UPPSALA UNIVERSITET

TVE-F 19004
Examensarbete 15 hp
Juni 2019

Development of real time audio equalizer application using MATLAB App Designer

Johannes Langelaar
Adam Strömme Mattsson
Filip Natvig

UPPSALA UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Development of real time audio equalizer application using MATLAB App Designer

Johannes Langelaar, Adam Strömme Mattsson, Filip Natvig

This paper outlines the design of a high-precision graphic audio equalizer with digital filters in parallel, along with its implementation in MATLAB App Designer. The equalizer is comprised of 31 bands separated with a one-third octave frequency ratio, and its frequency response is controlled by 63 filters. Furthermore, the application can process audio signals, in real time, recorded by microphone and from audio files. While processing, it displays an FFT plot of the output sound, also in real time, equipped with a knob by which the refreshing pace can be adjusted. The actual frequency response proved to match the desired one accurately, but the matching is computationally demanding for the computer. An even higher accuracy would entail a computational complexity beyond the power of ordinary computers, and was thus concluded to be inappropriate. As a result, the final application manages to provide most laptops with both high precision and proper functionality.

Handledare: Ping Wu
Ämnesgranskare: Christian Strietzel, Mikael Sternad
Examinator: Martin Sjödin
ISSN: 1401-5757, 19004
Tryckt av: Uppsala Universitet

Sammanfattning

För många kan ordet "equalizer" låta abstrakt och ovidkommande. Den generella beskrivningen av en equalizer brukar lyda: "Ett verktyg som justerar balansen mellan frekvenskomponenter hos en elektronisk signal." Denna mening tycks intetsägande för den mindre insatte. I generella termer må den låta högtravande, men equalizern förekommer icke desto mindre i de mest vardagliga av sammanhang, och inte bara i publikationer av promoverade doktorer i signalbehandling. Equalizerns tillgänglighet kan belysas med ett kort praktiskt exempel som följer.

På en del musikhögtalare av den högre prisklassen hittar man bland annat två vred avsedda för bas och diskant. Dessa vred tillåter justering av ljudvolymen på det låga och höga frekvensområdet var för sig, till skillnad från en vanlig volymratt som verkar på hela frekvensspektrumet. Lyssnaren kan med andra ord själv ställa in volymen på bas och diskant efter behag. Detta verktyg är inget mindre än en enkel så kallad audio-equalizer. Ur ett inlärningsperspektiv är just audio-equalizers tacksamma, då de opererar på signaler som vi är skapta för att uppfatta och förstå, nämligen ljud.

När det kommer till de signaler som framkallar våra sinnesförnimmelser, förefaller signalernas egenskaper plötsligt högst konkreta och reella. Innebörden av både frekvens och intensitet blir glasklar. Även de som inte är bekanta med begreppen kan enkelt skilja på ett dovt buller och ett gällt skrik, så väl som på rött och violett ljus. Lättheten i att förstå sig på signaler i form av ljud är en stor anledning till att detta arbete är gjort i en sådan kontext.

Equalizern i högtalaren i exemplet ovan är en tvåbandsequalizer: Antalet band svarar för antalet frekvensintervall med manuellt justerbar volym. I detta arbete konstrueras en equalizer med 31 band. Storleken och placeringen av dessa band är logaritmiskt fördelade mellan 20 och 20000 Hz, där dessa gränser är satta utifrån motsvarande på frekvensomfånget av vår perception.

Detaljreglering av låtar i strävan efter förstärkt musikupplevelse är inte det enda syftet som audio-equalizern tjänar. Dess förmåga att påverka ljudet i realtid banar väg för ytterligare användningsområden, såsom reglering av ljud som tas in via mikrofon och önskas spelas upp direkt i högtalare. Det kan exempelvis vara i syfte att de närvarande i en hörsal, låt säga under en föreläsning, ska nås av ljud med högsta möjliga skärpa. Ett annat exempel är för att minska eko i kyrkor, vars akustik annars ger upphov till en utdragen efterklang. Sammanfattningsvis kan det konstateras att audio-equalizern påverkar tillräckligt många människors vardag för att förtjäna att kallas för just vardaglig. Den är med andra ord mycket mer än ett ämne för teoretiska samtal och rapporter invigda signalbehandlare emellan.

Contents

1 Introduction

1.1 Background

Audio equalization is the process of regulating the frequencies of an electronic signal, and the tool that enables this process is called an equalizer. This tool is employed in auditoriums to reduce undesirable noise and reverb, as well as when recording and reproducing music. Its employment in the music industry is supposedly the one that most people think of in relationship to equalizers.

In recording and reproducting sound, the most common kind of equalizer is called graphic equalizer which is a set of manually adjustable volume sliders, where each corresponds to a certain frequency. By using such a tool, the sound can be shaped by the listener to a certain extent. By boosting the lower end of the frequency spectrum of a signal, i.e., to increase its amplitude, the bass will get more prominent. This effect will become even more pronounced if the higher end of the same spectrum is cut, meaning that the amplitude of those frequencies is lowered. Likewise, the treble (higher frequencies) can be magnified by reversing this procedure.

In the music industry, much more sophisticated and flexible equalizers are used in pursuit of the highest achievable sound quality. With knowledge of the circumstances under which the equalizer is supposed to operate, the producer can choose a suitable setting. In a club for example, the mixer might want to turn up the bass, whereas on a lecture in an auditorium, the technicians would choose a configuration intensifying some of the higher frequencies.

There are two mutually independent ways of categorizing audio equalizers. One way is to split them into digital and analog equalizers, and the other is into parametric and graphic ones. The digital and analog categorization refers to the type of signal that the equalizer operates on. The division into parametric and graphic, on the other hand, is related to the way and the degree to which the bands can be controlled.

The bands of the parametric equalizer can be adjusted with respect to gain, center frequency and bandwidth whereas the graphic equalizer, by contrast, only allows for the altering of the gains. This makes the former superior to the latter in terms of capacity and flexibility [1].

As to the user-friendly aspect however, the graphic equalizer can be considered superior. For non-professionals in particular, the parametric equalizer is rather cumbersome to use. This is partly due to the fundamental trade-off between straightforwardness and capacity for high detail control: One can only be attained at the cost of the other.

Another equally fundamental trade-off when implementing an equalizer, is between accuracy and computational speed. A tight match between the target response (desired frequency response) and the actual frequency response requires a filter of high order. The higher the order of the filter, the larger the number of computations necessary to calculate it. When altering the target response, i.e., changing the volume at one or more frequency points, the filter will change accordingly. Thus these calculations are executed when, and only when, doing so.

In relation to the users interaction with the equalizer and its impact on sound, there are often a plethora of designing paths to relatively similar results. Two equalizers of differing designs might appear indistinguishable from the perspective of the user. The frequency response, for example, is controlled in the exact same way in a graphic equalizer with a cascaded filter design as in one with filters arranged in parallel. Even with regard to their impact on the output sound, granted a certain frequency response, it is hard to differentiate between the two. As far as an amateur user is concerned, the two equalizers might as well be the same.

In summary, there are numerous ways of constructing an audio equalizer, and the optimal way depends on the purpose it is meant to serve. Each kind of equalizer is characterized by inherent limitations, and each purpose makes unique demands on the equalizer. Hence, knowing the specifics of the situation in which the equalizer is supposed to be applied, one can identify which qualities to optimize for and which ones to forgo. Herein, due to the purpose of making an equalizer simple enough for anyone to use, straightforwardness and intuitiveness are the valued qualities. Therefore, as recently mentioned, the graphic equalizer has been the natural choice of design.

1.2 Objectives

The main objective in this project is to implement a graphic equalizer in MATLAB. The equalizer is intended to process audio signals in real time from a number of audio file-types as well as recorded by a microphone. The ultimate target is an equalizer with one-third octave frequency bands with 31 command gains (sliders). In addition to the main objective, there are some ancillary aims such as to make the equalizer maximally easy to use, to correct for potential errors and bugs and to develop an original look.

MATLAB App Designer is used for the programming and the construction of the graphical user interface (GUI), although the visual part of the GUI is complemented by the image editor program Pixelmator.

1.3 Theory

In this section the essential theory necessary for designing the equalizer is highlighted.

1.3.1 Analog versus digital signals

Physical audio signals that we can perceive can be modeled as analog signals, which are continuous both in time and amplitude. However, a computer can not store an infinite number of values which disallows the storing of a signal continuous in time. In order for a discrete signal to be classified as digital, the set of values that its magnitude is allowed to adopt must be finite. Thus, an analog signal can be transformed into a digital one by first discretizing it and then approximating the discretized values. Essentially, the discretization and approximation are both inevitable when storing a signal into a computer, and a prerequisite to digital signal processing [2].

1.3.2 Sampling and frames

The process by which an analog signal is converted into a digital is called sampling. Sampling a continuous signal means to periodically measure it and to round and store the measured values. When processing an audio signal, an adequate sampling frequency must be chosen in order to avoid aliasing. The *Nyquist-Shannon sampling theorem* states that the minimal sampling rate required for perfect reconstruction of the original signal is twice the maximum component frequency of the sampled signal [2]. As a consequence of this theorem, audio signals are typically sampled over 44kHz in order to exceed twice the upper limit of the human perception spectrum of sound by margin, which lies around 20kHz.

In order to facilitate the handling of the sampled signal, the computer groups the samples into arrays with predetermined lengths called data frames, depicted in figure 1. The processing of one frame is finished before the next is acquired. This frame-based data format is more efficient regarding speed and is therefore commonly used in real time systems. When choosing a suitable length of the frame, two properties must be balanced: Playback delay and computational speed. A long frame implies more delay but requires less computational power, and the reverse goes for a short frame [5].

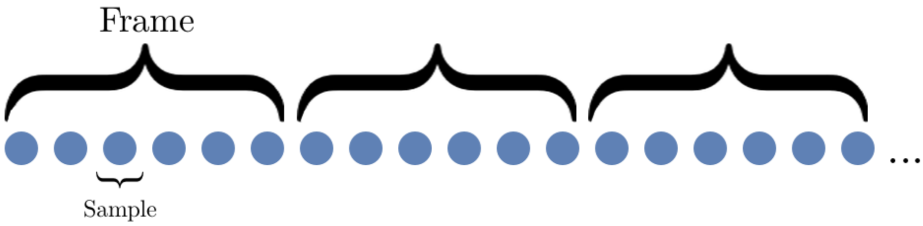

Figure 1: Illustration of the relationship between samples and frames. Here using a frame size of 6.

1.3.3 The Z-transform

Regardless of weather a signal is analog or digital, filters are not designed in the time domain. Instead, the signal is converted into a complex frequency-domain representation in order to access and manipulate the frequency components. For digital signals this conversion is done by the Z-transform. For a discrete sequence $x[k]$, its bilateral Z-transform is defined as

$$X(z) = \sum_{k=-\infty}^{\infty} x[k]z^{-k} \tag{1}$$

where k is an integer and $z = Ae^{j\omega}$ for any real numbers A and ω such that the sum in equation (1) converges [2]. A and ω denote the magnitude and the complex argument (phase angle) of z respectively. The Z-transform is derived such that $x[k-n]$ in time domain is represented by $z^{-n}X(z)$ in the complex z-domain. Consequently, difference equations in the time domain transform into algebraic equations in the z-domain.

1.3.4 FIR filters

The term FIR (finite impulse response) refers to digital filters whose transfer functions depend on a linear combination of previous input signals [2]. The transfer function $H(z)$ for a general FIR filter is given by

$$H(z) = \sum_{n=0}^{N-1} b_n z^{-n} \tag{2}$$

where b_n is a coefficient for each n and $N < \infty$. The output equation of a FIR filter in the time domain is a corollary of equation (2), written as

$$y[k] = \sum_{n=0}^{N-1} b_n x[k-n]. \tag{3}$$

1.3.5 IIR filters

As opposed to FIR filters, the transfer function of an IIR (infinite impulse response) filter depends on prior input *and* output signals [2]. The transfer function of a general IIR filter is given by

$$H(z) = \frac{\sum_{n=0}^{N} b_n z^{-n}}{1 + \sum_{m=1}^{M} a_m z^{-m}} \tag{4}$$

where a_m and b_n are unique coefficients for each m and n. The corresponding output equation can be produced by a transformation of equation (4) into the time domain by applying the inverse Z-transform, which yields

$$y[k] = \sum_{n=0}^{N} b_n x[k-n] - \sum_{m=1}^{M} a_m y[k-m]. \tag{5}$$

1.3.6 Minimum phase and the Hilbert transform

A linear, time-invariant digital filter is called minimum phase if and only if it is causal and its transfer function's poles and zeros lies inside the unit circle of the z-plane. An equivalent condition to the latter, is the requirement of stability for the system *and* its inverse. Minimum phase filters are sometimes also referred to as minimum delay filters due to the property of having the energy of its impulse response maximally concentrated at the beginning. This means that, for the set of all causal filters with impulse response $h_i[k]$ that have identical magnitude response, the minimum phase filter with impulse response $h_{mp}[k] \in h_i[k]$ will always satisfy

$$\sum_{k=0}^{K} \left|h_{mp}[k]\right|^2 \geq \sum_{k=0}^{K} \left|h_k[k]\right|^2, \qquad K = 0, 1, 2... \tag{6}$$

for the first $K+1$ samples [3]. Only for minimum phase filters, the phase response and the magnitude response are uniquely related to one another [4]. In discrete-time, this relation is given by

$$\arg[H(z)] = -\mathcal{H}\{\log(|H(z)|)\} \tag{7}$$

where $\mathcal{H}$ denotes the Hilbert transform and $H(z)$ represents the frequency response of the minimum phase filter. This relation holds true for continuous-time filters as well.

2 Method

This section outlines the sheer mathematical design of the equalizer on the one hand, and its implementation in MATLAB on the other. The mathematical design is further differentiated into filter design and equalizer design. The procedure of the equalizer design adopted in this paper was first proposed by Jussi Rämö, Vesa Välimäki and Balázs Bank in a paper released in 2014 titled *High-Precision Parallel Graphic Equalizer* [6].

2.1 Filter design

The equalizer herein is composed of 62 second order filters and one static gain filter, all arranged in parallel. The zeros of the filters are adjustable and the poles are fixed. The filter structure in question was introduced by Balázs Bank in 2007 and was not explicitly constructed for graphic equalizers [7]. However, this paper will devote no further assessment to the wider range of purposes of the filter structure, but focus on its adequacy for this graphic equalizer. The design of this filter structure is thoroughly walked through in this subsection.

2.1.1 Filter structure

The transfer function of the resulting filter is given by

$$H(z) = c_0 + \sum_{k=1}^{K} \frac{b_{k,0} + b_{k,1}z^{-1}}{1 + a_{k,1}z^{-1} + a_{k,2}z^{-2}} \tag{8}$$

where K decides the number of second order filters arranged in parallel and c_0 represents the direct path gain. This is illustrated in a block diagram in figure 2 where $X(z)$ denotes the input signal and $Y(z)$ denotes the filtered output signal.

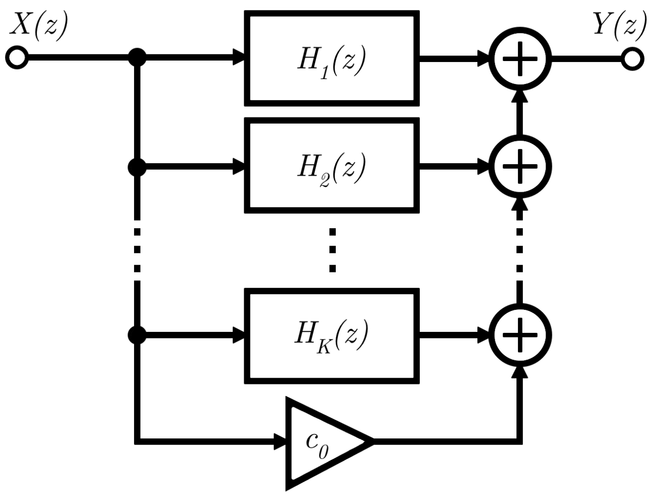

Figure 2: Illustration of the parallel filter structure.

2.1.2 Derivation of the denominators

Primarily, to start the filter design, the frequencies to which the poles are fixed have to be established. These frequencies were distributed logarithmically, meaning that they were evenly distributed on a logarithmic scale. It appears reasonable to set the pole radii $|p_k|$ such that the transfer functions of two neighbouring filters intersect at the point where each have dropped $3dB$ [6]. This is obtained by

$$\theta_k = \frac{2\pi f_k}{f_s}, \qquad k = 1, 2, ..., K \tag{9a}$$

$$|p_k| = e^{\frac{-\Delta\theta_k}{2}} \tag{9b}$$

where f_k and f_s denote the predetermined center frequency series and the sampling frequency respectively. By equation (9a) these two yield the series of normalized center frequencies, referred to as pole frequencies and denoted by θ_k. The bandwidths of the filter sections are obtained by the two adjacent pole frequencies as follows

$$\Delta\theta_k = \frac{\theta_{k+1} - \theta_{k-1}}{2}, \qquad k = 2, 3, \ldots, K-1 \tag{10}$$

with the necessary exceptions for the two filters at the upper and lower edge:

$$\Delta\theta_1 = \theta_2 - \theta_1, \tag{11a}$$

$$\Delta\theta_K = \theta_K - \theta_{K-1}. \tag{11b}$$

The coefficients in the denominator of each transfer function are given by

$$a_{k,1} = -2|p_k|\cos\theta_k \tag{12a}$$

$$a_{k,2} = |p_k|^2. \tag{12b}$$

2.1.3 Derivation of the numerators

When the parameters have been set, the problem reduces to a linear system which is a matrix representation of equation (8)

$$\mathbf{h} = \mathbf{Mb}, \tag{13}$$

where $\mathbf{M}$ is a matrix comprised of the denominators of the filter sections and their delayed counterparts, given by

$$\mathbf{M} = \begin{pmatrix} \frac{1}{den(1,1)} & \frac{e^{-j\omega_1}}{den(1,1)} & \cdots & \frac{1}{den(1,K)} & \frac{e^{-j\omega_1}}{den(1,K)} & 1 \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ \frac{1}{den(N,1)} & \frac{e^{-j\omega_N}}{den(N,1)} & \cdots & \frac{1}{den(N,K)} & \frac{e^{-j\omega_N}}{den(N,K)} & 1 \end{pmatrix} \tag{14}$$

where $den(n,k)$ denote the denominators of the filter sections: $1 + a_{k,1}e^{-j\omega_n} + a_{k,2}e^{-j2\omega_n}$. The last column is filled with ones due to multiplication by the direct path gain. $\mathbf{b}$ is a column vector with the free numerator coefficients and the direct path gain c_0. Hence, as the numerators is multiplied by their corresponding denominators, $\mathbf{h}$ becomes the vector with the resulting frequency response. $\mathbf{b}$ and $\mathbf{h}$ is given by the column vectors $\begin{pmatrix} b_{1,0} & b_{1,1} & \ldots & b_{K,0} & b_{K,1} & c_0 \end{pmatrix}^T$ and $\begin{pmatrix} H(\omega_1) & \ldots & H(\omega_N) \end{pmatrix}^T$ respectively, where N denotes the number of target frequency points. The unit of ω_n is 2π *radians/sample*.

Now, in case of an underdetermined system, there is at least one way to choose $\mathbf{b}$ such that multiplication with $\mathbf{M}$ yields the exact frequency response vector $\mathbf{h}$. If the system is overdetermined however, i.e., when there are more equations than unknowns, it is probably inconsistent (insoluble). In that case the optimal set of numerator coefficients $\mathbf{b}_{opt}$, making as tight match as possible of the frequency response vector $\mathbf{h}$ and the target response vector $\mathbf{h}_t$, is next to be calculated. This vector is obtained by the method of least-squares

$$\mathbf{b}_{opt} = \mathbf{M}^{+}\mathbf{h}_t \tag{15a}$$

$$\mathbf{M}^{+} = (\mathbf{M}^H\mathbf{M})^{-1}\mathbf{M}^H \tag{15b}$$

where "+" denotes the Moore-Penrose pseudo-inverse and "H" the conjugate transpose. The least square method minimizes the error

$$e_{LS} = \sum_{n=1}^{N} \left| H(e^{j\omega_n}) - H_t(w_n) \right|^2. \tag{16}$$

Since $\mathbf{M}$ is not affected by changes in $\mathbf{h}_t$, it can be precomputed and stored, leaving the matrix multiplication as the only operation necessary for determining $\mathbf{b}_{opt}$. This alleviates the computational burden substantially.

All of the elements in equation (15a) are complex which makes the matrix operations computationally demanding. However, equation (15a) can be manipulated so as to contain real elements only, and thus yield real numerator coefficients. By separating the real and imaginary parts of each element in $\mathbf{M}^+$ and $\mathbf{h}_t$ when constructing the target response, demands are made on the real and imaginary parts separately. This yields a linear system with the least square solution

$$\mathbf{b}_{opt} = \mathbf{M}_r^+ \mathbf{h}_{t,r} \tag{17}$$

in which

$$\mathbf{M}_r = \begin{pmatrix} \mathrm{Re}\{\mathbf{M}\} \\ \mathrm{Im}\{\mathbf{M}\} \end{pmatrix} \tag{18a}$$

$$\mathbf{h}_{t,r} = \begin{pmatrix} \mathrm{Re}\{\mathbf{h}_t\} \\ \mathrm{Im}\{\mathbf{h}_t\} \end{pmatrix} \tag{18b}$$

and where both contain real elements only. A transfer function with real coefficients has a real impulse response, which makes the frequency response conjugate symmetric [6]. Since the matrix $\mathbf{M}_r$ has its real and imaginary parts placed in tandem, the new linear system obtains twice the number of demands (equations) while retaining the same number of numerator coefficients (unknowns). The dimensions of the new matrix $\mathbf{M}_r$ are thus $(2N, 2K+1)$, in contrast to those of the former unmanipulated one $\mathbf{M}$, which are $(N, 2K+1)$. In this particular filter design, $2N$ is always chosen to be a greater number than $2K+1$. Consequently, because of an overdetermined system, the solution $\mathbf{b}_{opt}$ is obtained by the least square method.

2.1.4 Weighting

Without weighting the frequency points, the least square method will minimize the square sum of the deviations from the target response. This may seem desired, but the scale for the magnitude - the y-axis - is not linear; it is logarithmic. A certain absolute deviation makes a significantly larger deflection in the lower regions of the decibel scale than it does in the higher. If the frequency points instead are weighted, a new error is obtained, given by

$$e_{LSW} = \sum_{n=1}^{N} W(w_n) \left| H(e^{j\omega_n}) - H_t(w_n) \right|^2. \tag{19}$$

With regards to the computational complexity, the most efficient way of implementing the weighting function $W(w_n)$ is to multiply all of the elements in the modelling matrix $\mathbf{M}$ and in the target response vector $\mathbf{h}_t$ by the square root of their correspondent weighting factor $\sqrt{W(w_n)}$ before the matrix operations are executed. The downside of weighting however, granted that the weighting function depends on the target response, is that the modelling matrix can no longer be precomputed and stored. Because if it does depend on the target response, changing the target response requires recomputation of the modelling matrix.

2.2 Designing the equalizer

2.2.1 Filter distribution and order

A good resemblance between the frequency and target response is chiefly made by a high number of pole frequencies. However, in order not to dispense with the computational efficiency, this number has to be kept within reasonable limits. The equalizer presented in this paper is one with 31 command gains logarithmically distributed on the frequency spectrum from $20Hz$ to $20kHz$. This seemingly obscure number, 31, is not at all chosen arbitrarily. In a third-octave equalizer, the factor with which the bands are separated is $\sqrt[3]{2}$. Hence, the multiplications necessary to cover the span limited by $20Hz$ and $20kHz$, adds up to 31.

Now, having established the number of command gains, the number of poles is next to be settled. Herein, this number is set to twice the number of command gains by placing one pole frequency at each command frequency and one at each respective upper band edge. There is no pole frequency at the upper band edge of the 31th band though; it is instead placed below the lowest command frequency, at $10Hz$.

In the equalizer herein, the filter order exceeds the number of frequency points with one, they are 124 and 123 respectively. In general, to prevent the frequency response from oscillating between the frequency points of the target response, the filter order has to be kept below the number of frequency points. A minor transgression of the limit such as in this case however, won't necessarily create any redundant oscillations. See the full specification of the command, pole and target frequency points in table 2 in appendix.

2.2.2 Target response

A graphic equalizer is characterized by a magnitude response controlled by the command gains at a set of fixed command frequency points. In order to find a suitable magnitude response, a target response must be computed from the command gains prior to executing the matrix operations. The configuration of the command gains constitutes a discrete function on a logarithmic frequency grid, which the target magnitude response vector is supposed to emulate. However, the target response has a higher frequency resolution than the slider configuration, which is therefore obtained by interpolating it.

The interpolation can be done by first fitting a curve to the data points - constructing a target response - and then sampling it in the desired frequency points. This curve is created with the MATLAB function `pchip` (shape-preserving piecewise cubic interpolation), that takes the values and the their derivatives into account, see figure 3.

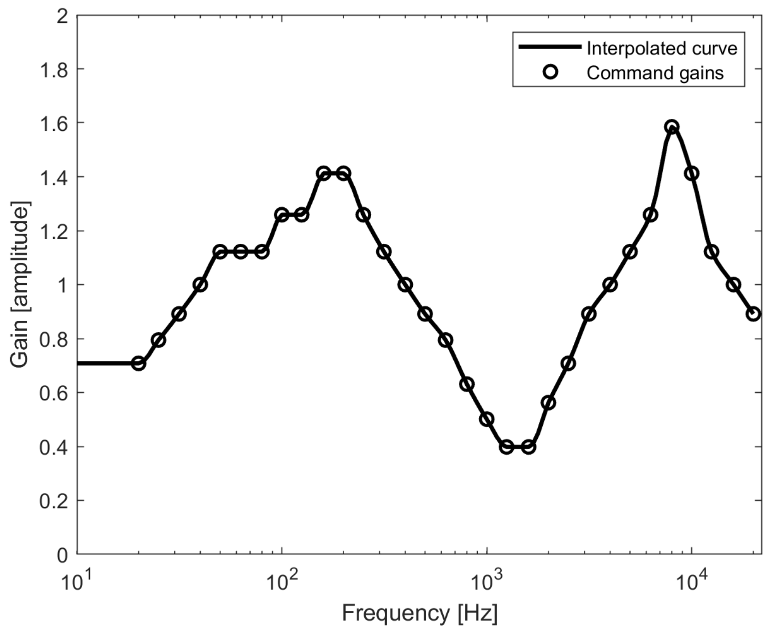

Figure 3: Construction of target magnitude response (`pchip`-interpolation).

Now, having finished the magnitude response, the phase response is yet to be constructed to complete the frequency response. With regard to a parallel graphic equalizer, the appropriate way of determining the phase response is such that the frequency response represents a minimum phase system [6]. Knowing the target magnitude response, as described in section 1.3.6, this phase response can be distinctly determined by equation (7). An interpolated curve on the whole frequency spectrum ($-\frac{f_s}{2}$ to $\frac{f_s}{2}$), symmetric around the y-axis, was sampled at 2^{16} linearly distributed frequency points. This frequency magnitude response vector is converted into a frequency phase response vector by the Hilbert transformation, which is then resampled in the target frequency points. The Hilbert transform is calculated with the built in MATLAB function `hilbert`.

2.2.3 Weighting of frequency points

The frequency response is enhanced by assigning the frequency points the appropriate weighting function $W(w_n) = 1/\left|H_t(w_n)\right|^2$ [6]. The modelling matrix and the target response is thus multiplied by the weighting factor $\sqrt{W(w_n)} = 1/\left|H_t(w_n)\right|$ prior to the execution of the matrix operations. This choice of weighting derives from the higher sensitivity to deviations at low magnitudes and it equates for this dissimilarity. This yields an error minimization with respect to the frequency magnitude response's *relative* deviation from the target magnitude response.

2.3 Implementation

As the title suggests, this chapter outlines the implementation of the equalizer. The code was written in MathWork's application development environment *MATLAB App Designer* with *Audio Toolbox 2.0* installed.

2.3.1 Program structure

For educational purposes, a simplified model of the program will be illustrated together with its functions. Figure 4 is a simplified scheme of the application as a whole. This chart provides a schematic overview of the overall structure of the program. However, most of the components require further clarification. The operation "Processing the frame" is particularly complex, and is therefore illustrated by an ancillary flowchart, shown in the next few paragraphs.

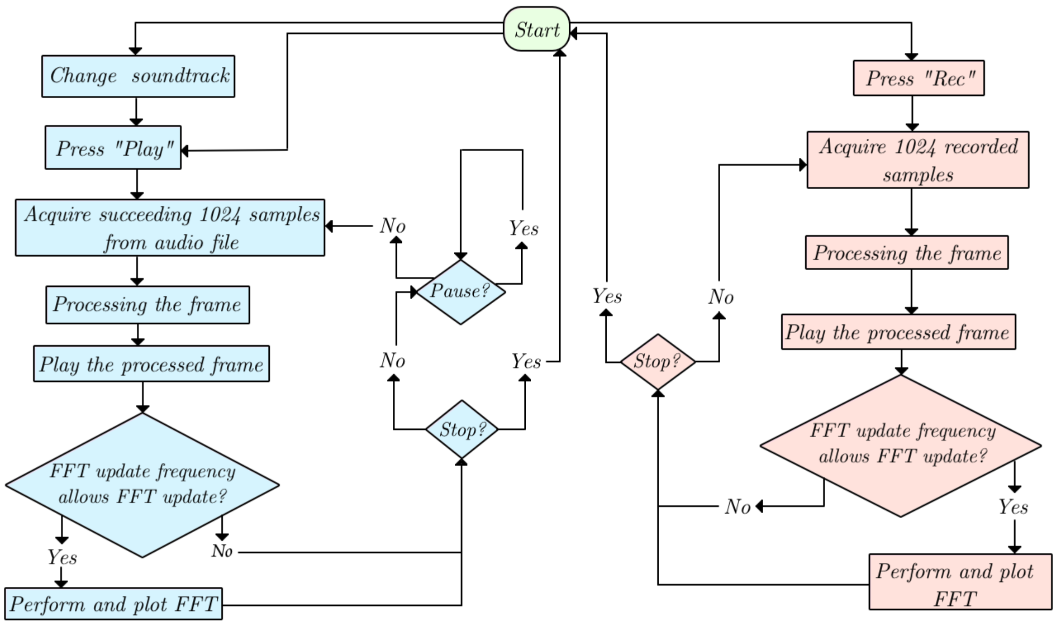

Figure 4: Flowchart of the main loop.

2.3.2 Processing the frame

Frames are, as highlighted in the theory section, arrays of samples with a certain length serving the purpose of enhancing the computational efficiency. In this case the length of the frame array is 1024 which was concluded to be a moderate compromise between delay and computational speed.

Shortly, the processing of frames is the intermediary between the input and the output signals. It carries out the filtering of the signals, i.e., the operation of manipulating the input signals into the desired output signals. In this context, the desired output signals are specified by manually customizing the system's frequency response through the use of 31 sliders. To make the application more convenient, some predetermined configurations of the sliders were implemented, called presets. Every music genre have a tailor-made slider configuration ascribed to it, named by the genre they aspire to suit. When selecting a preset, the current slider configuration shifts instantaneously and so does the sound. However, the reason for the specifics of the configurations, and why a given one suits a certain genre, is outside the domain of this paper and is thus not delved into here.

Figure 5 displays a flowchart of the process where the big square in bright blue is a closer look into the operation *processing the frame.* There is no difference in the processing of the frames between audio files and recorded sound.

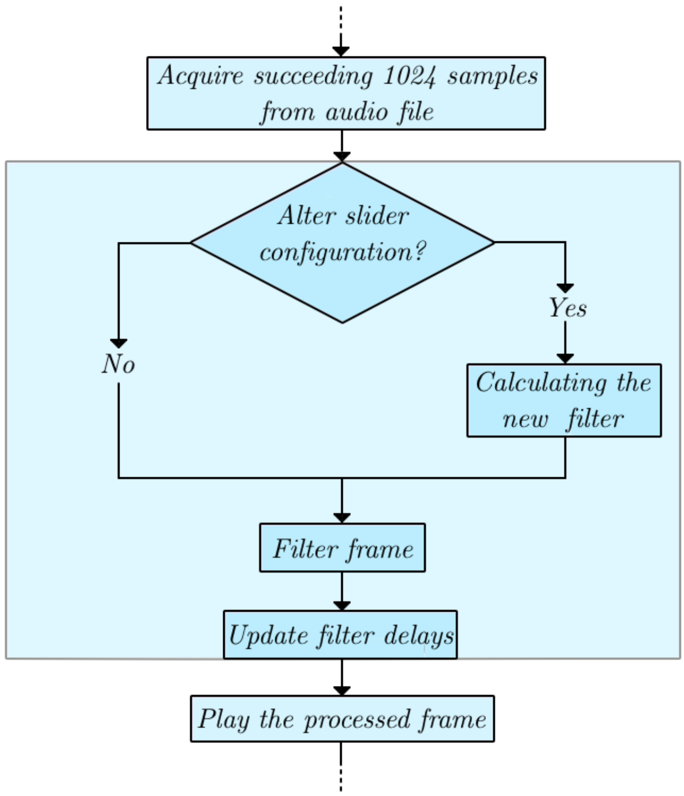

Figure 5: Flowchart of the operation *processing the frame*.

The second process from the bottom in figure 5 is denoted as "update filter delays". The term *filter delay* refers to an accumulation of previous values of the input and the output signal. The "Filter frame" process is essentially the operation that applies the calculated filters to the input signal, which in this case was done by the MATLAB function `filter`. The algorithm in the `filter`-function is an implementation of equation (5). See section 2.1 for a more thorough explanation of the process "Calculating the new filter" in figure 5.

2.3.3 FFT plot

For aesthetic reasons, the equalizer's GUI includes a plot of the frequency content of the audio signal, displayed in real-time. This can be done using the built in MATLAB function `fft` that calculates the *Fast Fourier Transform* of a discrete vector. In this case, the FFT is applied to each processed frame and then plotted before the next frame is acquired. However, updating a plot in MATLAB is computationally demanding for most computers. So in order to make the computational load of the application commensurate with the power of the computer using it, the frequency with which the plot is updated has been made adjustable. The plot can be chosen to be updated every frame, all the way down to every 50th frame, as well as completely turned off.

2.3.4 Graphical User Interface

The term GUI refers to the interface with which the user interact. It includes all the buttons, sliders and knobs as well as the appearance of the application. At first glance the interface may seem complex whereas, in fact, it was quite straight forward to create. This is due to MATLAB App Designer which allows the user to drag and drop visual components to lay out the design of the GUI. Additionally, the behavior of the components can easily be defined. The visual features of the interface was complemented in the program Pixelmator, in which two pictures were drawn: a background frame for the buttons and sliders, and a frame for the "FFT-screen". However, the version of the application with the two pictures is only compatible with MATLAB R2019a.

2.3.5 Frequency response

The equalizer is primarily evaluated on the basis of the match between the target magnitude response and the actual magnitude response. However, there is no straight forward way of assessing this match. MATLAB lacks the built-in function for plotting the frequency response of a large number of paralleled filters. An alternative way of plotting has thus to be found in order to circumvented this obstacle.

The equalizer consists of IIR-filters, which can be approximated by a high order FIR filter. The sample values of an IIR filter's impulse response correspond to the coefficients in its FIR-counterpart. The more samples (coefficients) taken into account, the better the approximation. Furthermore, since the filters are all linear, the impulse response of the whole filter structure is simply obtained by adding them up. The impulse response was generated by the MATLAB function `impz`, and the FIR filter approximation was then plotted by the function `freqz`. In this case the system was approximated by a FIR filter with 40000 coefficients, since a further increase in filter order would imply a negligible enhancement in accuracy.

3 Results

The final version of the audio equalizer of this paper is a high-precision parallel graphic equalizer with fixed poles. In this section the finished equalizer is laid forth in its entirety. The results of the previous section (method) is exhaustively accounted for here. Concisely, the method was outlining *how* the equalizer was constructed, whereas this section declares *what* has been built, as well as what has been observed in relationship to the equalizer and its features. The next section, discussion, is in contrast contemplating *why* these observations have been made.

3.1 Functionality

The application, depicted in figure 6, has an originally looking and straight forward interface. In addition to the slider plate, i.e., the actual equalizer (at the bottom of figure 6), the application is equipped with some other features making it easier and more engaging. The music can be paused, stopped and changed at any time. While at play, the intensities of all frequencies can be displayed in real time, with an adjustable refreshing pace, in the window right above the slider plate. Furthermore, 13 different presets were implemented out of which three are named *rock*, *hip-hop* and *pop* by the genre they are customized for.

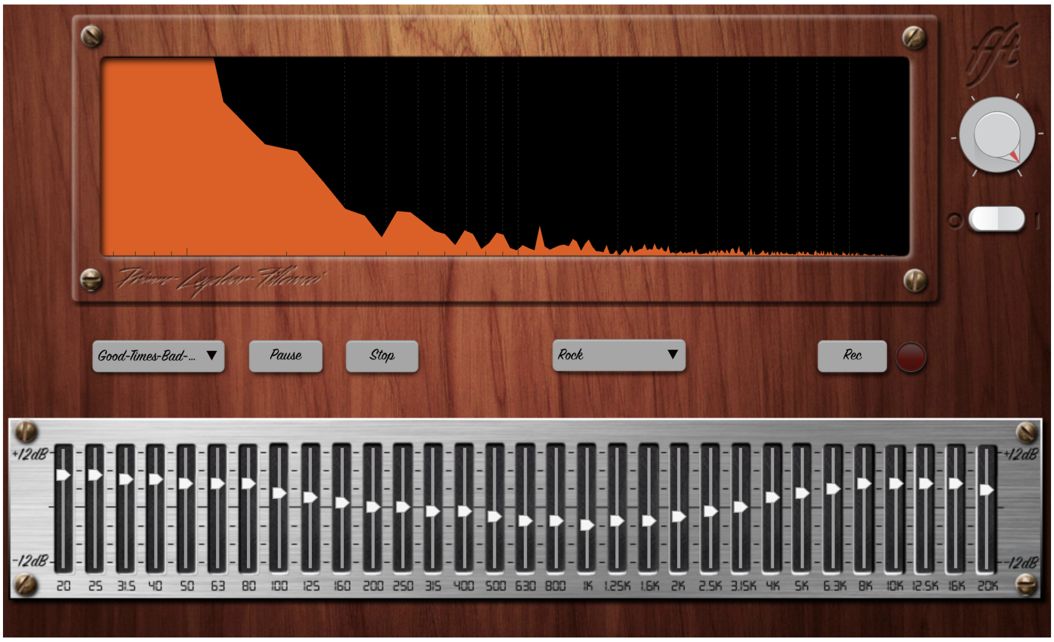

Figure 6: Interface of the application.

Various functionality tests highlighted some important limitations of the program. Making additional demands on a computer, e.g., high frequent updating of the FFT-plot, at real-time playback turned out to impair its performance and cause playback lag. The higher the update frequency, the more vulnerable to slider movements the computer became. Some of the most computationally demanding operations during playback were timed and the average results are shown in table 1 for two different computers: *MacBook Pro - 2.3 GHz Intel Core i5 - 8GB RAM* and *ASUS ZenBook UX305CA - Intel(R) Core(TM) m3-6Y30 CPU @ 0.90 GHz 1.51 GHz - 8 GB RAM.* Table 1 indicates by part what was already known: The *Macbook Pro* manages a higher update frequency of the FFT plot, without playback lag, than the *ASUS ZenBook*.

Table 1: Computation time for some demanding operations for two different computers.

Computer	MacBook Pro	ASUS ZenBook UX305CA
FFT plot update	8.5 ms	15 ms
Filter calculation (without weighting)	10 ms	20 ms
Filter calculation (with weighting)	70 ms	140 ms

3.2 Accuracy

With regards to the accuracy of the equalizer, everything has unfolded according to plan. This equalizer design provided a tight match between the actual frequency response and the target response, alluded to by its epithet "*high-precision*". This was partly comfirmed by the immediate response of the equalizer in the form of a shift in output sound as the sliders were pulled while music was played through it. Furthermore, with regards to the phase response and its implications on the sound, no delays or other undesirable noise was perceived.

An experiment was conducted with the quality of the filters, in which the equalizer was exposed to an input signal in the form of a single sine wave, whose frequency matched one of the center frequencies of a band. Then, the sliders of the adjacent bands were moved, and the intensity of the sound was monitored by listening and watching the FFT plot. No difference was perceived during these movements. Fortunately, the slider corresponding to the frequency of the sine wave did affect the output sound.

The frequency response of the system without implemented weighting, displayed in figure 7 where the command gains are set to either 13 or -13 dB, exhibits the largest errors at the low gain frequency points with high gain neighbors.

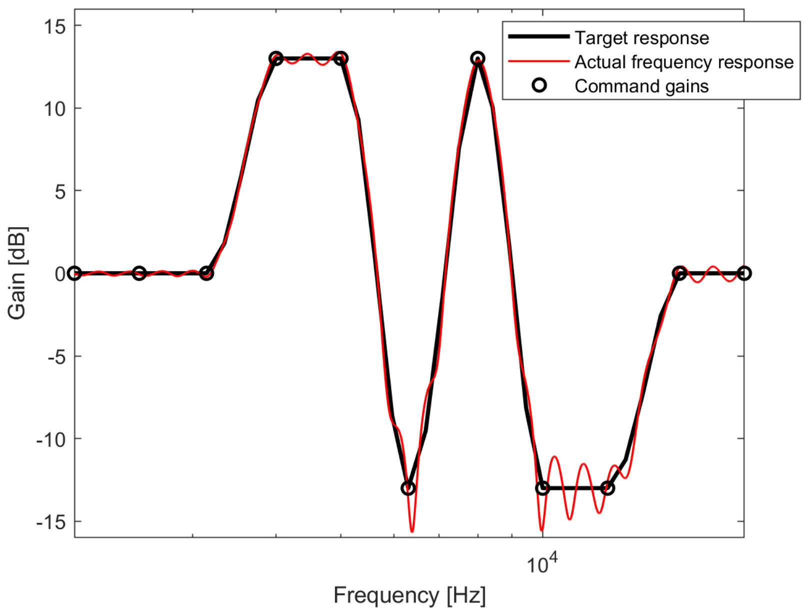

Figure 7: Frequency response without weighting from the 20th band.

The equalizer with implemented weighting on the other hand, displayed in figure 8, exhibited a tighter match between the frequency and target magnitude response. The enhancement of the weighting was most evident at the frequency points with low gain. At high gain in contrast, there was only modest differences in magnitude response between the weighted and the unweighted system.

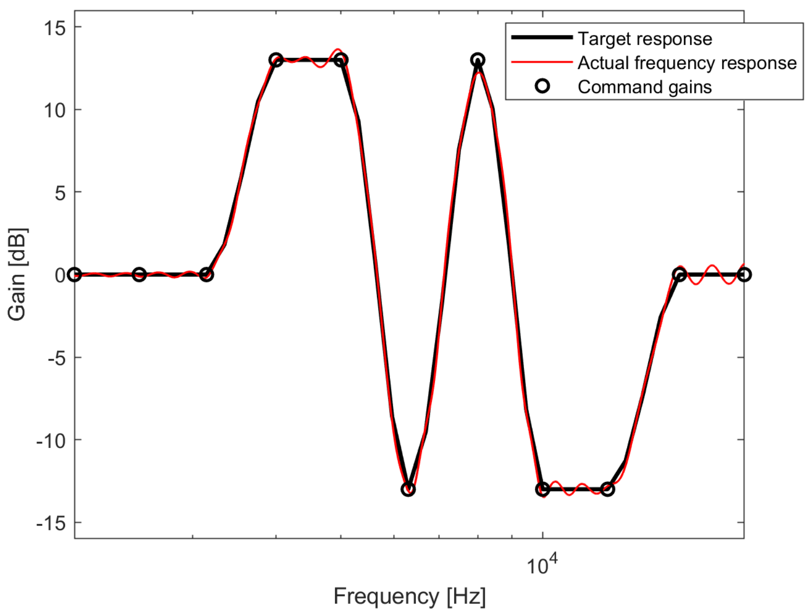

Figure 8: Weighted frequency response from the 20th band.

Figure 9 displays the frequency response of the filters separately, by the same slider configuration as in figure 7 and 8. The command band filters in blue are the ones with pole frequencies corresponding to the centre frequencies of the bands, whereas the slave filters in dashed pink are the auxiliary ones with pole frequencies at the bands' upper edges, all tabulated in table 2 in appendix.

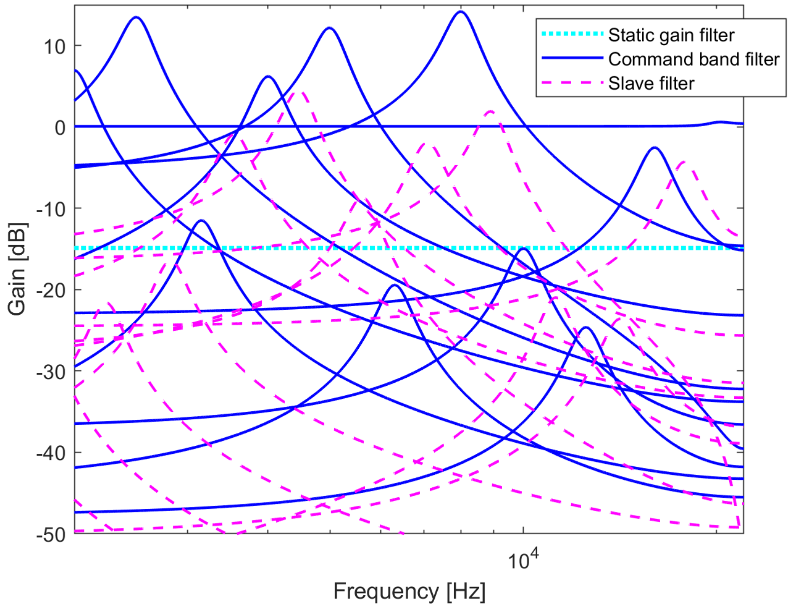

Figure 9: The filter sections plotted separately from the 20th band.

4 Discussion

This section is interpreting the findings of this project in the light of what is already known in this domain. The conclusion by contrast - the next section - is supposed to reconnect to the introduction as well as assessing the degree to which the objectives have been achieved.

As experimented with and presented in the previous section, moving a slider during playback proved sometimes to cause playback lag, especially when refreshing the FFT at a high pace simultaneously. When a slider is moved the target response changes, whereupon new numerator coefficients have to be generated in order for the frequency response to fit the new target response. This requires matrix operations and recomputation of the matrix $\mathbf{M}_r^+$. The need for recomputation is because the matrix is weighted with a factor dependent on the target response. The time required to execute such complex calculations might be sufficient to cause perceivable delay, depending on the computational power of the computer. This means that these calculations is the source of the playback lag, which also explains why it is caused by pulling the sliders.

Furthermore, this playback lag is the price to pay for high accuracy. As outlined in the introduction, an appropriate compromise between accuracy and computational speed is imperative to find. However, slider movements followed by lag may indicate that the compromise is skewed: Too much computational speed has been traded off for accuracy. Therefore, In spite of the enhancement in accuracy introduced by the weighting, it might be worth to consider omitting. Since the equalizer is dedicated to the typical person rather than to sound engineers, it would be more appropriate to skew the trade-off in the other direction (i.e., towards low computation time). If the equalizer is supposed to be available for anyone, the running time must be low enough so that even a slower computer can use it without playback lag. Herein, the weighted system is nevertheless concluded to be sufficiently fast for this purpose. As a means of reducing running time in case of lag, the program is instead constructed to allow the user to manually disable the FFT-plot.

With regards to the weighting $W(w_n) = 1/\left|H_t(w_n)\right|^2$ and its impact on the frequency response, the weighted system exhibited a significantly tighter match between the magnitude response and the target response, than did the unweighted system. With this particular weighting, the deviation (error) subject to minimization is a relative one, as opposed to the system without weighting, wherein it is absolute. Instead of minimizing the square sum of the absolute deviations, the weighted system minimizes the square sum of the *factors* with which the frequency response deviates from the target response. As is well known, a difference in decibel is nothing other than a factor, which hence makes the weighting factor proposed herein the optimal one for error minimization on a decibel scale.

As laid forth in the results, the minimum phase system created an output sound free from perceivable noise. Since the impulse responses of a minimum phase systems has their maximal energy at the beginning, as outlined in section 1.3.6, all the unwanted noise will be preceded by a sound peak. Such an impulse response is shown in figure 10. By virtue of this, the human ear will barely apprehend the noise because of the deafening effect of the sound peak preceding it. The reverse to this, i.e., when the noise comes before the sound peaks, is called preringing. In such cases, the ears have conversely not been deafened by some intense sound and are hence, as by default, susceptible to the most modest of noises. Thus, in a phase system lacking this property, the output sound would perhaps contain perceivable distortions.

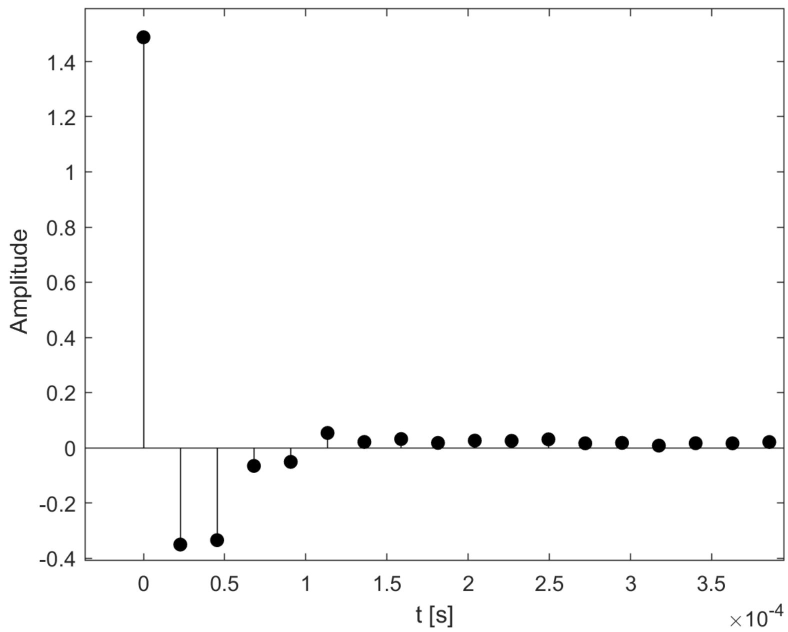

Figure 10: The first 18 values in the impulse response of the whole filter structure for the "rock" preset.

Ultimately, since minimizing the errors of the entire parallel filter structure instead of the filters one by one, the filter structure is jointly optimized to fit the target response. Such filter structures prevent interaction between adjacent filters and does hence outperform the ones with filters optimized separately. This property, vague as it might seem, is key to a high accuracy and has thus played a fundamental role in propelling this equalizer to the advanced tool it now has become.

5 Conclusion

In this paper, a graphic equalizer with 31 bands has been constructed with an intuitive interface, in complete keeping with the objectives. Additionally, the equalizer is to be found in MathWorks, free to download [8]. 31 band equalizers with high precision are at large quite scarce, let alone such equalizers free of charge.

The application includes a number of presets, i.e., beforehand programmed configurations of the volume sliders. These offers the user to swiftly, with one button, customize the frequency response for the genre of the track at play. Furthermore, the high frequency resolution and accuracy of the equalizer makes it a good subject for experimentation. For example, one can try to move a slider and pay attention to the subsequent shift of the output sound. Or why not create an own configuration of the sliders according to preferences. because of a manually adjustable frequency magnitude response, the equalizer enables enhancement of the music experience irrespective of the genre or style.

The subject of this paper is a *graphic* equalizer with *parallel* filters. The former epithet denotes a frequency magnitude response regulated with frequency-specific sliders, and the latter means a frequency response formed by the *sum* of the individual filters. The other main type of equalizer is called *parametric* equalizer, which enables manual control over more parameters but does consequently take a professional to handle. The altering of the frequency response is faster, but not as straight forward as in graphic ones. As to the filters, the alternative to a parallel filter structure is a cascaded one, in which the frequency response is obtained by the *product* of the individual filters.

The two sorts of equalizers as well as filter structures have their own specialties and shortcomings. The specifics of the equalizer in this paper are chosen such as to widen the range of potential users. For example, straightforwardness has been prioritized over detail control and possibility for quick altering of the target response.

A generic description of the multifaceted term *audio equalizer* is a tool enabling regulation and adjustment of electronic signals. Reproduction of music is only one of the facets which too, in its turn, can be more nuanced. For the sake of an example different from music reproduction, the tool is also applied to correct frequency responses of telephone lines [1].

For deep insight into the process of creating a functioning equalizer, this paper as a whole has to be read. Nonetheless, the overarching structure of the process can be laid out briefly. The filter structure is the foundation upon which the equalizer has been built. In its essential, the structure was made by first placing all of the denominators of the individual filter sections into a matrix. Then, a vector of their corresponding numerators was calculated such that multiplication with the denominator matrix yielded the desired frequency response. Somewhat simplified, the discrete function that constitutes the desired frequency response is an interpolated version of the one made up by the slider configuration. Consequently, the filters are controlled by the sliders, and thus, so is the output sound.

Essentially, the epithet *audio* means *sound*, which is something we are familiar with and can apprehend. *Audio equalizer* means an equalizer operating on sound. This virtue dispels part of the obscurity with regards to the properties of a signal, e.g., frequency and intensity. People with unimpaired hearing can easily distinguish between different frequency levels. A high frequency is recognized as shrill, and a low as deep. A trained ear can apprehend even a tiny offset of a tone from its key. This sensitivity to frequencies makes us capable of appreciating an equalizer with a high frequency resolution, and the freedom that comes along with it, i.e., the enablement of the user to experiment his or her way to a favorite configuration as well as to just play around with the frequency response. The equalizer herein is therefore constructed with 31 bands, which is a large number in comparison to other audio equalizers.

Differences in the intensity of the sound manifest as different volumes, which after all is the only adjustable property in audio equalizers. The bass preset for example, does not lower any frequencies, but turns up the volume on the ones that are already low, and vice versa. A high tone is thus not made low by such a preset; it is made weak.

For some computers, the complexity of the calculations followed by the slider movements were high enough to cause playback lag. To overcome this, a due subject for future studies might be to evaluate a similar equalizer constructed in a low-level programming language. Specific code that calculates little but what is necessary could thus be written, which could improve the efficiency of the application. Such a study could for example include a comparison of computational efficiency between the equalizer constructed in the low-level language and an identical one designed in MATLAB. That would be a natural extension of this study.

Holistic presentations of the designing procedure of advanced equalizers is something prior research falls short of. This paper outlines the filter design, sketches the design of the equalizer and also, ultimately, demonstrates the implementation of the equalizer. The overview, rendered by presenting the construction of an equalizer from scratch, enables researchers and students to design one themselves as well as to understand the procedure. It helps the reader to see the big picture, which is somewhat unique to this paper. Furthermore, designing an equalizer is an excellent gateway into the realm of signal processing, which otherwise may be daunting to enter. This paper could thus play an important role in bridging that gap. After all, the more brains operating in a domain the higher the probability for a breakthrough.

Appendix A

Table 2: Fixed frequency points specification in Hz where f_g, f_k and f_n denote the command, pole and target frequency points respectively.

Band			**1**				**2**				**3**	
f_g			20				25				31.5	
f_k	10		20		22.4		25		28.2		31.5	
f_n	10	14	20	21.2	22.4	23.7	25	26.6	28.2	29.9	31.5	33.5
Band			**4**				**5**				**6**	
f_g			40				50				63	
f_k	35.5		40		44.7		50		56.2		63	
f_n	35.5	37.6	40	42.2	44.7	47.3	50	53.1	56.2	59.6	63	66.9
Band			**7**				**8**				**9**	
f_g			80				100				125	
f_k	70.8		80		89.1		100		112		125	
f_n	70.8	75	80	84.2	89.1	94.5	100	106	112	119	125	133
Band			**10**				**11**				**12**	
f_g			160				200				250	
f_k	141		160		178		200		224		250	
f_n	141	150	160	168	178	188	200	211	224	237	250	266
Band			**13**				**14**				**15**	
f_g			315				400				500	
f_k	282		315		355		400		447		500	
f_n	282	299	315	335	355	376	400	422	447	473	500	531
Band			**16**				**17**				**18**	
f_g			630				800				1k	
f_k	562		630		708		800		891		1k	
f_n	562	596	630	669	708	750	800	842	891	945	1k	1.06k
Band			**19**				**20**				**21**	
f_g			1.25k				1.6k				2k	
f_k	1.12k		1.25k		1.41k		1.6k		1.78k		2k	
f_n	1.12k	1.19k	1.25k	1.33k	1.41k	1.5k	1.6k	1.68k	1.78k	1.88k	2k	2.11k
Band			**22**				**23**				**24**	
f_g			2.5k				3.15k				4k	
f_k	2.24k		2.5k		2.82k		3.15k		3.55k		4k	
f_n	2.24k	2.37k	2.5k	2.66k	2.82k	2.99k	3.15k	3.35k	3.55k	3.76k	4k	4.22k
Band			**25**				**26**				**27**	
f_g			5k				6.3k				8k	
f_k	4.47k		5k		5.62k		6.3k		7.08k		8k	
f_n	4.47k	4.73k	5k	5.31k	5.62k	5.96k	6.3k	6.69k	7.08k	7.5k	8k	8.42k
Band			**28**				**29**				**30**	
f_g			10k				12.5k				16k	
f_k	8.91k		10k		11.2k		12.5k		14.1k		16k	
f_n	8.91k	9.45k	10k	10.6k	11.2k	11.9k	12.5k	13.3k	14.1k	15k	16k	16.8k
Band			**31**									
f_g			20k									
f_k	17.8k		20k									
f_n	17.8k	18.8k	20k									

Appendix B

MATLAB App Designer code:

```
classdef app1gui < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure       matlab.ui.Figure
        Image          matlab.ui.control.Image
        Slider_1       matlab.ui.control.Slider
        Slider_2       matlab.ui.control.Slider
        Slider_3       matlab.ui.control.Slider
        Slider_4       matlab.ui.control.Slider
        Slider_5       matlab.ui.control.Slider
        Slider_6       matlab.ui.control.Slider
        Slider_7       matlab.ui.control.Slider
        Slider_8       matlab.ui.control.Slider
        Slider_9       matlab.ui.control.Slider
        Slider_10      matlab.ui.control.Slider
        Slider_11      matlab.ui.control.Slider
        Slider_12      matlab.ui.control.Slider
        Slider_13      matlab.ui.control.Slider
        Slider_14      matlab.ui.control.Slider
        Slider_15      matlab.ui.control.Slider
        Slider_16      matlab.ui.control.Slider
        Slider_17      matlab.ui.control.Slider
        Slider_18      matlab.ui.control.Slider
        Slider_19      matlab.ui.control.Slider
        Slider_20      matlab.ui.control.Slider
        Slider_21      matlab.ui.control.Slider
        Slider_22      matlab.ui.control.Slider
        Slider_23      matlab.ui.control.Slider
        Slider_24      matlab.ui.control.Slider
        Slider_25      matlab.ui.control.Slider
        Slider_26      matlab.ui.control.Slider
        Slider_27      matlab.ui.control.Slider
        Slider_28      matlab.ui.control.Slider
        Slider_29      matlab.ui.control.Slider
        Slider_30      matlab.ui.control.Slider
        Slider_31      matlab.ui.control.Slider
        UIAxes         matlab.ui.control.UIAxes
        TrackDropDown  matlab.ui.control.DropDown
        PlayButton     matlab.ui.control.Button
        StopButton     matlab.ui.control.Button
        DropDown       matlab.ui.control.DropDown
        RecButton      matlab.ui.control.Button
        Lamp           matlab.ui.control.Lamp
        Switch         matlab.ui.control.RockerSwitch
        Knob           matlab.ui.control.Knob
        Image2         matlab.ui.control.Image
    end
```

```

    %Properties that corresponds to app functionality
    properties (Access = private)

        isRec = 0;
        isPlay = 0;
        fs = 44100;
        fk = [10 20 22.4 25 28.2 31.5 35.5 40 44.7 50 56.2 ...
            63 ...
            70.8 80 89.1 100 112 125 141 160 178 200 224 ...
                250 ...
            282 315 355 400 447 500 562 630 708 800 891 ...
                1000 ...
            1120 1250 1410 1600 1780 2000 2240 2500 2820 ...
            3150 3550 4000 4470 5000 5620  6300 7080 8000 ...
            8910 10000 11200 12500 14100 16000 17800 20000];

        wn = 2*pi*[10 14 20 21.2 22.4 23.7 25 26.6 28.2 ...
            29.9 ...
            31.5 33.5 35.5 37.6 40 42.2 44.7 47.3 50 53.1 ...
            56.2 59.6 63 66.9  70.8 75 80 84.2 89.1 94.5 ...
                100 ...
            106 112 119 125 133 141 150 160 168 178 188 200 ...
            211 224 237 250 266 282 299 315 335 355  376 ...
            400 422 447 473 500 531 562 596 630 669 708 ...
            750 800 842 891 945 1000 1060 1120 1190 1250 ...
            1330 1410 1500 1600 1680  1780 1880 2000 2110 ...
            2240 2370 2500 2660 2820 2990 3150 3350 ...
            3550 3760 4000 4220 4470 4730 5000 5310 5620 ...
                5960 ...
            6300 6690 7080 7500 8000 8420 8910 9450 10000 ...
            10600 11200 11900 12500 13300 14100 15000 16000 ...
            16800 17800 18800 20000];

         fg = [20 25 31.5 40 50 63 80 100 125 160 200 250 ...
             315 400 500  630 800 1000 1250 1600 2000 2500 ...
             3150 4000 5000 6300 8000 ...
             10000 12500 16000 20000];

        isStop = 0;

    end

    methods (Access = private)

        %Calculation of the denominator of the k:th filter
        function a = den(app,k,Fs)
            thetak = 2*pi*app.fk(k)/Fs;
            if k>1 && k<62
            dthetak = ...
                (2*pi*app.fk(k+1)/Fs-2*pi*app.fk(k-1)/Fs)/2;
            elseif k == 1
                dthetak = 2*pi*app.fk(2)/Fs-2*pi*app.fk(1)/Fs;
            else
```

```
                    dthetak = ...
                        2*pi*app.fk(62)/Fs-2*pi*app.fk(61)/Fs;
                end
                pk = exp(-dthetak/2);
                a = [1 -2*pk*cos(thetak) pk^2];

            end

            %Calculation of the modelling matrix
             function Mrplus = Mrp(app,Fs)

                M = zeros(123,124);
                M(:,125) = ones(123,1);
                sqW = Weight(app,app.Slider_1.Value,...
                    app.Slider_2.Value,...
                    app.Slider_3.Value,app.Slider_4.Value,...
                    app.Slider_5.Value,app.Slider_6.Value,...
                    app.Slider_7.Value,app.Slider_8.Value,...
                    app.Slider_9.Value,app.Slider_10.Value,...
                    app.Slider_11.Value,app.Slider_12.Value,...
                    app.Slider_13.Value,app.Slider_14.Value,...
                    app.Slider_15.Value,app.Slider_16.Value,...
                    app.Slider_17.Value,app.Slider_18.Value,...
                    app.Slider_19.Value,app.Slider_20.Value,...
                    app.Slider_21.Value,app.Slider_22.Value,...
                   app.Slider_23.Value,app.Slider_24.Value,...
                   app.Slider_25.Value,app.Slider_26.Value,...
                   app.Slider_27.Value,app.Slider_28.Value,...
                   app.Slider_29.Value,app.Slider_30.Value,...
                   app.Slider_31.Value);

                for n = 1:123
                    for k = 1:62
                       M(n,2*k-1) = 1/(den(app,k,Fs)*...
                           [1; exp(-app.wn(n)/Fs*1i); ...
                           exp(-2*app.wn(n)/Fs*1i)]);
                           M(n,2*k) = exp(-app.wn(n)/Fs*1i)/...
                           (den(app,k,Fs)*[1; ...
                              exp(-app.wn(n)/Fs*1i);...
                           exp(-2*app.wn(n)/Fs*1i)]);
                    end
                    M(n,:) = M(n,:)*sqW(n);
                end

                Mr = [real(M);imag(M)];

                Mrplus = (transpose(Mr)*Mr)\transpose(Mr);

            end

            %Calculation of the target response vector
            function htr = target(app,G1,G2,G3,G4,G5,G6,G7,...
                G8,G9,G10,G11,G12,G13,G14,G15,G16,G17,G18,G19,...
                G20,G21,G22,G23,G24,G25,G26,G27,G28,G29,G30,G31)
```

```
            y = 10.^(1/20*pchip([-flip(app.fg) app.fg],...
                [flip([G1,G2,G3,G4,G5,G6,G7,G8,G9,G10,...
                G11,G12,G13,G14,G15,G16,G17,G18,G19,G20,...
                G21,G22,G23,G24,G25,G26,G27,...
                G28,G29,G30,G31])...
                [G1,G2,G3,G4,G5,G6,G7,G8,G9,G10,G11,G12,...
                G13,G14,G15,G16,G17,G18,G19,G20,G21,G22,...
                G23,G24,G25,G26,G27,G28,G29,G30,G31]],...
                linspace(-app.fs/2,app.fs/2,2^16)))';

            phase = unwrap(imag(-hilbert(log(y))));
            phase = phase(32769:64124);

            i = 1;
            fi = zeros(1,123);
            for w = app.wn/(2*pi)
                if round(w*length(phase)/21100) > length(phase)
                    fi(i) = phase(length(phase));
                else
                fi(i) = phase(round(w*length(phase)/21100));
                end
                i=i+1;
            end

            htr = [real(exp(1i*fi'));imag(exp(1i*fi'))];

        end

        %Calculation of the optimal numerator coefficients
        function popt = num(app,htr,Mrplus)
            popt = Mrplus*htr;

        end

        %Filtering process algorithm
        function yk = filterNew(app,bWithIndex,xk1,...
            ybuffer,xbuffer,Fs,n)
            yk = filter(bWithIndex',den(app,n,Fs),...
                xk1,filtic(bWithIndex',den(app,n,Fs),...
                ybuffer,xbuffer));


        end

        %Calculation of the Weighting factors
        function sqW = ...
           Weight(app,G1,G2,G3,G4,G5,G6,G7,G8,G9,...
            G10,G11,G12,G13,G14,G15,G16,G17,G18,G19,G20,G21,...
            G22,G23,G24,G25,G26,G27,G28,G29,G30,G31)
            ht = 10.^(1/20*pchip([10 app.fg],[G1 ...
               G1,G2,G3,G4,...
                G5,G6,G7,G8,G9,G10,G11,G12,G13,G14,G15,G16,...
                G17,G18,G19,G20,G21,G22,G23,G24,G25,G26,G27,...
```

```
                G28,G29,G30,G31],app.wn/(2*pi)))';
            sqW1 = zeros(123,1);
            for i = 1:123
                sqW1(i) = 1/ht(i);
            end
            sqW = sqW1;
        end
        end


    % Callbacks that handle component events
    methods (Access = private)

        % Code that executes after component creation
        function Start(app)

            app.Lamp.Enable = 'off';
            tracks = struct2cell(dir('*.mp3'));
            tracks = tracks(1,:);
            app.TrackDropDown.Items = tracks;
            wav = struct2cell(dir('*.wav'));
            wav = wav(1,:);
            for i = 1:length(wav)
                app.TrackDropDown.Items(i+...
                length(tracks(1,:))) = wav(i);
            end
            if isempty(app.TrackDropDown.Items) == 1
                uialert(app.UIFigure,...
                ['Your current folder does not contain any ...
                   audio files'],...
                    'Info','Icon','info');
            end

        end

         % Button pushed function: PlayButton
        function play(app, event)
        if isempty(app.TrackDropDown.Items) == 1
            uialert(app.UIFigure,'No audio file found',...
                'Error','Icon','error');
        elseif app.isRec == 1
            uialert(app.UIFigure,...
                'Stop recording before playback of a file',...
                'Tip','Icon','warning');

        elseif strcmp(app.PlayButton.Text,'Play') && ...
                app.isPlay == 0 && app.isRec == 0

            %Acquiring the audio file
            fileReader = dsp.AudioFileReader(...
                char(app.TrackDropDown.Value),...
                'SamplesPerFrame',1024);
            deviceWriter = audioDeviceWriter('SampleRate',...
                fileReader.SampleRate);
```

```
            app.PlayButton.Text = 'Pause';
            app.isPlay = 1;
            app.fs = fileReader.SampleRate;
            Fs = app.fs;

            %Initializing the delays
              xbuffer1 = 0;
              xbuffer2 = 0;
              ybuffer1 = zeros(62,3);
              ybuffer2 = zeros(62,3);


            S1 = app.Slider_1.Value;
            S2 = app.Slider_2.Value;
            S3 = app.Slider_3.Value;
            S4 = app.Slider_4.Value;
            S5 = app.Slider_5.Value;
            S6 = app.Slider_6.Value;
            S7 = app.Slider_7.Value;
            S8 = app.Slider_8.Value;
            S9 = app.Slider_9.Value;
            S10 = app.Slider_10.Value;
            S11 = app.Slider_11.Value;
            S12 = app.Slider_12.Value;
            S13 = app.Slider_13.Value;
            S14 = app.Slider_14.Value;
            S15 = app.Slider_15.Value;
            S16 = app.Slider_16.Value;
            S17 = app.Slider_17.Value;
            S18 = app.Slider_18.Value;
            S19 = app.Slider_19.Value;
            S20 = app.Slider_20.Value;
            S21 = app.Slider_21.Value;
            S22 = app.Slider_22.Value;
            S23 = app.Slider_23.Value;
            S24 = app.Slider_24.Value;
            S25 = app.Slider_25.Value;
            S26 = app.Slider_26.Value;
            S27 = app.Slider_27.Value;
            S28 = app.Slider_28.Value;
            S29 = app.Slider_29.Value;
            S30 = app.Slider_30.Value;
            S31 = app.Slider_31.Value;

            Mrplus = Mrp(app,Fs);

           b = num(app,target(app,S1,S2,S3,S4,S5,S6,S7,...
           S8,S9,S10,S11,S12,S13,S14,S15,S16,S17,S18,...
           S19,S20,S21,S22,S23,S24,S25,S26,S27,S28,...
           S29,S30,S31),Mrplus);

            dF = Fs/1024;
           f = -Fs/2:dF:Fs/2-dF;
```

```
           i = 0;
%Playback loop
while ~isDone(fileReader)
    %Acquiring the succeeding frame
    xk = fileReader();

    if length(xk(1,:))~=2
        xk = [xk,xk];
        xk1 = xk(:,1)';
        xk2 = xk(:,2)';
    else
    xk1 = xk(:,1)';
    xk2 = xk(:,2)';
    end

    if strcmp(app.PlayButton.Text,'Play') == 1
        %Pause loop
        while strcmp(app.PlayButton.Text,'Play') == 1 && ...
                app.isStop == 0
            pause(1);
        end
    end

    pause(0);

    %Checking if slider configuration changed
    if app.Slider_1.Value ~= S1 || app.Slider_2.Value ~= S2 ...
        ||...
            app.Slider_3.Value ~= S3 ||...
            S4 ~= app.Slider_4.Value ||...
            S5 ~= app.Slider_5.Value || ...
          app.Slider_6.Value ~= S6 || ...
          app.Slider_7.Value ~= S7 || ...
          app.Slider_8.Value ~= S8 || ...
          S9 ~= app.Slider_9.Value || ...
          S10 ~= app.Slider_10.Value || ...
          app.Slider_11.Value ~= S11 || ...
          app.Slider_12.Value ~= S12 || ...
          app.Slider_13.Value ~= S13 || ...
          S14 ~= app.Slider_14.Value || ...
          S15 ~= app.Slider_15.Value || ...
          app.Slider_16.Value ~= S16 || ...
          app.Slider_17.Value ~= S17 || ...
          app.Slider_18.Value ~= S18 || ...
          S19 ~= app.Slider_19.Value || ...
          S20 ~= app.Slider_20.Value || ...
          app.Slider_21.Value ~= S21 || ...
          app.Slider_22.Value ~= S22 || ...
          app.Slider_23.Value ~= S23 || ...
          S24 ~= app.Slider_24.Value || ...
          S25 ~= app.Slider_25.Value || ...
          app.Slider_26.Value ~= S26 || ...
          app.Slider_27.Value ~= S27 || ...
```

```
            app.Slider_28.Value ~= S28 || ...
            S29 ~= app.Slider_29.Value || ...
            S30 ~= app.Slider_30.Value || ...
            app.Slider_31.Value ~= S31

              S1 = app.Slider_1.Value;
              S2 = app.Slider_2.Value;
              S3 = app.Slider_3.Value;
              S4 = app.Slider_4.Value;
              S5 = app.Slider_5.Value;
              S6 = app.Slider_6.Value;
              S7 = app.Slider_7.Value;
              S8 = app.Slider_8.Value;
              S9 = app.Slider_9.Value;
              S10 = app.Slider_10.Value;
              S11 = app.Slider_11.Value;
              S12 = app.Slider_12.Value;
              S13 = app.Slider_13.Value;
              S14 = app.Slider_14.Value;
              S15 = app.Slider_15.Value;
              S16 = app.Slider_16.Value;
              S17 = app.Slider_17.Value;
              S18 = app.Slider_18.Value;
              S19 = app.Slider_19.Value;
              S20 = app.Slider_20.Value;
              S21 = app.Slider_21.Value;
              S22 = app.Slider_22.Value;
              S23 = app.Slider_23.Value;
              S24 = app.Slider_24.Value;
              S25 = app.Slider_25.Value;
              S26 = app.Slider_26.Value;
              S27 = app.Slider_27.Value;
              S28 = app.Slider_28.Value;
              S29 = app.Slider_29.Value;
              S30 = app.Slider_30.Value;
              S31 = app.Slider_31.Value;

          %Calculating the new filters
          Mrplus = Mrp(app,Fs);
          b = num(app,target(app,S1,S2,S3,S4,S5,S6,S7,S8,S9, ...
                 S10,S11,S12,S13,S14,S15,S16,S17,S18,S19,S20,...
                 S21,S22, ...
                     S23,S24,S25,S26,S27,S28,S29,S30,S31),...
                 Mrplus);


      end

      %Filtering process
         for n=1:63
             if n<63
                 ykNew1(n,:) = filterNew(app,b((2*n-1):(2*n)),...
                     xk1,ybuffer1(n,:),xbuffer1,Fs,n);
                 ykNew2(n,:) = filterNew(app,b((2*n-1):(2*n)),...
```

```
                    xk2,ybuffer2(n,:),xbuffer2,Fs,n);
            else
                ykNew1(n,:) = xk1*b(125);
                ykNew2(n,:) = xk2*b(125);
            end

        end

        yk1=0;
        yk2=0;
        for n=1:63
            yk1=yk1 + ykNew1(n,:);

            yk2=yk2 + ykNew2(n,:);
        end

    %Playback of frame
    deviceWriter([0.25*yk1',0.25*yk2']);

    %Delay updates
    xbuffer1 = flip(xk1(length(xk1)-1:length(xk1)));
    xbuffer2 = flip(xk2(length(xk2)-1:length(xk2)));
   for n=1:62

             ybuffer1(n,:)=flip(ykNew1(n,...
                 (length(ykNew1(n,:))-2):(length(ykNew1(n,:)))));

             ybuffer2(n,:)=flip(ykNew2(n,...
                 (length(ykNew2(n,:))-2):(length(ykNew2(n,:)))));

    end

    %FFT plot update if allowed
    if mod(i,51-round(app.Knob.Value))==0 && ...
            strcmp(app.Switch.Value,'On') == 1
    z = fftshift(fft(yk1));
    area(app.UIAxes,f,abs(z)/1024)
    drawnow limitrate;

    end
    if app.isStop == 1
                release(fileReader);
                release(deviceWriter);
                app.PlayButton.Text = 'Play';
                app.isPlay = 0;
                app.isStop = 0;
    end

    i = i+1;

end

release(fileReader);
release(deviceWriter);
```

```
app.PlayButton.Text = 'Play';
app.isPlay = 0;

            elseif strcmp(app.PlayButton.Text,'Play') && ...
                    app.isPlay == 1
                app.PlayButton.Text = 'Pause';
            elseif app.isStop == 1
                release(fileReader);
                release(deviceWriter);
                app.PlayButton.Text = 'Play';
                app.isPlay = 0;
                app.isStop = 0;

            else

                app.PlayButton.Text = 'Play';
            end

        end

 % Button pushed function: RecButton
        function Rec(app, event)
           if app.isPlay == 1
            uialert(app.UIFigure,...
                'Stop playback before recordning',...
                'Tip','Icon','warning');
           elseif app.isRec == 0
            %Initializing the microphone
            deviceReader = audioDeviceReader(44100,1024,...
                'NumChannels',2);
                deviceWriter = ...
                    audioDeviceWriter('SampleRate',...
                    deviceReader.SampleRate);
                app.Lamp.Enable = 'on';
                app.isRec = 1;
                app.RecButton.Text = 'Stop';

               %Initializing the delays
              xbuffer1 = 0;
              xbuffer2 = 0;
              ybuffer1 = zeros(62,3);
              ybuffer2 = zeros(62,3);





            S1 = app.Slider_1.Value;
            S2 = app.Slider_2.Value;
            S3 = app.Slider_3.Value;
            S4 = app.Slider_4.Value;
            S5 = app.Slider_5.Value;
            S6 = app.Slider_6.Value;
            S7 = app.Slider_7.Value;
```

```
            S8 = app.Slider_8.Value;
            S9 = app.Slider_9.Value;
            S10 = app.Slider_10.Value;
            S11 = app.Slider_11.Value;
            S12 = app.Slider_12.Value;
            S13 = app.Slider_13.Value;
            S14 = app.Slider_14.Value;
            S15 = app.Slider_15.Value;
            S16 = app.Slider_16.Value;
            S17 = app.Slider_17.Value;
            S18 = app.Slider_18.Value;
            S19 = app.Slider_19.Value;
            S20 = app.Slider_20.Value;
            S21 = app.Slider_21.Value;
            S22 = app.Slider_22.Value;
            S23 = app.Slider_23.Value;
            S24 = app.Slider_24.Value;
            S25 = app.Slider_25.Value;
            S26 = app.Slider_26.Value;
            S27 = app.Slider_27.Value;
            S28 = app.Slider_28.Value;
            S29 = app.Slider_29.Value;
            S30 = app.Slider_30.Value;
            S31 = app.Slider_31.Value;

            Fs=44100;
            app.fs = Fs;
            Mrplus = Mrp(app,Fs);

           b = num(app,target(app,S1,S2,S3,S4,...
           S5,S6,S7,S8,S9,S10,S11,S12,S13,S14,S15,...
               S16,S17,S18,S19,S20,S21,S22,S23,...
               S24,S25,S26,S27,S28,S29,S30,S31),...
               Mrplus);

           dF = Fs/1024;
           f = -Fs/2:dF:Fs/2-dF;

           i = 1;

%Record and playback loop
while app.isRec == 1

    %Acquiring the next frame
    xk = deviceReader();

    xk1 = xk(:,1)';
    xk2 = xk(:,2)';

    pause(0);

     %Checking if slider configuration changed
     if app.Slider_1.Value ~= S1 || app.Slider_2.Value ~= ...
        S2 ||...
```

```
            app.Slider_3.Value ~= S3 || S4 ~= ... app.Slider_4.Value ||...
            S5 ~= app.Slider_5.Value || app.Slider_6.Value ~= ... S6 ||...
            app.Slider_7.Value ~= S7 || app.Slider_8.Value ~= ... S8 ||...
            S9 ~= app.Slider_9.Value || S10 ~= ... app.Slider_10.Value || ...
            app.Slider_11.Value ~= S11 ||...
            app.Slider_12.Value ~= S12 ||...
            app.Slider_13.Value ~= S13 ||...
            S14 ~= app.Slider_14.Value ||...
            S15 ~= app.Slider_15.Value ||...
            app.Slider_16.Value ~= S16 ||...
            app.Slider_17.Value ~= S17 ||...
            app.Slider_18.Value ~= S18 ||...
            S19 ~= app.Slider_19.Value ||...
            S20 ~= app.Slider_20.Value ||...
            app.Slider_21.Value ~= S21 ||...
            app.Slider_22.Value ~= S22 ||...
            app.Slider_23.Value ~= S23 ||...
            S24 ~= app.Slider_24.Value ||...
            S25 ~= app.Slider_25.Value ||...
            app.Slider_26.Value ~= S26 ||...
            app.Slider_27.Value ~= S27 ||...
            app.Slider_28.Value ~= S28 ||...
            S29 ~= app.Slider_29.Value ||...
            S30 ~= app.Slider_30.Value ||...
            app.Slider_31.Value ~= S31

              S1 = app.Slider_1.Value;
              S2 = app.Slider_2.Value;
              S3 = app.Slider_3.Value;
              S4 = app.Slider_4.Value;
              S5 = app.Slider_5.Value;
              S6 = app.Slider_6.Value;
              S7 = app.Slider_7.Value;
              S8 = app.Slider_8.Value;
              S9 = app.Slider_9.Value;
              S10 = app.Slider_10.Value;
              S11 = app.Slider_11.Value;
              S12 = app.Slider_12.Value;
              S13 = app.Slider_13.Value;
              S14 = app.Slider_14.Value;
              S15 = app.Slider_15.Value;
              S16 = app.Slider_16.Value;
              S17 = app.Slider_17.Value;
              S18 = app.Slider_18.Value;
              S19 = app.Slider_19.Value;
              S20 = app.Slider_20.Value;
              S21 = app.Slider_21.Value;
              S22 = app.Slider_22.Value;
              S23 = app.Slider_23.Value;
              S24 = app.Slider_24.Value;
```

```
                S25 = app.Slider_25.Value;
                S26 = app.Slider_26.Value;
                S27 = app.Slider_27.Value;
                S28 = app.Slider_28.Value;
                S29 = app.Slider_29.Value;
                S30 = app.Slider_30.Value;
                S31 = app.Slider_31.Value;

            %Calculating the new filters
            Mrplus = Mrp(app,Fs);
            b = num(app,target(app,S1,S2,S3,S4,S5,...
            S6,S7,S8,S9,S10,S11,S12,S13,S14,S15,...
            S16,S17,S18,S19,S20,S21,S22,...
            S23,S24,S25,S26,S27,S28,S29,S30,S31),...
            Mrplus);

        end

            %Filtering process
           for n=1:63
               if n<63
                   ykNew1(n,:) = filterNew(app,...
                       b((2*n-1):(2*n)),xk1,ybuffer1(n,:),...
                       xbuffer1,Fs,n);
                   ykNew2(n,:) = filterNew(app,...
                       b((2*n-1):(2*n)),xk2,ybuffer2(n,:),...
                       xbuffer2,Fs,n);
               else
                   ykNew1(n,:) = xk1*b(125);
                   ykNew2(n,:) = xk2*b(125);

               end

           end

           yk1=0;
           yk2=0;
           for n=1:63
               yk1=yk1 + ykNew1(n,:);

               yk2=yk2 + ykNew2(n,:);
           end

        %Playback of frame
        deviceWriter([0.25*yk1',0.25*yk2']);

        %Updating the delays
        xbuffer1 = flip(xk1(length(xk1)-1:length(xk1)));
        xbuffer2 = flip(xk2(length(xk2)-1:length(xk2)));
       for n=1:62
                ybuffer1(n,:)=flip(ykNew1(n,...
                    (length(ykNew1(n,:))-2):(length(ykNew1(n,:)))));
                ybuffer2(n,:)=flip(ykNew2(n,...
                    (length(ykNew2(n,:))-2):(length(ykNew2(n,:)))));
```

```
    end

    %FFT plot update if allowed
    if mod(i,51-round(app.Knob.Value))==0 && ...
            strcmp(app.Switch.Value,'On') == 1
    z = fftshift(fft(yk1));
    area(app.UIAxes,f,abs(z)/1024)
    drawnow limitrate;
    end

    %Blinking of the lamp
    if  mod(i,15) == 0
        if strcmp(app.Lamp.Enable,'on') == 1
        app.Lamp.Enable = 'off';
        else
        app.Lamp.Enable = 'on';
        end
    end

    i = i+1;

    end
release(deviceReader);
release(deviceWriter);
         else
            app.isRec = 0;
            app.RecButton.Text = 'Rec';
            end

        end

        % Button pushed function: StopButton
        function Stop(app, event)
            app.isStop = 1;
        end

        % Value changed function: DropDown
        %Setting the slider presets
        function Preset(app, event)
            value = app.DropDown.Value;
            if strcmp('Rock',value) == 1
            app.Slider_1.Value = 7;
            app.Slider_2.Value = 7;app.Slider_3.Value = 6;
            app.Slider_4.Value = 6;app.Slider_5.Value = 5;
            app.Slider_6.Value = 5;app.Slider_7.Value = 5;
            app.Slider_8.Value = 3;app.Slider_9.Value = 2;
            app.Slider_10.Value = 1;app.Slider_11.Value = 0;
            app.Slider_12.Value = 0;app.Slider_13.Value = -1;
            app.Slider_14.Value = -1;app.Slider_15.Value = -2;
            app.Slider_16.Value = -3;app.Slider_17.Value = -3;
            app.Slider_18.Value = -4;app.Slider_19.Value = -3;
            app.Slider_20.Value = -3;app.Slider_21.Value = -2;
            app.Slider_22.Value = -1;app.Slider_23.Value = 0;
            app.Slider_24.Value = 2;app.Slider_25.Value = 3;
```

```
            app.Slider_26.Value = 4;app.Slider_27.Value = 5;
            app.Slider_28.Value = 5;app.Slider_29.Value = 5;
            app.Slider_30.Value = 5;app.Slider_31.Value = 4;

            elseif strcmp('Flat',value) == 1
                app.Slider_1.Value = 0;
            app.Slider_2.Value = 0;app.Slider_3.Value = 0;
            app.Slider_4.Value = 0;app.Slider_5.Value = 0;
            app.Slider_6.Value = 0;app.Slider_7.Value = 0;
            app.Slider_8.Value = 0;app.Slider_9.Value = 0;
            app.Slider_10.Value = 0;app.Slider_11.Value = 0;
            app.Slider_12.Value = 0;app.Slider_13.Value = 0;
            app.Slider_14.Value = 0;app.Slider_15.Value = 0;
            app.Slider_16.Value = 0;app.Slider_17.Value = 0;
            app.Slider_18.Value = 0;app.Slider_19.Value = 0;
            app.Slider_20.Value = 0;app.Slider_21.Value = 0;
            app.Slider_22.Value = 0;app.Slider_23.Value = 0;
            app.Slider_24.Value = 0;app.Slider_25.Value = 0;
            app.Slider_26.Value = 0;app.Slider_27.Value = 0;
            app.Slider_28.Value = 0;app.Slider_29.Value = 0;
            app.Slider_30.Value = 0;app.Slider_31.Value = 0;

            elseif strcmp('Pop',value) == 1
                app.Slider_1.Value = 0;
            app.Slider_2.Value = 0;app.Slider_3.Value = 0;
            app.Slider_4.Value = 0;app.Slider_5.Value = 1;
            app.Slider_6.Value = 1;app.Slider_7.Value = 2;
            app.Slider_8.Value = 3;app.Slider_9.Value = 4;
            app.Slider_10.Value = 4;app.Slider_11.Value = 4;
            app.Slider_12.Value = 3;app.Slider_13.Value = 2;
            app.Slider_14.Value = 1;app.Slider_15.Value = 0;
            app.Slider_16.Value = 0;app.Slider_17.Value = 0;
            app.Slider_18.Value = -1;app.Slider_19.Value = -1;
            app.Slider_20.Value = -2;app.Slider_21.Value = -2;
            app.Slider_22.Value = -2;app.Slider_23.Value = -2;
            app.Slider_24.Value = -1;app.Slider_25.Value = -1;
            app.Slider_26.Value = 0;app.Slider_27.Value = 1;
            app.Slider_28.Value = 0;app.Slider_29.Value = -1;
            app.Slider_30.Value = -1;app.Slider_31.Value = -1;

            elseif strcmp('Bass',value) == 1
                app.Slider_1.Value = 12;
            app.Slider_2.Value = 12;app.Slider_3.Value = 12;
            app.Slider_4.Value = 12;app.Slider_5.Value = 12;
            app.Slider_6.Value = 12;app.Slider_7.Value = 12;
            app.Slider_8.Value = 12;app.Slider_9.Value = 11;
            app.Slider_10.Value = 10;app.Slider_11.Value = 9;
            app.Slider_12.Value = 8;app.Slider_13.Value = 7;
            app.Slider_14.Value = 6;app.Slider_15.Value = 5;
            app.Slider_16.Value = 4;app.Slider_17.Value = 3;
            app.Slider_18.Value = 2;app.Slider_19.Value = 1;
            app.Slider_20.Value = 0;app.Slider_21.Value = -1;
            app.Slider_22.Value = -2;app.Slider_23.Value = -3;
            app.Slider_24.Value = -4;app.Slider_25.Value = -5;
```

```
        app.Slider_26.Value = -6;app.Slider_27.Value = -7;
        app.Slider_28.Value = -8;app.Slider_29.Value = -9;
        app.Slider_30.Value = -10;app.Slider_31.Value = ...
            -11;

        elseif strcmp('Treble',value) == 1
            app.Slider_1.Value = -12;
        app.Slider_2.Value = -11;app.Slider_3.Value = -10;
        app.Slider_4.Value = -10;app.Slider_5.Value = -10;
        app.Slider_6.Value = -10;app.Slider_7.Value = -9;
        app.Slider_8.Value = -8;app.Slider_9.Value = -7;
        app.Slider_10.Value = -7;app.Slider_11.Value = -7;
        app.Slider_12.Value = -6;app.Slider_13.Value = -6;
        app.Slider_14.Value = -5;app.Slider_15.Value = -5;
        app.Slider_16.Value = -3;app.Slider_17.Value = -2;
        app.Slider_18.Value = 0;app.Slider_19.Value = 3;
        app.Slider_20.Value = 5;app.Slider_21.Value = 6;
        app.Slider_22.Value = 8;app.Slider_23.Value = 9;
        app.Slider_24.Value = 9;app.Slider_25.Value = 10;
        app.Slider_26.Value = 11;app.Slider_27.Value = 12;
        app.Slider_28.Value = 12;app.Slider_29.Value = 12;
        app.Slider_30.Value = 12;app.Slider_31.Value = 12;

        elseif strcmp('Vocal',value) == 1
            app.Slider_1.Value = -8;
        app.Slider_2.Value = -7;app.Slider_3.Value = -6;
        app.Slider_4.Value = -5;app.Slider_5.Value = -5;
        app.Slider_6.Value = -4;app.Slider_7.Value = -3;
        app.Slider_8.Value = -3;app.Slider_9.Value = 0;
        app.Slider_10.Value = 0;app.Slider_11.Value = 2;
        app.Slider_12.Value = 5;app.Slider_13.Value = 6;
        app.Slider_14.Value = 7;app.Slider_15.Value = 7;
        app.Slider_16.Value = 7;app.Slider_17.Value = 7;
        app.Slider_18.Value = 7;app.Slider_19.Value = 7;
        app.Slider_20.Value = 6;app.Slider_21.Value = 4;
        app.Slider_22.Value = 3;app.Slider_23.Value = 2;
        app.Slider_24.Value = 0;app.Slider_25.Value = -2;
        app.Slider_26.Value = -5;app.Slider_27.Value = -5;
        app.Slider_28.Value = -5;app.Slider_29.Value = -5;
        app.Slider_30.Value = -5;app.Slider_31.Value = -7;

        elseif strcmp('Classical',value) == 1
            app.Slider_1.Value = -3;
        app.Slider_2.Value = -2;app.Slider_3.Value = -1;
        app.Slider_4.Value = 0;app.Slider_5.Value = 1;
        app.Slider_6.Value = 1;app.Slider_7.Value = 1;
        app.Slider_8.Value = 2;app.Slider_9.Value = 2;
        app.Slider_10.Value = 3;app.Slider_11.Value = 3;
        app.Slider_12.Value = 2;app.Slider_13.Value = 1;
        app.Slider_14.Value = 0;app.Slider_15.Value = -1;
        app.Slider_16.Value = -2;app.Slider_17.Value = -4;
        app.Slider_18.Value = -6;app.Slider_19.Value = -8;
        app.Slider_20.Value = -8;app.Slider_21.Value = -5;
        app.Slider_22.Value = -3;app.Slider_23.Value = -1;
```

```
        app.Slider_24.Value = 0;app.Slider_25.Value = 1;
        app.Slider_26.Value = 2;app.Slider_27.Value = 4;
        app.Slider_28.Value = 3;app.Slider_29.Value = 1;
        app.Slider_30.Value = 0;app.Slider_31.Value = -1;

        elseif strcmp('Hip-Hop',value) == 1
            app.Slider_1.Value = 4;
        app.Slider_2.Value = 4;app.Slider_3.Value = 4;
        app.Slider_4.Value = 4;app.Slider_5.Value = 3;
        app.Slider_6.Value = 3;app.Slider_7.Value = 2;
        app.Slider_8.Value = 1;app.Slider_9.Value = 1;
        app.Slider_10.Value = 1;app.Slider_11.Value = 0;
        app.Slider_12.Value = 0;app.Slider_13.Value = -1;
        app.Slider_14.Value = -1;app.Slider_15.Value = -2;
        app.Slider_16.Value = -2;app.Slider_17.Value = -3;
        app.Slider_18.Value = -3;app.Slider_19.Value = -2;
        app.Slider_20.Value = -2;app.Slider_21.Value = -1;
        app.Slider_22.Value = 0;app.Slider_23.Value = 1;
        app.Slider_24.Value = 1;app.Slider_25.Value = 2;
        app.Slider_26.Value = 3;app.Slider_27.Value = 3;
        app.Slider_28.Value = 3;app.Slider_29.Value = 4;
        app.Slider_30.Value = 3;app.Slider_31.Value = 2;

        elseif strcmp('Dance',value) == 1
            app.Slider_1.Value = 6;
        app.Slider_2.Value = 6;app.Slider_3.Value = 6;
        app.Slider_4.Value = 6;app.Slider_5.Value = 6;
        app.Slider_6.Value = 7;app.Slider_7.Value = 6;
        app.Slider_8.Value = 4;app.Slider_9.Value = 2;
        app.Slider_10.Value = 1;app.Slider_11.Value = 1;
        app.Slider_12.Value = 0;app.Slider_13.Value = -1;
        app.Slider_14.Value = -2;app.Slider_15.Value = -2;
        app.Slider_16.Value = -1;app.Slider_17.Value = -1;
        app.Slider_18.Value = 0;app.Slider_19.Value = 1;
        app.Slider_20.Value = 1;app.Slider_21.Value = 2;
        app.Slider_22.Value = 3;app.Slider_23.Value = 2;
        app.Slider_24.Value = 2;app.Slider_25.Value = 1;
        app.Slider_26.Value = 0;app.Slider_27.Value = -1;
        app.Slider_28.Value = -1;app.Slider_29.Value = -2;
        app.Slider_30.Value = -2;app.Slider_31.Value = -2;

        elseif strcmp('Jazz',value) == 1
            app.Slider_1.Value = -3;
        app.Slider_2.Value = -2;app.Slider_3.Value = -2;
        app.Slider_4.Value = 0;app.Slider_5.Value = 2;
        app.Slider_6.Value = 3;app.Slider_7.Value = 1;
        app.Slider_8.Value = -2;app.Slider_9.Value = -6;
        app.Slider_10.Value = -3;app.Slider_11.Value = -1;
        app.Slider_12.Value = 0;app.Slider_13.Value = 1;
        app.Slider_14.Value = 3;app.Slider_15.Value = 6;
        app.Slider_16.Value = 5;app.Slider_17.Value = 4;
        app.Slider_18.Value = 3;app.Slider_19.Value = 2;
        app.Slider_20.Value = 2;app.Slider_21.Value = 1;
        app.Slider_22.Value = 1;app.Slider_23.Value = 0;
```

```
        app.Slider_24.Value = 0;app.Slider_25.Value = 0;
        app.Slider_26.Value = 0;app.Slider_27.Value = 0;
        app.Slider_28.Value = 0;app.Slider_29.Value = -1;
        app.Slider_30.Value = -1;app.Slider_31.Value = -1;

        elseif strcmp('Powerfull',value) == 1
            app.Slider_1.Value = 9;
        app.Slider_2.Value = 8;app.Slider_3.Value = 8;
        app.Slider_4.Value = 8;app.Slider_5.Value = 8;
        app.Slider_6.Value = 7;app.Slider_7.Value = 7;
        app.Slider_8.Value = 6;app.Slider_9.Value = 4;
        app.Slider_10.Value = 1;app.Slider_11.Value = -1;
        app.Slider_12.Value = -2;app.Slider_13.Value = -3;
        app.Slider_14.Value = -4;app.Slider_15.Value = -4;
        app.Slider_16.Value = -4;app.Slider_17.Value = -4;
        app.Slider_18.Value = -4;app.Slider_19.Value = -3;
        app.Slider_20.Value = -2;app.Slider_21.Value = 0;
        app.Slider_22.Value = 1;app.Slider_23.Value = 3;
        app.Slider_24.Value = 5;app.Slider_25.Value = 6;
        app.Slider_26.Value = 8;app.Slider_27.Value = 8;
        app.Slider_28.Value = 8;app.Slider_29.Value = 8;
        app.Slider_30.Value = 8;app.Slider_31.Value = 8;

        elseif strcmp('Shitty music',value) == 1
            app.Slider_1.Value = -13;
        app.Slider_2.Value = -13;app.Slider_3.Value = -13;
        app.Slider_4.Value = -13;app.Slider_5.Value = -13;
        app.Slider_6.Value = -13;app.Slider_7.Value = -13;
        app.Slider_8.Value = -13;app.Slider_9.Value = -13;
        app.Slider_10.Value = -13;app.Slider_11.Value = ...
            -13;
        app.Slider_12.Value = -13;app.Slider_13.Value = ...
            -13;
        app.Slider_14.Value = -13;app.Slider_15.Value = ...
            -13;
        app.Slider_16.Value = -13;app.Slider_17.Value = ...
            -13;
        app.Slider_18.Value = -13;app.Slider_19.Value = ...
            -13;
        app.Slider_20.Value = -13;app.Slider_21.Value = ...
            -13;
        app.Slider_22.Value = -13;app.Slider_23.Value = ...
            -13;
        app.Slider_24.Value = -13;app.Slider_25.Value = ...
            -13;
        app.Slider_26.Value = -13;app.Slider_27.Value = ...
            -13;
        app.Slider_28.Value = -13;app.Slider_29.Value = ...
            -13;
        app.Slider_30.Value = -13;app.Slider_31.Value = ...
            -13;

        elseif strcmp('MUU',value) == 1
            app.Slider_1.Value = -12;
```

```
            app.Slider_2.Value = -3;app.Slider_3.Value = 4;
            app.Slider_4.Value = 12;app.Slider_5.Value = 6;
            app.Slider_6.Value = 3;app.Slider_7.Value = 6;
            app.Slider_8.Value = 12;app.Slider_9.Value = 4;
            app.Slider_10.Value = -3;app.Slider_11.Value = -12;
            app.Slider_12.Value = 12;app.Slider_13.Value = 3;
            app.Slider_14.Value = -5;app.Slider_15.Value = -9;
            app.Slider_16.Value = -11;app.Slider_17.Value = ...
                -11;
            app.Slider_18.Value = -9;app.Slider_19.Value = -5;
            app.Slider_20.Value = 3;app.Slider_21.Value = 12;
            app.Slider_22.Value = 12;app.Slider_23.Value = 3;
            app.Slider_24.Value = -5;app.Slider_25.Value = -9;
            app.Slider_26.Value = -11;app.Slider_27.Value = ...
                -11;
            app.Slider_28.Value = -9;app.Slider_29.Value = -5;
            app.Slider_30.Value = 3;app.Slider_31.Value = 12;

            end

        end

          % Value changing function: Slider_1
        function Custom(app, event)
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_2
        function Custom2(app, event)
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_3
        function Custom3(app, event)
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_4
        function Custom4(app, event)
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_5
        function Custom5(app, event)
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_6
        function Custom6(app, event)
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_7
        function Custom7(app, event)
```

```
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_8
        function Custom8(app, event)
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_9
        function Custom9(app, event)
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_10
        function Custom10(app, event)
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_11
        function Custom11(app, event)
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_12
        function Custom12(app, event)
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_13
        function Custom13(app, event)
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_14
        function Custom14(app, event)
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_15
        function Custom15(app, event)
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_16
        function Custom16(app, event)
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_17
        function Custom17(app, event)
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_18
```

```
        function Custom18(app, event)
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_19
        function Custom19(app, event)
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_20
        function Custom20(app, event)
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_21
        function Custom21(app, event)
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_22
        function Custom22(app, event)
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_23
        function Custom23(app, event)
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_24
        function Custom24(app, event)
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_25
        function Custom25(app, event)
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_26
        function Custom26(app, event)
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_27
        function Custom27(app, event)
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_28
        function Custom28(app, event)
            app.DropDown.Value = 'Custom';
        end

```

```
        % Value changing function: Slider_29
        function Custom29(app, event)
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_30
        function Custom30(app, event)
            app.DropDown.Value = 'Custom';
        end

        % Value changing function: Slider_31
        function Custom31(app, event)
            app.DropDown.Value = 'Custom';
        end
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure and hide until all components ...
                are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Color = [0.8 0.8 0.8];
            app.UIFigure.Position = [150 80 990 600];
            app.UIFigure.Name = 'UI Figure';

            % Create Image
            app.Image = uiimage(app.UIFigure);
            app.Image.Position = [0 0 990 600];
            app.Image.ImageSource = 'GUI.png';

            % Create Slider_1
            app.Slider_1 = uislider(app.UIFigure);
            app.Slider_1.Limits = [-13 13];
            app.Slider_1.MajorTicks = [];
            app.Slider_1.MajorTickLabels = {''};
            app.Slider_1.Orientation = 'vertical';
            app.Slider_1.ValueChangingFcn = ...
                createCallbackFcn(app, @Custom, true);
            app.Slider_1.MinorTicks = [];
            app.Slider_1.Position = [59 69 3 112];

            % Create Slider_2
            app.Slider_2 = uislider(app.UIFigure);
            app.Slider_2.Limits = [-13 13];
            app.Slider_2.MajorTicks = [];
            app.Slider_2.MajorTickLabels = {''};
            app.Slider_2.Orientation = 'vertical';
            app.Slider_2.ValueChangingFcn = ...
                createCallbackFcn(app, @Custom2, true);
            app.Slider_2.MinorTicks = [];
```

```
            app.Slider_2.Position = [89 69 3 113];

            % Create Slider_3
            app.Slider_3 = uislider(app.UIFigure);
            app.Slider_3.Limits = [-13 13];
            app.Slider_3.MajorTicks = [];
            app.Slider_3.MajorTickLabels = {''};
            app.Slider_3.Orientation = 'vertical';
            app.Slider_3.ValueChangingFcn = ...
                createCallbackFcn(app, @Custom3, true);
            app.Slider_3.MinorTicks = [];
            app.Slider_3.Position = [118 69 3 113];

            % Create Slider_4
            app.Slider_4 = uislider(app.UIFigure);
            app.Slider_4.Limits = [-13 13];
            app.Slider_4.MajorTicks = [];
            app.Slider_4.MajorTickLabels = {''};
            app.Slider_4.Orientation = 'vertical';
            app.Slider_4.ValueChangingFcn = ...
                createCallbackFcn(app, @Custom4, true);
            app.Slider_4.MinorTicks = [];
            app.Slider_4.Position = [146 69 3 113];

            % Create Slider_5
            app.Slider_5 = uislider(app.UIFigure);
            app.Slider_5.Limits = [-13 13];
            app.Slider_5.MajorTicks = [];
            app.Slider_5.MajorTickLabels = {''};
            app.Slider_5.Orientation = 'vertical';
            app.Slider_5.ValueChangingFcn = ...
                createCallbackFcn(app, @Custom5, true);
            app.Slider_5.MinorTicks = [];
            app.Slider_5.Position = [174 69 3 113];

            % Create Slider_6
            app.Slider_6 = uislider(app.UIFigure);
            app.Slider_6.Limits = [-13 13];
            app.Slider_6.MajorTicks = [];
            app.Slider_6.MajorTickLabels = {'', ' ', ' ', ' ...
                ', ' ', ' ', ' ', ' ', ' '};
            app.Slider_6.Orientation = 'vertical';
            app.Slider_6.ValueChangingFcn = ...
                createCallbackFcn(app, @Custom6, true);
            app.Slider_6.MinorTicks = [];
            app.Slider_6.Position = [204 69 3 113];

            % Create Slider_7
            app.Slider_7 = uislider(app.UIFigure);
            app.Slider_7.Limits = [-13 13];
            app.Slider_7.MajorTicks = [];
            app.Slider_7.MajorTickLabels = {'', ' ', ' ', ' ...
                ', ' ', ' ', ' ', ' ', ' '};
            app.Slider_7.Orientation = 'vertical';
```

```
        app.Slider_7.ValueChangingFcn = ...
            createCallbackFcn(app, @Custom7, true);
        app.Slider_7.MinorTicks = [];
        app.Slider_7.Position = [233 69 3 113];

        % Create Slider_8
        app.Slider_8 = uislider(app.UIFigure);
        app.Slider_8.Limits = [-13 13];
        app.Slider_8.MajorTicks = [];
        app.Slider_8.MajorTickLabels = {'', ' ', ' ', ' ...
            ', ' ', ' ', ' ', ' ', ' '};
        app.Slider_8.Orientation = 'vertical';
        app.Slider_8.ValueChangingFcn = ...
            createCallbackFcn(app, @Custom8, true);
        app.Slider_8.MinorTicks = [];
        app.Slider_8.Position = [262 69 3 113];

        % Create Slider_9
        app.Slider_9 = uislider(app.UIFigure);
        app.Slider_9.Limits = [-13 13];
        app.Slider_9.MajorTicks = [];
        app.Slider_9.MajorTickLabels = {'', ' ', ' ', ' ...
            ', ' ', ' ', ' ', ' ', ' '};
        app.Slider_9.Orientation = 'vertical';
        app.Slider_9.ValueChangingFcn = ...
            createCallbackFcn(app, @Custom9, true);
        app.Slider_9.MinorTicks = [];
        app.Slider_9.Position = [291 69 3 113];

        % Create Slider_10
        app.Slider_10 = uislider(app.UIFigure);
        app.Slider_10.Limits = [-13 13];
        app.Slider_10.MajorTicks = [];
        app.Slider_10.MajorTickLabels = {'', ' ', ' ', ...
            ' ', ' ', ' ', ' ', ' ', ' '};
        app.Slider_10.Orientation = 'vertical';
        app.Slider_10.ValueChangingFcn = ...
            createCallbackFcn(app, @Custom10, true);
        app.Slider_10.MinorTicks = [];
        app.Slider_10.Position = [321 69 3 113];

        % Create Slider_11
        app.Slider_11 = uislider(app.UIFigure);
        app.Slider_11.Limits = [-13 13];
        app.Slider_11.MajorTicks = [];
        app.Slider_11.MajorTickLabels = {'', ' ', ' ', ...
            ' ', ' ', ' ', ' ', ' ', ' '};
        app.Slider_11.Orientation = 'vertical';
        app.Slider_11.ValueChangingFcn = ...
            createCallbackFcn(app, @Custom11, true);
        app.Slider_11.MinorTicks = [];
        app.Slider_11.Position = [350 69 3 113];

        % Create Slider_12
```

```
            app.Slider_12 = uislider(app.UIFigure);
            app.Slider_12.Limits = [-13 13];
            app.Slider_12.MajorTicks = [];
            app.Slider_12.MajorTickLabels = {'', ' ', ' ', ...
                ' ', ' ', ' ', ' ', ' ', ' '};
            app.Slider_12.Orientation = 'vertical';
            app.Slider_12.ValueChangingFcn = ...
                createCallbackFcn(app, @Custom12, true);
            app.Slider_12.MinorTicks = [];
            app.Slider_12.Position = [378 69 3 113];

            % Create Slider_13
            app.Slider_13 = uislider(app.UIFigure);
            app.Slider_13.Limits = [-13 13];
            app.Slider_13.MajorTicks = [];
            app.Slider_13.MajorTickLabels = {'', ' ', ' ', ...
                ' ', ' ', ' ', ' ', ' ', ' '};
            app.Slider_13.Orientation = 'vertical';
            app.Slider_13.ValueChangingFcn = ...
                createCallbackFcn(app, @Custom13, true);
            app.Slider_13.MinorTicks = [];
            app.Slider_13.Position = [406 69 3 113];

            % Create Slider_14
            app.Slider_14 = uislider(app.UIFigure);
            app.Slider_14.Limits = [-13 13];
            app.Slider_14.MajorTicks = [];
            app.Slider_14.MajorTickLabels = {'', ' ', ' ', ...
                ' ', ' ', ' ', ' ', ' ', ' '};
            app.Slider_14.Orientation = 'vertical';
            app.Slider_14.ValueChangingFcn = ...
                createCallbackFcn(app, @Custom14, true);
            app.Slider_14.MinorTicks = [];
            app.Slider_14.Position = [436 69 3 113];

            % Create Slider_15
            app.Slider_15 = uislider(app.UIFigure);
            app.Slider_15.Limits = [-13 13];
            app.Slider_15.MajorTicks = [];
            app.Slider_15.MajorTickLabels = {'', ' ', ' ', ...
                ' ', ' ', ' ', ' ', ' ', ' '};
            app.Slider_15.Orientation = 'vertical';
            app.Slider_15.ValueChangingFcn = ...
                createCallbackFcn(app, @Custom15, true);
            app.Slider_15.MinorTicks = [];
            app.Slider_15.Position = [464 69 3 113];

            % Create Slider_16
            app.Slider_16 = uislider(app.UIFigure);
            app.Slider_16.Limits = [-13 13];
            app.Slider_16.MajorTicks = [];
            app.Slider_16.MajorTickLabels = {'', ' ', ' ', ...
                ' ', ' ', ' ', ' ', ' ', ' '};
            app.Slider_16.Orientation = 'vertical';
```

```
            app.Slider_16.ValueChangingFcn = ...
                createCallbackFcn(app, @Custom16, true);
            app.Slider_16.MinorTicks = [];
            app.Slider_16.Position = [493 69 3 113];

            % Create Slider_17
            app.Slider_17 = uislider(app.UIFigure);
            app.Slider_17.Limits = [-13 13];
            app.Slider_17.MajorTicks = [];
            app.Slider_17.MajorTickLabels = {'', ' ', ' ', ...
                ' ', ' ', ' ', ' ', ' ', ' '};
            app.Slider_17.Orientation = 'vertical';
            app.Slider_17.ValueChangingFcn = ...
                createCallbackFcn(app, @Custom17, true);
            app.Slider_17.MinorTicks = [];
            app.Slider_17.Position = [522 69 3 113];

            % Create Slider_18
            app.Slider_18 = uislider(app.UIFigure);
            app.Slider_18.Limits = [-13 13];
            app.Slider_18.MajorTicks = [];
            app.Slider_18.MajorTickLabels = {'', ' ', ' ', ...
                ' ', ' ', ' ', ' ', ' ', ' '};
            app.Slider_18.Orientation = 'vertical';
            app.Slider_18.ValueChangingFcn = ...
                createCallbackFcn(app, @Custom18, true);
            app.Slider_18.MinorTicks = [];
            app.Slider_18.Position = [551 69 3 113];

            % Create Slider_19
            app.Slider_19 = uislider(app.UIFigure);
            app.Slider_19.Limits = [-13 13];
            app.Slider_19.MajorTicks = [];
            app.Slider_19.MajorTickLabels = {'', ' ', ' ', ...
                ' ', ' ', ' ', ' ', ' ', ' '};
            app.Slider_19.Orientation = 'vertical';
            app.Slider_19.ValueChangingFcn = ...
                createCallbackFcn(app, @Custom19, true);
            app.Slider_19.MinorTicks = [];
            app.Slider_19.Position = [579 69 3 113];

            % Create Slider_20
            app.Slider_20 = uislider(app.UIFigure);
            app.Slider_20.Limits = [-13 13];
            app.Slider_20.MajorTicks = [];
            app.Slider_20.MajorTickLabels = {'', ' ', ' ', ...
                ' ', ' ', ' ', ' ', ' ', ' '};
            app.Slider_20.Orientation = 'vertical';
            app.Slider_20.ValueChangingFcn = ...
                createCallbackFcn(app, @Custom20, true);
            app.Slider_20.MinorTicks = [];
            app.Slider_20.FontWeight = 'bold';
            app.Slider_20.Position = [609 69 3 113];
```

```
            % Create Slider_21
            app.Slider_21 = uislider(app.UIFigure);
            app.Slider_21.Limits = [-13 13];
            app.Slider_21.MajorTicks = [];
            app.Slider_21.MajorTickLabels = {'', ' ', ' ', ...
                ' ', ' ', ' ', ' ', ' ', ' '};
            app.Slider_21.Orientation = 'vertical';
            app.Slider_21.ValueChangingFcn = ...
                createCallbackFcn(app, @Custom21, true);
            app.Slider_21.MinorTicks = [];
            app.Slider_21.Position = [637 69 3 113];

            % Create Slider_22
            app.Slider_22 = uislider(app.UIFigure);
            app.Slider_22.Limits = [-13 13];
            app.Slider_22.MajorTicks = [];
            app.Slider_22.MajorTickLabels = {'', ' ', ' ', ...
                ' ', ' ', ' ', ' ', ' ', ' '};
            app.Slider_22.Orientation = 'vertical';
            app.Slider_22.ValueChangingFcn = ...
                createCallbackFcn(app, @Custom22, true);
            app.Slider_22.MinorTicks = [];
            app.Slider_22.Position = [667 69 3 113];

            % Create Slider_23
            app.Slider_23 = uislider(app.UIFigure);
            app.Slider_23.Limits = [-13 13];
            app.Slider_23.MajorTicks = [];
            app.Slider_23.MajorTickLabels = {'', ' ', ' ', ...
                ' ', ' ', ' ', ' ', ' ', ' '};
            app.Slider_23.Orientation = 'vertical';
            app.Slider_23.ValueChangingFcn = ...
                createCallbackFcn(app, @Custom23, true);
            app.Slider_23.MinorTicks = [];
            app.Slider_23.Position = [695 69 3 113];

            % Create Slider_24
            app.Slider_24 = uislider(app.UIFigure);
            app.Slider_24.Limits = [-13 13];
            app.Slider_24.MajorTicks = [];
            app.Slider_24.MajorTickLabels = {'', ' ', ' ', ...
                ' ', ' ', ' ', ' ', ' ', ' '};
            app.Slider_24.Orientation = 'vertical';
            app.Slider_24.ValueChangingFcn = ...
                createCallbackFcn(app, @Custom24, true);
            app.Slider_24.MinorTicks = [];
            app.Slider_24.Position = [725 69 3 113];

            % Create Slider_25
            app.Slider_25 = uislider(app.UIFigure);
            app.Slider_25.Limits = [-13 13];
            app.Slider_25.MajorTicks = [];
            app.Slider_25.MajorTickLabels = {'', ' ', ' ', ...
                ' ', ' ', ' ', ' ', ' ', ' '};
```

```
            app.Slider_25.Orientation = 'vertical';
            app.Slider_25.ValueChangingFcn = ...
                createCallbackFcn(app, @Custom25, true);
            app.Slider_25.MinorTicks = [];
            app.Slider_25.Position = [753 69 3 113];

            % Create Slider_26
            app.Slider_26 = uislider(app.UIFigure);
            app.Slider_26.Limits = [-13 13];
            app.Slider_26.MajorTicks = [];
            app.Slider_26.MajorTickLabels = {'', ' ', ' ', ...
                ' ', ' ', ' ', ' ', ' ', ' '};
            app.Slider_26.Orientation = 'vertical';
            app.Slider_26.ValueChangingFcn = ...
                createCallbackFcn(app, @Custom26, true);
            app.Slider_26.MinorTicks = [];
            app.Slider_26.Position = [782 69 3 113];

            % Create Slider_27
            app.Slider_27 = uislider(app.UIFigure);
            app.Slider_27.Limits = [-13 13];
            app.Slider_27.MajorTicks = [];
            app.Slider_27.MajorTickLabels = {'', ' ', ' ', ...
                ' ', ' ', ' ', ' ', ' ', ' '};
            app.Slider_27.Orientation = 'vertical';
            app.Slider_27.ValueChangingFcn = ...
                createCallbackFcn(app, @Custom27, true);
            app.Slider_27.MinorTicks = [];
            app.Slider_27.Position = [811 69 3 113];

            % Create Slider_28
            app.Slider_28 = uislider(app.UIFigure);
            app.Slider_28.Limits = [-13 13];
            app.Slider_28.MajorTicks = [];
            app.Slider_28.MajorTickLabels = {'', ' ', ' ', ...
                ' ', ' ', ' ', ' ', ' ', ' '};
            app.Slider_28.Orientation = 'vertical';
            app.Slider_28.ValueChangingFcn = ...
                createCallbackFcn(app, @Custom28, true);
            app.Slider_28.MinorTicks = [];
            app.Slider_28.Position = [841 69 3 113];

            % Create Slider_29
            app.Slider_29 = uislider(app.UIFigure);
            app.Slider_29.Limits = [-13 13];
            app.Slider_29.MajorTicks = [];
            app.Slider_29.MajorTickLabels = {'', ' ', ' ', ...
                ' ', ' ', ' ', ' ', ' ', ' '};
            app.Slider_29.Orientation = 'vertical';
            app.Slider_29.ValueChangingFcn = ...
                createCallbackFcn(app, @Custom29, true);
            app.Slider_29.MinorTicks = [];
            app.Slider_29.Position = [869 69 3 113];
```

```
        % Create Slider_30
        app.Slider_30 = uislider(app.UIFigure);
        app.Slider_30.Limits = [-13 13];
        app.Slider_30.MajorTicks = [];
        app.Slider_30.MajorTickLabels = {'', ' ', ' ', ...
            ' ', ' ', ' ', ' ', ' ', ' '};
        app.Slider_30.Orientation = 'vertical';
        app.Slider_30.ValueChangingFcn = ...
            createCallbackFcn(app, @Custom30, true);
        app.Slider_30.MinorTicks = [];
        app.Slider_30.Position = [897 69 3 113];

        % Create Slider_31
        app.Slider_31 = uislider(app.UIFigure);
        app.Slider_31.Limits = [-13 13];
        app.Slider_31.MajorTicks = [];
        app.Slider_31.MajorTickLabels = {'', '', '', ''};
        app.Slider_31.Orientation = 'vertical';
        app.Slider_31.ValueChangingFcn = ...
            createCallbackFcn(app, @Custom31, true);
        app.Slider_31.MinorTicks = [];
        app.Slider_31.Position = [926 68 3 113];

        % Create UIAxes
        app.UIAxes = uiaxes(app.UIFigure);
        title(app.UIAxes, '')
        xlabel(app.UIAxes, '')
        ylabel(app.UIAxes, '')
        app.UIAxes.PlotBoxAspectRatio = ...
            [4.10230179028133 1 1];
        app.UIAxes.XLim = [50 14000];
        app.UIAxes.YLim = [0 0.1];
        app.UIAxes.ClippingStyle = 'rectangle';
        app.UIAxes.ColorOrder = [0.9216 0.3373 0;0.851 ...
            0.3255 0.098;0.9294 0.6941 0.1255;0.4941 ...
            0.1843 0.5569;0.4667 0.6745 0.1882;0.302 ...
            0.7451 0.9333;0.6353 0.0784 0.1843];
        app.UIAxes.GridColor = [1 1 1];
        app.UIAxes.MinorGridColor = [1 1 1];
        app.UIAxes.XTickLabel = '';
        app.UIAxes.XMinorTick = 'on';
        app.UIAxes.YTick = [];
        app.UIAxes.Color = [0 0 0];
        app.UIAxes.XMinorGrid = 'on';
        app.UIAxes.XScale = 'log';
        app.UIAxes.BackgroundColor = [0 0 0];
        app.UIAxes.Position = [78 325 771 260];

        % Create TrackDropDown
        app.TrackDropDown = uidropdown(app.UIFigure);
        app.TrackDropDown.Items = {};
        app.TrackDropDown.FontName = 'Batang';
        app.TrackDropDown.BackgroundColor = [0.651 ...
            0.651 0.651];
```

```
            app.TrackDropDown.Position = [88 253 124 30];
            app.TrackDropDown.Value = {};

            % Create PlayButton
            app.PlayButton = uibutton(app.UIFigure, 'push');
            app.PlayButton.ButtonPushedFcn = ...
                createCallbackFcn(app, @play, true);
            app.PlayButton.BackgroundColor = [0.651 0.651 ...
                0.651];
            app.PlayButton.FontName = 'Batang';
            app.PlayButton.Position = [235 253 69 30];
            app.PlayButton.Text = 'Play';

            % Create StopButton
            app.StopButton = uibutton(app.UIFigure, 'push');
            app.StopButton.ButtonPushedFcn = ...
                createCallbackFcn(app, @Stop, true);
            app.StopButton.BackgroundColor = [0.651 0.651 ...
                0.651];
            app.StopButton.FontName = 'Batang';
            app.StopButton.Position = [326 253 68 30];
            app.StopButton.Text = 'Stop';

            % Create DropDown
            app.DropDown = uidropdown(app.UIFigure);
            app.DropDown.Items = {'Flat', 'Rock', 'Pop', ...
                'Bass', 'Treble', 'Vocal', 'Classical', ...
                'Hip-Hop', 'Dance', 'Jazz', 'Powerfull', ...
                'Shitty music', 'MUU', 'Custom'};
            app.DropDown.ValueChangedFcn = ...
                createCallbackFcn(app, @Preset, true);
            app.DropDown.FontName = 'Batang';
            app.DropDown.BackgroundColor = [0.651 0.651 0.651];
            app.DropDown.Position = [520 254 125 30];
            app.DropDown.Value = 'Flat';

            % Create RecButton
            app.RecButton = uibutton(app.UIFigure, 'push');
            app.RecButton.ButtonPushedFcn = ...
                createCallbackFcn(app, @Rec, true);
            app.RecButton.BackgroundColor = [0.651 0.651 ...
                0.651];
            app.RecButton.FontName = 'Batang';
            app.RecButton.Position = [769 253 65 30];
            app.RecButton.Text = 'Rec';

            % Create Lamp
            app.Lamp = uilamp(app.UIFigure);
            app.Lamp.Position = [842 253 29 29];
            app.Lamp.Color = [1 0 0];

            % Create Switch
            app.Switch = uiswitch(app.UIFigure, 'rocker');
            app.Switch.Orientation = 'horizontal';
```

```
            app.Switch.Visible = 'off';
            app.Switch.Tooltip = {'FFT off/on'};
            app.Switch.Position = [910 385 54 24];
            app.Switch.Value = 'On';

            % Create Knob
            app.Knob = uiknob(app.UIFigure, 'continuous');
            app.Knob.Limits = [1 50];
            app.Knob.MajorTicks = [1 50];
            app.Knob.MajorTickLabels = {''};
            app.Knob.MinorTicks = [10 20 30 40];
            app.Knob.Tooltip = {'FFT update frequency'};
            app.Knob.Position = [901 440 72 72];
            app.Knob.Value = 25;

            % Create Image2
            app.Image2 = uiimage(app.UIFigure);
            app.Image2.Position = [59 299 831 310];
            app.Image2.ImageSource = 'GUI frame.png';

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = app1gui

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            % Execute the startup function
            runStartupFcn(app, @Start)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

References

[1] Vesa Välimäki, Joshua D. Reiss, *All About Audio Equalization: Solutions and Frontier*, Department of Signal Processing and Acoustics, Aalto University, May 2016.

[2] Steffi Knorn, *Signals and Systems*, Signals and Systems at department of engineering sciences at Uppsala University, 2018.

[3] Julius O. Smith III, *Introduction to Digital Filters with Audio Applications*, Center for Computer Research in Music and Acoustics (CCRMA), September 2007 edition.

[4] Alan V. Oppenheim and Ronald W. Schafer, *Discrete-Time Signal Processing* - second edition, Prentice-Hall, 1999.

[5] *Sample- and Frame-Based Concepts*, https://www.mathworks.com/help/dsp/ug/sample-and-frame-based-concepts.html. Accessed: 2019-05-07.

[6] J. Rämö, V. Välimäki, and B. Bank. High-Precision Parallel Graphic Equalizer. IEEE/ACM Transactions on Audio, Speech and Language Processing, Vol. 22, No. 12, pp. 1894-1904, December 2014. DOI: 10.1109/TASLP.2014.2354241

[7] Balázs Bank,*Direct design of parallel second-order filters for instrument body modeling*, Laboratory of Acoustics and Audio Signal Processing at Helsinki University of Technology, 2007.

[8] Johannes Langelaar (2019). One third octave graphic equalizer (https://www.mathworks.com/matlabcentral/fileexchange/71618-one-third-octave-graphic-equalizer), MATLAB Central File Exchange. Retrieved May 23, 2019.