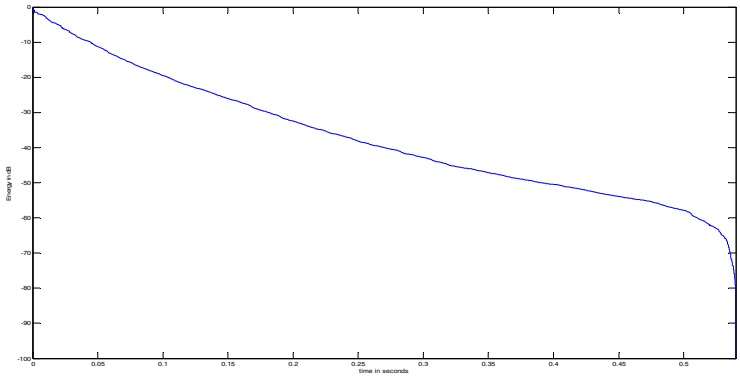


Abbildung 9: EDC mit Schröderintegration

Mit dem gleichen Verfahren kann man nun auch den zeitlichen Verlauf der Spektral-Linien eines Spektrogrammes glätten. Fügt man die Schroeder Integration also als Zwischenschritt in die EDC Funktion ein, gewinnt man ein geglättetes Wasserfall-Diagramm [6]:

```
…
EDR_spect=schroeder_edr(EDR_spect, fs, N);
…
```

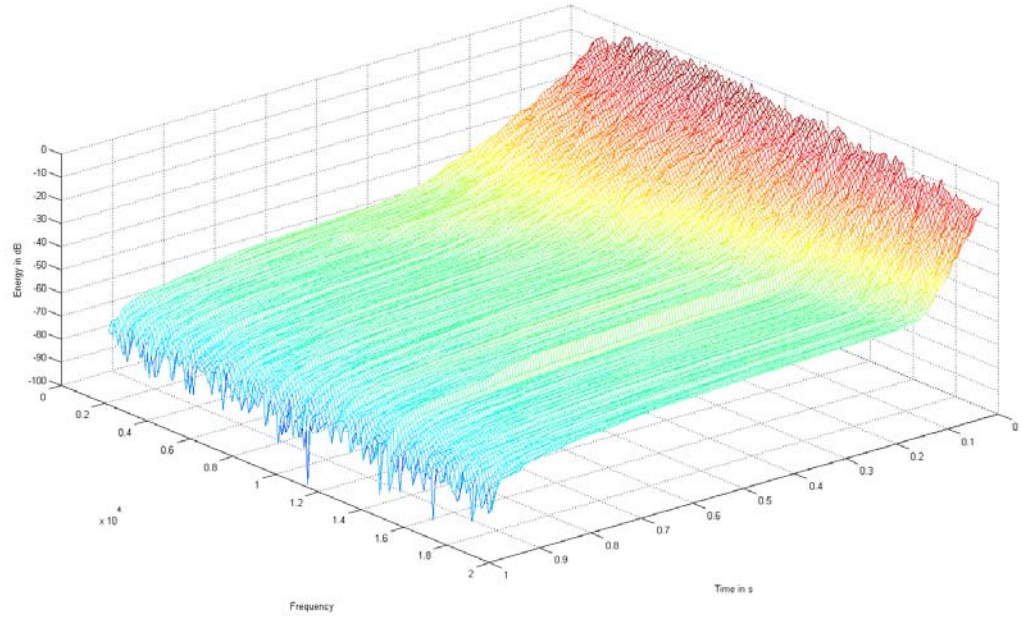


Abbildung 10: EDR mit Schröderintegration

Wie aus den Übergabeparametern der Schroederfunktion und auch aus der Formel ersichtlich gibt es einen wesentlichen Parameter für die Schroederintegration: die Integrationsgrenze N.

Noise-Floor Estimation

Die Integrationsgrenze ist bei der Schröder-Rückwärtsintegration sehr entscheidend.
Wählt man diese zu kurz, hat man unter Umständen eine zu kurze EDC für die Nachhallbestimmung.

Wählt man die Integrationsgrenze zu lang, gelangt man in den Bereich des Noise-Floors und die EDC wird verfälscht, was aus folgender Abbildung ersichtlich wird (rote Linie entspricht Integration mit der „optimalen“ Integrationsgrenze).

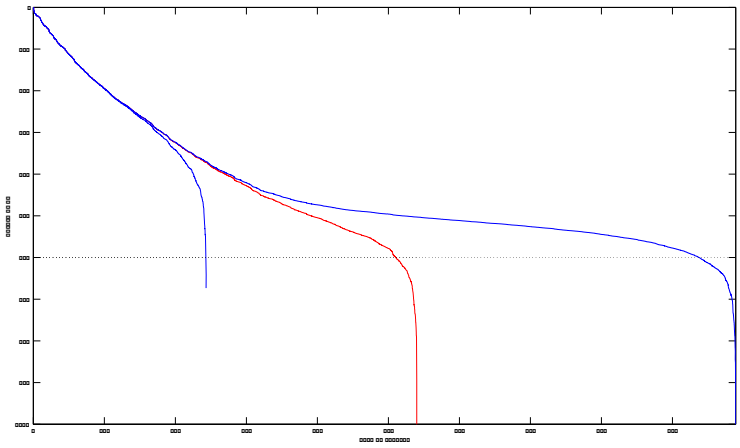


Abbildung 11: Verschiedene Integrationsgrenzen bei der Schröder-Rückwärtsintegration

Das Ziel ist es also diese „optimale“ Grenze zu finden – also den Schnittpunk von Nachhallkurve und dem Grundgeräuschpegel.

Hierzu gibt es mehrere Möglichkeiten, von denen 2 kurz vorgestellt werden sollen:

- *Einfache Schätzung*
  Bei dieser einfachen Methode wird der Pegel des Noise-Floors mittels RMS-Berechnung ermittelt (z.B. aus separater Aufnahme des Noise-Floors).
  Die Integrationsgrenze ist dann der Schnittpunkt an dem die EDC diesen Pegel das erste Mal unterschreitet.

- *Lundeby Noise Estimation*
  Im Gegensatz zur einfachen Schätzung ist das Verfahren nach Lundeby [7] iterativ, somit optimiert sich das Ergebnis: Hier der schematische Ablauf der Optimierung:

  (1) Mittelung der Quadratischen Impulsantwort über kure Zeitintervalle
  (2) Bestimmung des Noise-Levels aus den letzten 10% der aufgenommenen IR
  (3) Lineare Regression zw. 0dB und 5-10dB über Noise Level
  (4) Gewinn des ersten Schnittpunktes
  (5) neue Mittelung so dass 3-10 Mittelungs-Intervalle pro 10dB Abfall
  (6) Erneute Berechnung des Noise-Levels (5-10dB Abfall der Regressionsgeraden nach dem Schnittpunkt)
  (7) Neue Lineare Regression (wieder 5-10dB über NL, jedoch nur für einen Dynamikbereich von 10-20dB)
  (8) Neuer Schnittpunkt

Die Schritte 6-8 werden nun wiederholt biss keine Veränderung mehr entsteht (maximal 5 Iterationen [7]). In der folgenden Abbildung sind die Iterationen grafisch dargestellt:

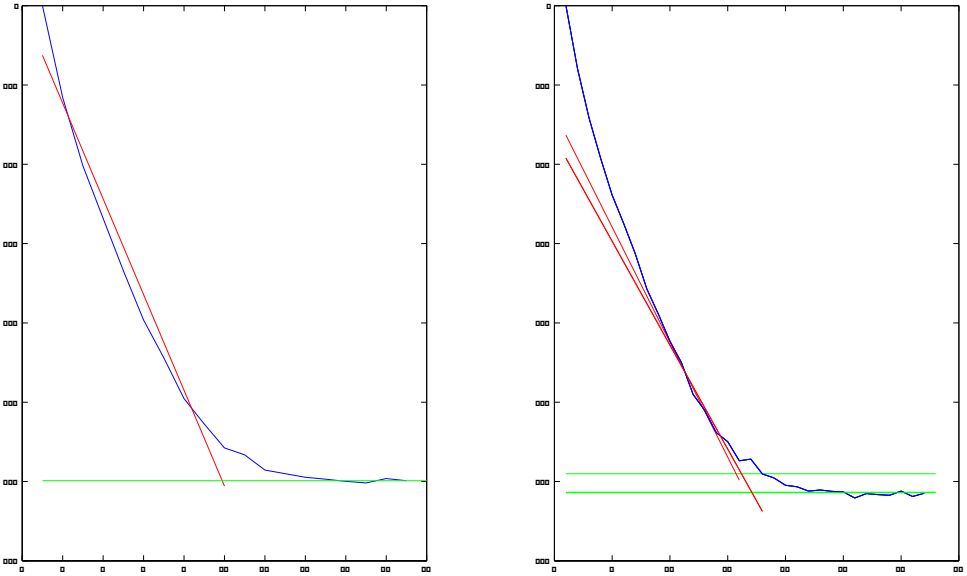


Abbildung 12: Lundeby Noise Estimation

Andere Methoden um dem Problem des Noise-Floors bei der Schröder Rückwärtsintegration Herr zu werden sind:

- *Methode nach Chu*
  Chu schlägt vor den quadratischen Mittelwert des Noisefloors von der quadratischen Impulsantwort abzuziehen – dies funktioniert in der Praxis jedoch nicht, da das Noise nicht stationär ist [7].

- *Hirata’s Method*
  Anstatt einer Quadrierung der Impulsantwort multipliziert Hirata zwei an der gleichen Stelle gemessene Impulsantworten miteinander – da die Impulsantworten theoretisch identisch sein sollten, das stochastische Noise jedoch nicht müsste letzteres somit herausgerechnet werden [7].
  Dies funktioniert jedoch nur dann wenn die Impulsantworten über eine lange Zeit stationär sind, was jedoch in großen Konzertsälen nicht gegeben ist.

Wir haben uns für eine Implementierung der „Lundeby Noise Estimation“ entschieden – diese ist in MATLAB relativ leicht programmierbar, wenngleich der Code durch die vielen Schritte sehr umfangreich ist.

Generell muss man aber auch erwähnen dass bei einer T20 Berechnung die Integrationsgrenze kaum einen Einfluss aus die RT60 hat. Denn wie man in Abbildung 11 sieht, verändert sich die EDC erst nach einem Pegelabfall von 30 dB durch verschiedene Integrationsgrenzen.

Gewinn der T20 und T30 aus der EDC

Da nun ein geeignetes Verfahren zur Glättung der EDC gefunden wurde, kann man die T20 und die T30 aus dieser gewinnen.

Hierfür gibt es zwei Methoden:

Bei der bereits weiter oben beschriebenen, einfacheren Methode werden lediglich die Zeitpunkte des -5dB und des -25/-35dB Abfalls gesucht, die Differenz berechnet und anschließend multipliziert um die RT60 zu erhalten [2].
Diese Berechnung ist sehr einfach zu realisieren und kommt noch aus einer zeit in der Pegelabfälle mit Pegelschreibern Aufgezeichnet wurden und durch Geradenlegung die Nachhallzeit berechnet wurde.

Um jedoch zu einem genaueren Ergebnis zu kommen kann man sich den Vorteil einer „digitalen EDC“ zu Nutze machen um somit nicht einfach eine Gerade durch Anfangs- und Endpunkt zu legen, sondern die Kurve zwischen diesen zwei Punkten mittels einer Geraden anzunähren [7].

In MATLAB wird dies mittels der Funktion „polyfit“ realisiert.

Dieser Vorgehensweise ist sinnvoll, da die EDC nur Nährungsweise linear ist. Abhängig vom vermessenen Raum kann es Abweichungen geben, welche bei der zweiten komplexeren Methode berücksichtigt werden.

Die folgenden Abbildungen stellen die Unterschiede der Realisierung grafisch dar.

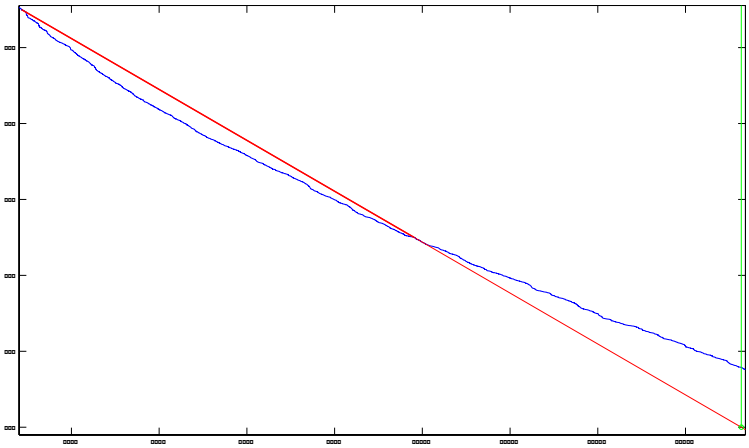


Abbildung 13: Geradenlegung durch den Anfangs- und Endpunkt

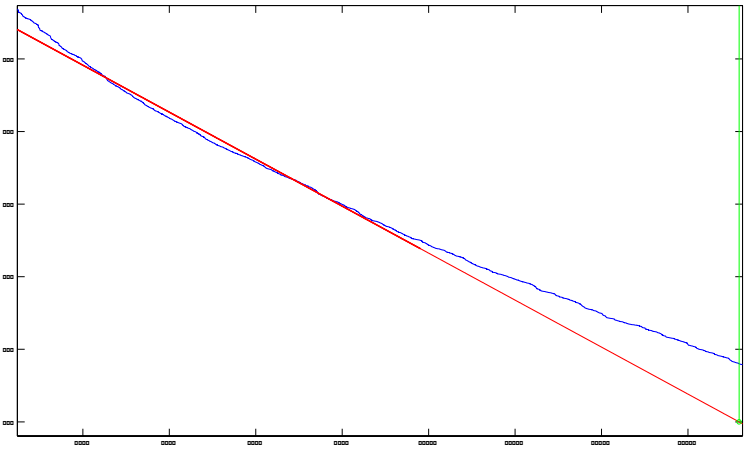


Abbildung 14: Annährung der Kurve zwischen Anfangs und Endpunkt

Hat man die Kurve nun bestmöglich angenährt wird die Gerade verlängert. Der Zeitpunkt zu dem die Gerade die -60dB Marke überschreitet ist die resultierende Nachhallzeit RT60.

Die Unterschiede zur „normalen“ Berechnung mittels Geradenlegung sind im Normalfall marginal, unter bestimmten Umständen jedoch ausschlaggebend (z.B. im Fall sehr ausgeprägter isolierter Reflexionen).

Frequenzabhängige Nachhallzeit und die Schröderfrequenz:

In den bisherigen Betrachtungen wurde die Frequenzabhängigkeit der Nachhallzeit immer unberücksichtigt gelassen [6].

Für allgemeine Berechnungen, wie z.B. dem Hallradius, ist dies auch zulässig – reicht jedoch bei weitem nicht aus wenn es darum geht einen Raum ganzheitlich zu beschreiben.

Die Frequenzabhängigkeit der RT60 hat verschiedene Ursachen.

Zum einen die Dissipation des Schalls bei der Ausbreitung in Luft, was eine relativ stärkere Dämpfung der hohen Frequenzen zur Folge hat.

Zum Anderen spielt die Wellenlänge des Schalls bei der Raumakustik eine wichtige Rolle, da diese zu Phänomenen wie Absorption und Beugung führt.

Zuletzt wären noch geometriebezogene Eigenschaften wie Raummoden zu nennen. Diese führen speziell bei tiefen Frequenzen zu Resonanzerscheinungen uns somit zu sehr langen Ausklingzeiten [1].

Allgemein kann gesagt werden, dass die Nachhallzeit für gewöhnlich zu hohen Frequenzen abnimmt. Dies liegt daran dass der in ***Formel 6*** und ***Formel 7*** angeführte Absorptionskoeffizient stark frequenzabhängig und für hohe Frequenzen bei handelsüblichen Materialien wie Holz oder Stoff wesentlich höher als für tiefe Frequenzen ist.

Um die Frequenzabhängigkeit der Nachhallzeiten zu berücksichtigen wird die Impulsantwort in mehrere Frequenzbänder aufgeteilt – laut ISO NORM 3382 mindestens in Oktavbänder [1]. Besser und exakter ist jedoch eine Aufteilung in 22 Terzbänder von 63 Hz bis 8 kHz [1].

Folgend eine Auflistung der Mitten- und Eckfrequenzen der in MATLAB implementierten Filterbank.

Nr	**f_m**	**f_u**	**f_o**
1	63 Hz	56 Hz	71 Hz
2	80 Hz	71 Hz	90 Hz
3	100 Hz	90 Hz	112 Hz
4	125 Hz	112 Hz	140 Hz
5	160 Hz	140 Hz	180 Hz
6	200 Hz	180 Hz	224 Hz
7	250 Hz	224 Hz	280 Hz
8	315 Hz	280 Hz	355 Hz
9	400 Hz	355 Hz	450 Hz
10	500 Hz	450 Hz	560 Hz
11	630 Hz	560 Hz	710 Hz
12	800 Hz	710 Hz	890 Hz
13	1000 Hz	890 Hz	1120 Hz
14	1250 Hz	1120 Hz	1410 Hz
15	1600 Hz	1410 Hz	1800 Hz
16	2000 Hz	1800 Hz	2240 Hz
17	2500 Hz	2240 Hz	2800 Hz
18	3150 Hz	2800 Hz	3550 Hz
19	4000 Hz	3550 Hz	45000 Hz
20	5000 Hz	45000 Hz	5600 Hz
21	6300 Hz	5600 Hz	7100 Hz
22	8000 Hz	7100 Hz	8000 Hz

Tabelle 1: fm=Mittenfrequenz, fu=untere, fo= obere Grenzfrequenz

Wichtig ist abzusichern dass vor allem die sehr schmalbandigen Filter im Bassbereich nicht aufgrund zu hoher Güte in Resonanz kommen, also überschwingen. Dies würde die Nachhallzeiten in diesen Bändern verlängern und die Ergebnisse wären verfälscht.

In MATLAB lässt sich dies sehr einfach graphisch mittels der Funktionen „freqz“ oder „fvtool“ überprüfen:

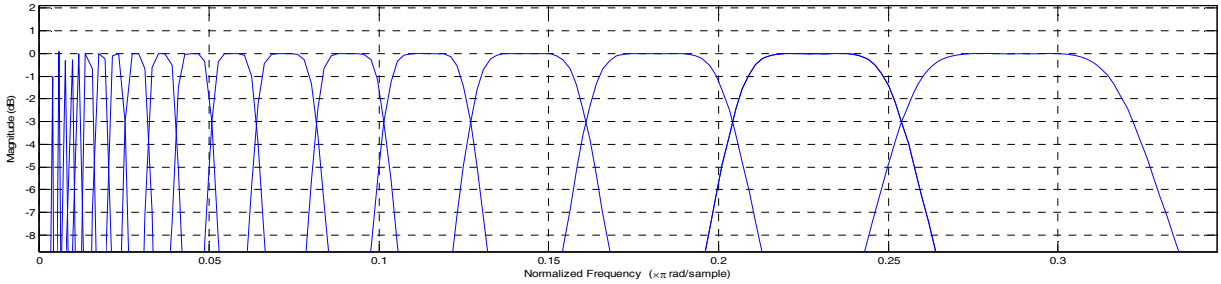


Abbildung 15: Frequenzgang der implementierten Filterbank laut ISO-Norm

Hat man nun das Signal aufgeteilt kann man die T20 bzw. T30 Berechnungen für die Teilbänder durchführen,
Man stellt nun speziell im Bassbereich starke Unterschiede fest, was an den genannten Raummoden liegen kann, jedoch auch an der so genannten Schröderfrequenz:

Formel 11:

$$f_{Schröder} \approx 2000\sqrt{\frac{T_N}{V_{Raum}}}$$

Diese gibt einen Vetrauensbereich für dieses statistische Verfahren an, ab welchem durch Schröder Rückwärtsintegration verlässliche Berechnungen zu erwarten sind (auf dieser beruht auch die Vorgestellte Berechnung der RT60).

Zuletzt noch eine grafische Darstellung der Nachhallzeiten in den 22 Teilbändern wie sie in MATLAB implementiert wurde.

Es sind sowohl Unterschiede bei T20 oder T30-Berechnung zu erkennen, als auch falsche Zeiten die unterhalb der Schröderfrequenz entstehen (diese lag in der Messung bei einem Raumvolumen von 65 m^2 und einer mittleren RT60 von ca. 0.4 s bei 156 Hz).

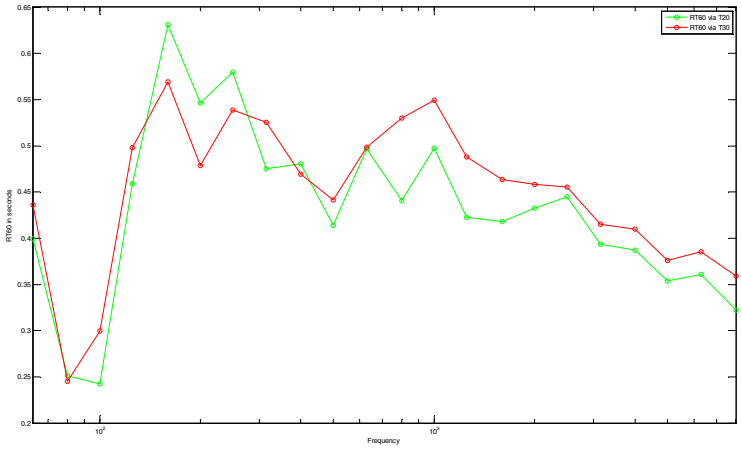


Abbildung 16: Nachhallzeiten in verschiedenen Terzbändern

Nachhallplateaus:

In unserer MATLAB-Implementierung wurde auch der Versuch unternommen die drei wichtigen Teilbereiche der Nachhallkurve in eine Grafik darzustellen:

→ Early Decay Time (EDT)
→ Late Decay Time (Nachhallfahne)
→ Noisefloor

Es wird also zuerst die Zeit, die für einen 10dB Abfall benötigt wird berechnet. Anschließend wird die T20 oder T30 berechnet, die Gerade bis zum Noise-Floor Level extrapoliert und alle drei Werte als Geraden in eine Grafik gelegt.

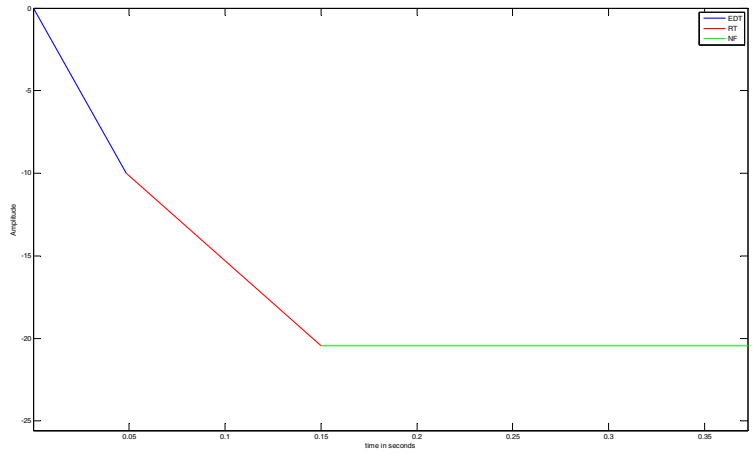


Abbildung 17: Darstellung des Nachhallverlauf s mittels Plateaus

Vorstellung und Bedienung der Software

Starten der GUI in MATLAB:

Startet man die Funktion „IR_gui.m“ in MATLAB, kommt man zuerst ins Hauptfenster.

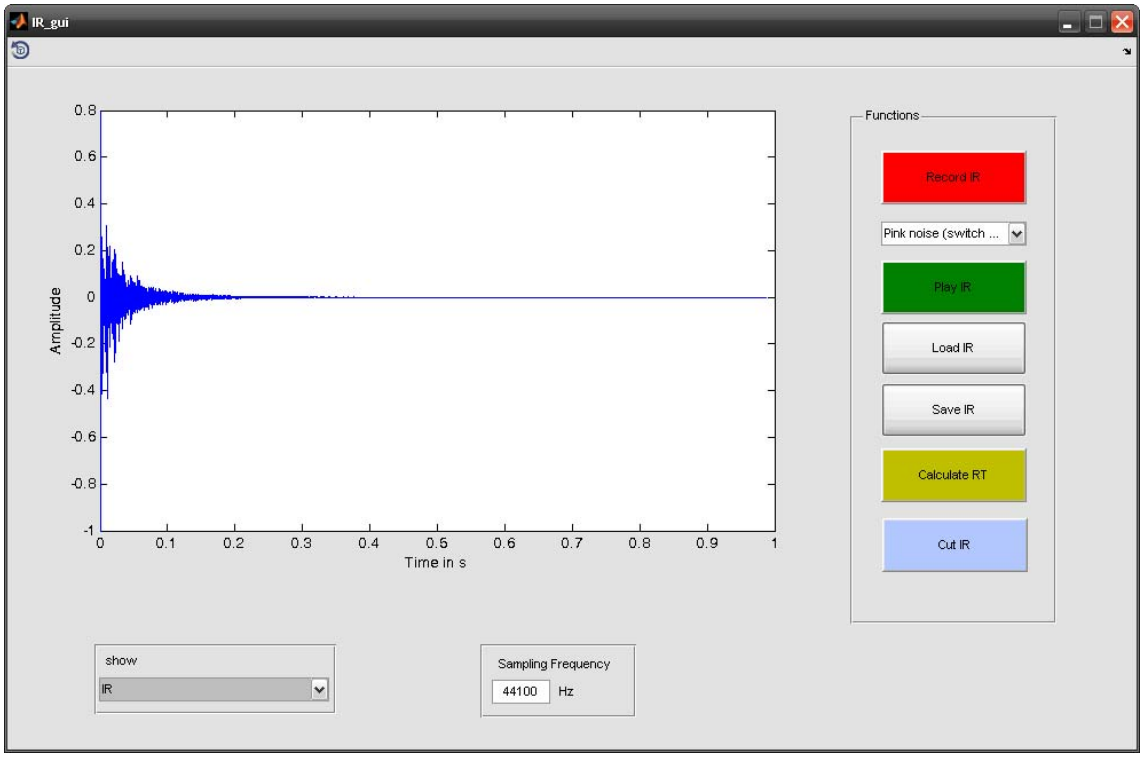


Abbildung 18: Abbildung des Programmfensters in MATLAB

Hier kann man entweder eine vorhandene Impulsantwort laden bzw. speichern, oder man kann eine neue messen. Für die Messungen erwartet MATLAB das Eingangssignal des Mikrofons (mit Kugel-Charakteristik) am linken Audioeingang, das Messsignal selbst wird über den beide Audioausgänge abgespielt – um Auslöschungen durch unterschiedliche Phasenlagen zu vermeiden sollte jedoch nur ein Lautsprecher zur Raumanregung verwendet werden. Oder zwei Rücken an Rücken stehende, um eine Kugel-ähnliche Abstrahlung und somit eine sehr diffuse Anregung zu erzielen.

Mögliche Messmethoden:

Die folgenden Messmethoden können im Drop-Down Menü ausgewählt werden:

- *Exponentieller Sweep:*
  Bei dieser Methode wird ein Sinus-sweep von 20 Hz bis 20kHz über die Lautsprecher in den Raum geschickt und wie schon oben beschrieben die Impulsantwort berechnet.

- *MLS:*
  Hier wird die Impulsantwort mittels eines MLS-Pseudorauschens gemessen, dessen Kreuzkorrelation mit der Systemantwort die gewünschte Impulsantwort ergibt. Diese Berechnung erfolgt wie beschrieben automatisch mit der in MATLAB integrierten Funktion „xcorr.m“.

- *Record Impuls (5 sec.):*
  Hierbei kann eine Dirac-ähnliche Anregung manuell erzeugt werden, z.B. mittels einem platzendem Luftballon. Ab dem Click auf Record hat man dafür 5 Sekunden zeit – es empfiehlt sich einen Augenblick zu warten, da das System unter Umständen nicht sofort mit der Aufnahme beginnt.

Schneiden von Impulsantworten:

Nach der Messung oder dem Laden einer Impulsantwort, empfiehlt es sich mit „Cut IR“ den Signalanteil vor der Impulsantwort wegzuschneiden, da dieser die nachfolgenden Berechnungen verfälschen würde.

Anzeige der EDC und EDR:

Im unteren Drop-Down Menü kann man sich auch die EDC oder die EDR anzeigen lassen, allerdings ohne eine vorherige Glättung durch die Schröder-Rückwärtsintegration.

Starten des Berechnungsprogramms „GUI_schroeder“:

Durch Betätigung des Buttons Calculate RT lässt sich die GUI zu den weiterführenden Berechnung der Seminararbeit starten.

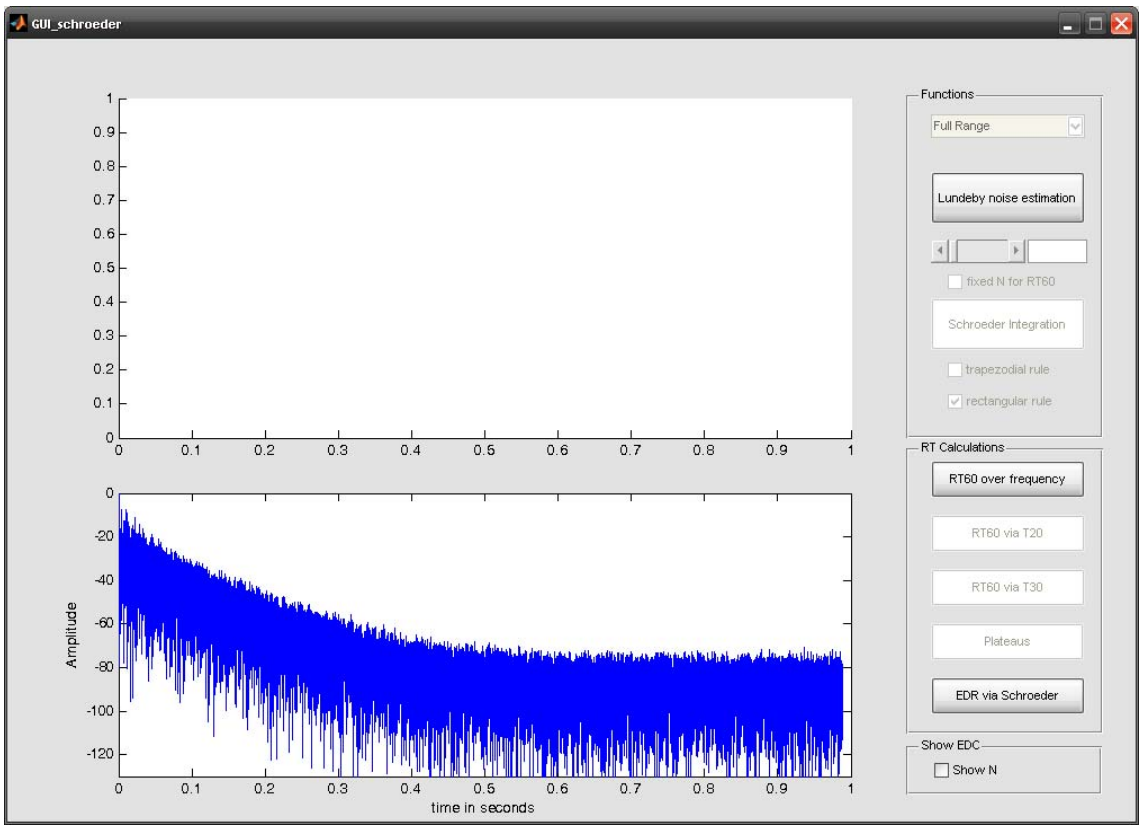


Abbildung 19: Abbildung des Berechnungsfensters in MATLAB

Im unteren Fenster wird standardmäßig die ungeglättete EDC angezeigt.

Manuelle Wahl der Integrationszeit der Schröder-Rückwärtsintegration:

Mit dem Slider für N kann man die Integrationsgrenze (siehe Oben) manuell einstellen.

Automatische Berechnung der Integrationszeit mittels des Lundeby Noise Estimation Algorithmus:

Weiters lässt sich mittels des Buttons „Lundeby Noise Estimation“ die optimale Integrationszeit der Schröder-Rückwärtsintegration berechnen lassen. In Folge werden erst dann weitere Funktionen wie die Schröder-Integration und in weiterer Folge die T30/T20 Berechnung frei geschalten.

Berechnung der „RT60 over Frequency“ und „EDR via Schröder“:

Von vornherein verfügbar sind die selbsterklärenden Buttons „RT60 over Frequency“ und „EDR via Schroeder“, welche in ihren Funktionen eine automatische Noise Estimation zur Schröder-Integration beinhalten – bei der RT60 abhängig von der Frequenz geschieht diese Berechnung sogar separat für jedes Terzband, sofern kein Hacken bei „fixed N for RT60“ gesetzt ist. Dies ist dann sinnvoll wenn der Lundeby Algorithmus für ein bestimmtes Band keinen Crosspoint finden kann und einen Fehler verursacht.

In diesem Fall wählt man mittels des Sliders ein sinnvolles N, also den Schnittpunkt von Nachhallfahne und Noise Floor, und setzt einen Hacken bei besagtem Kästchen.

Grafische Darstellung der Integrationsgrenze:

Mit „Show N“ rechts unten kann man sich das gewählte N (in Samples) in der EDC anzeigen lassen.

Anzeige der durch Schröder-Rückwärtsintegration optimierten EDC:

Hat man nun ein N bestimmt oder berechnen lassen, kann man sich mittels „Schroeder Integration“ die geglättete EDC berechnen und anzeigen lassen. Für die Berechnung kann man noch zwei verschiedene Integrationsregeln aus wählen (Trapez- oder Summationsregel), die aber zu fast gleichen Ergebnissen führen.

Berechnung und grafische Darstellung von T20 und T30:

Nun kann man sich noch die T20 der die T30 berechnen und in die EDC zeichnen lassen.

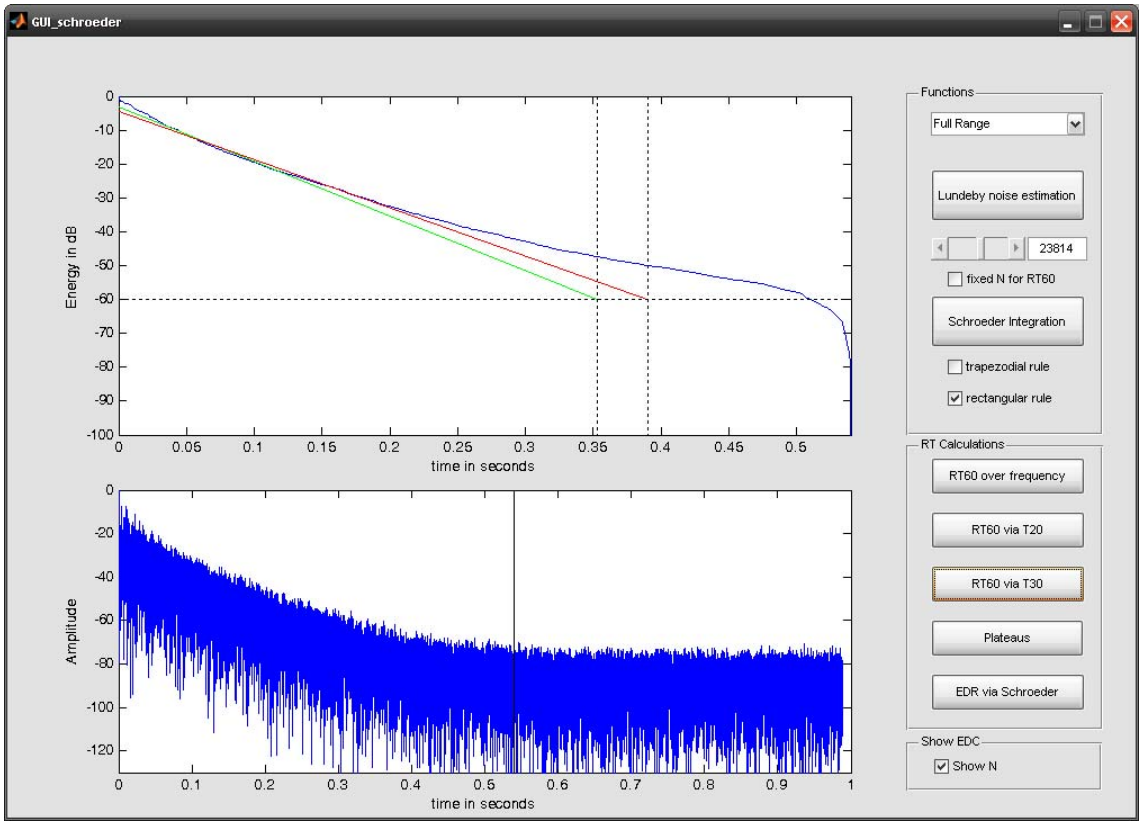


Abbildung 20: Darstellung nach Berechnung der T20 und T30

Darstellung von Nachhallplateus:

Mittels des Buttons „Plateaus“ ist eine Darstellung der bereits beschriebenen Nachhallplateaus möglich.

Auswahl des gewünschten Frequenzbereichs/-bandes:

Man kann die ganze Prozedur nun für jedes beliebige der 22 Terzbänder durchführen, dies geschieht mittels Auswahl des Bandes im Drop-Down Menü und anschließender Wiederholung der nun bekannten Schritte.

Einführung in die Implementierung in PD:

In der Open Source Software PD wurde wie bereits erwähnt die Messung mit abgeschaltetem Rauschen implementiert, allerdings in der „richtigen“ Version, also mit reiner Energiebetrachtung.

Starten des Patchs in Pure Data:

Man startet hierzu den Patch mit dem Namen „noise_thirdoctave_audio2mat.pd“.

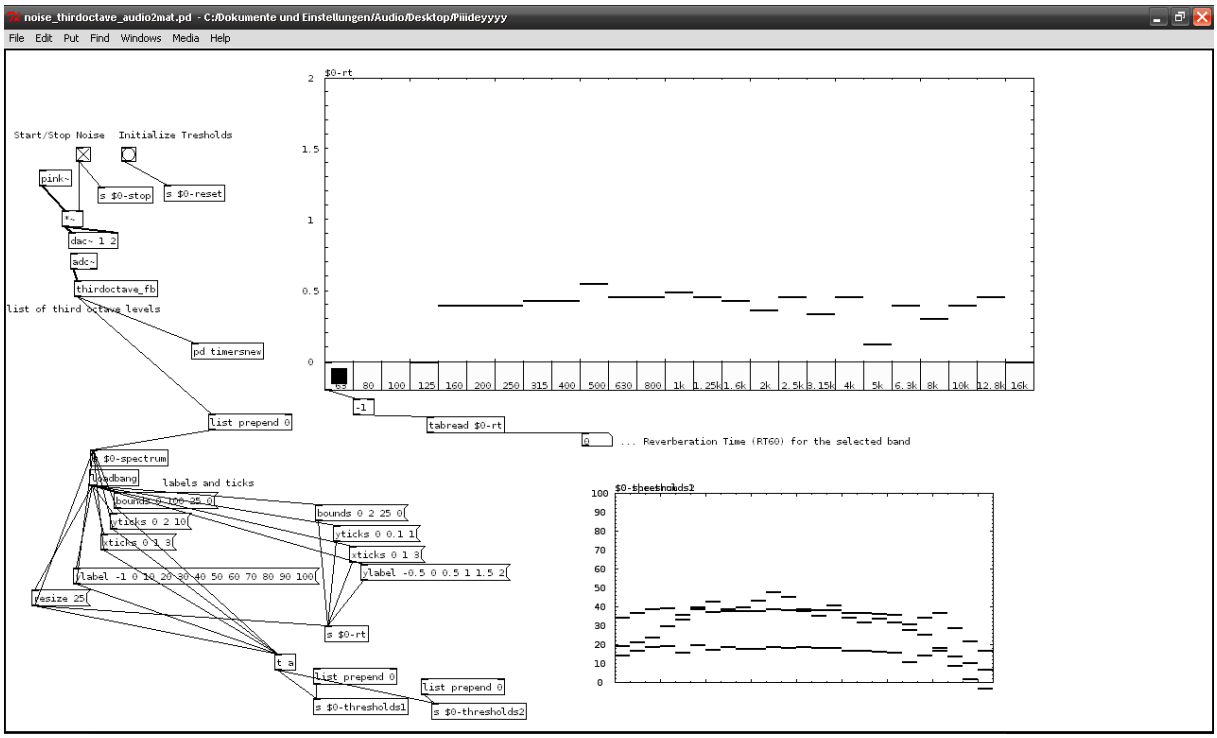


Abbildung 21: Darstellung des grafischen Patchs in Pure Data

Einspeisung des Mikrofonsignals und Ausgabe des Messsignals:

Das Mikrofonsignal wird an den linken Audioeingang angeschlossen, am rechten und linken Ausgang liegt das Messsignal an welches am besten mit zwei Rücken an Rücken aufgestellten Lautsprechern wiedergegeben wird.

Starten der Wiedergabe des rosa Rauschens:

Mit einem Click auf „Start/Stop Noise“ beginnt die Wiedergabe des Rosa Rauschens.

Ermittlung der Pegelschwellen:

Mit einem Klick auf den Button „Initialize Tresholds“ werden die gemittelten Schwellwerte berechnet und graphisch im Terzbandanalysator (rechts unten) dargestellt.

Stop der Rauchwiedergabe und Beginn der Messung:

Ein erneuter Klick auf „Start/Stop Noise“ schaltet das Rauschen ab und das Programm ermittelt – wie beschrieben – die RT60 indem die Zeit, die der Energieabfall zwischen oberem und unterem Schwellwert benötigt gemessen wird.

Grafische Darstellung der Nachhallzeit:

Im der Oberen Tabelle werden die Nachhallzeiten für die Terzbänder grafisch dargestellt und mit Auswahl des jeweiligen Bandes numerisch ausgegeben.

Bildverzeichnis

Quellennachweise

[1] ÖNORM EN ISO 3382: „Akustik – Messung der Nachhalzeit von Räumen mit Hinweis auf andere akustische Parameter“
[2] W. Weselak, G. Graber: Skript zur Vorlesung „Raumakustik“, Institut für Breitbandkommunikation – Technische Universität Graz
[3] M. Vorländer: Skript zur Vorlesung „Akustische Messtechnik“, Institut für Technische Akustik – Fakultät für Elektrotechnik und Informationstechnik Aachen
[4] P. Majdak: Skripten zur Vorlesung „Algorithmen in Akustik und Computermusik“, Institut für elektronische Musik – Kunstuniversität Graz
[5] C. Brown, MATLAB Central “An open exchange for the MATLAB and Simulink user community”: Datei “mls.m”
[6] J.M. Jot, L. Cerveau, O. Warusfel: “Analysis amd synthesis of room reverbation based on a statistical time-frequency model”
[7] M. Karjalainen, P. Antsalo, A. Mäkivirta, T. Peltonen, V. Välimäki: “Estimation of Modal Decay Parameters from Noisy Response Measurements“, J. Audio Eng. Soc., Vol. 50, No. 11, 2002 November