NTNU

Innovation and Creativity

Experience with the Construction and Use of Polyphonic Test Signals based on Single Monophonic Recordings for Localisation Listening Tests

Torbjørn Ursin

Master of Science in Electronics
Submission date: June 2007
Supervisor: Peter Svensson, IET

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Problem Description

Localising single instruments in the orchestra is normally an easy task. Even when we are listening to ensemble recordings from the concert hall the localisation of instruments and instrument families are still possible depending on the recording technique and listening facilities. We have reason to believe that spectral variations among instruments form major cues for awareness and localisation.

The present work is dedicated to the development of methods for laboratory tests of single source localisation based on highly controlled musical ensemble signals. The development and construction of polyphonic test signals based on one or more anechoic mono recordings is an essential part of the project.

Assignment given: 15. January 2007
Supervisor: Peter Svensson, IET

Abstract

The paper presents experiments made in search of answers to two principal questions:

1. Can one single musician be made to sound like several musicians playing together?
2. In a music ensemble, where one of its constituents has a distinctive spectrum; how do the deviant spectral components influence a listener's ability of localising the source?

In the first part of the experiment, a flute ensemble was attempted simulated. Based on a recoring of one flute playing a short piece, the flute was multiplied into a quintet. On the way, several properties were manipulated in an attempt to make the quintet sound like a real quintet; timing, spectrum, intensity, and phase.

In the second part, one flute in a quintet was subject to a spectral tilt, i.e. high frequency components were boosted while low frequency components were diminished.

A test panel was engaged to help evaluating the questions. First, the panel compared the simulated quintet to a reference quintet, trying to identify the simulation from the reference. Subsequently, listening to a reference quintet, the panel tried to localise the one flute which had undergone a spectral tilt. A musical piece was played 5 times; first, one of the flutes was moderately tilted, then the tilt's magnitude was increased for every run until eventually being noticeable. For each run, the test panel was asked to indicate the tilted flute, or a random flute if none appeared tilted to them.

The majority of the test panel did not manage to tell the simulated quintet from the reference. However, the reference may have been imperfect, and the simulation process somewhat affects sound quality. When it comes to localisation, a rather excessive tilt was necessary for the test panel to be able to localise it - even though more moderate tilts were clearly audible.

Contents

1 Introduction

The project covered in this report consists, roughly, of two main parts:

1. An investigation of the human ability to localise one spectrally distictive instrument in an ensemble of several instruments. This task aimed to set up a group of musicians, all equipped with the same instrument, and investigate how much of a spectral manipulation must be made to one instrument for a group of test persons to be able to localise it. Note how the term ”localise” differs from the related term ”detect” in this context; detecting means realising that there is a difference, localisation also implies determining the source of the aberration.

2. The simulation of a music ensemble based on one single instrument. This effort was initially meant as a test tool for the localisation experiment, but as time went by it grew to become a project on its own. The idea was to simplify the production of a music ensemble by recording one single instrument and then copying it into an arbitraty number of instruments - in a way that made it sound like a human ensemble. Eventually, a flautist was engaged to make the reference for the localisation, and the simulation received an evaluation on its own - against a reference made by the same flautist.

Neither topic has previously undergone much research. Synthetisation of musical instruments has been subject to extended research for decades, but multiplication of a real instrument has been shown little interest to date. The chorus effect, known from synthesisers, aims to perform just this by adding some time delays and pitch variance, but is usually employed to richen the sound, not to make a credible ensemble effect. Daniel Kahlin and Sten Ternström at KTH, Stockholm, experimented on making a similar multiplication of a singing voice to produce a choir [1].

Research on spectral characteristics normally focuses on localisation in the vertical plane, which in human hearing is based on spectral analysis (section 3.1.3). Extensive research in this field has been made by e.g. Jens Blauert [2] and Dale Purves et al. [3]. However, the impact of a spectral distinctivity in *horizontal* localisation has been paid little attention.

Basically, this report will aim to answer two questions:

1. Can one single music instrument be made to sound like a whole group in a way that sounds natural?
2. In a musical ensemble consisting of several similar instruments, of which one has undergone a spectral tilt, how strong a tilt is necessary for a listener to be able to localise it?

The report is structured as a standard report. The theory section provides some background to binaural hearing and spatial positioning through loudspeakers, a reader already familiar with these subjects may skip to section 4. The ensemble simulation and the listening tests are described, followed by results and some interpretations of the results.

2 Acknowledgements

This effort was carried out in the spring of 2007 at the Institute of Electronics and Telecommunication at the Norwegian University of Science and Technology, as part of a master's degree in acoustics. It was supervised by assistant professor Jan Tro (main supervisor) and professor Peter Svensson, both employed at the institute.

I would like to express my gratitide to the people who participated in the project, whatever their degree of participation. First of all, thanks to the voluntary test subjects who spent their time on my project without any reward but my gratitude. Further PhD student Audun Solvang and master student Espen Moberg for their programming contributions, as well as their assistance at the laboratory, associate professor and professional flautist Trine Karlsen, who kindly allowed me to make use of her excellent abilities as a flute player, and of course the supervisors Jan Tro and Peter Svensson who offered assistance and encouragement throughout the project, all made valuable contributions. Their help has been essential to the completion of this report.

3 Theory

3.1 Binaural hearing

Human ability of localising a sound is mainly based upon the fact that we have two ears, situated in a certain distance and with a muffling obstacle - the head - between them. This provides breeding ground for two methods of localisation in the horizontal plane: Analysis of **time differences (TD)** and **intensity differences (ID)** between the ears. As a third method, **spectral analysis** is used for localisation in the vertical plane.

3.1.1 Time difference

Due to the longer travel distance, a sound wave coming from the side will arrive slightly later at one ear than the other. The difference is little; given an air speed of 340 m/s, a person with a head diameter of 20 cm - which is quite large - listening to a sound source directly to the side, experiences an interaural difference of less than 0.6 ms. If the sound wave comes from any other angle, the difference is naturally even smaller. Tests show that humans, under certain circumstances, are able to detect interaural differences of a mere 10 μs. In terms of localisation, this corresponds to a remarkable resolution of 1°[3].

TD is used to localise low frequency sounds. For frequencies below 3 kHz, signals transmitted from the ear follow the incoming sound wave's phase (oscillations of more than 3 kHz are just too rapid to follow[1])[3]. Nerve impulses are sent towards the medial superior olive (MSO), a part of the brain specialised on translating interaural time differences into localisation data. The MSO contains neurons with bipolar dendrites connected to both cochlei, as shown in figure 1.

The detector neurons in the MSO may be considered "AND" ports, responding most strongly when they receive input from both cochlei simultaneously. Because the MSO extends laterally, each cell is positioned in a certain distance from each cochlea. It takes some time travelling along the neural pathway between the cochlea and the MSO; hence if a nerve signal from

[1]In terms of localisation of sounds, barn owls are by far our superiours. Their ears transmit phase locked signals up to 9 kHz.

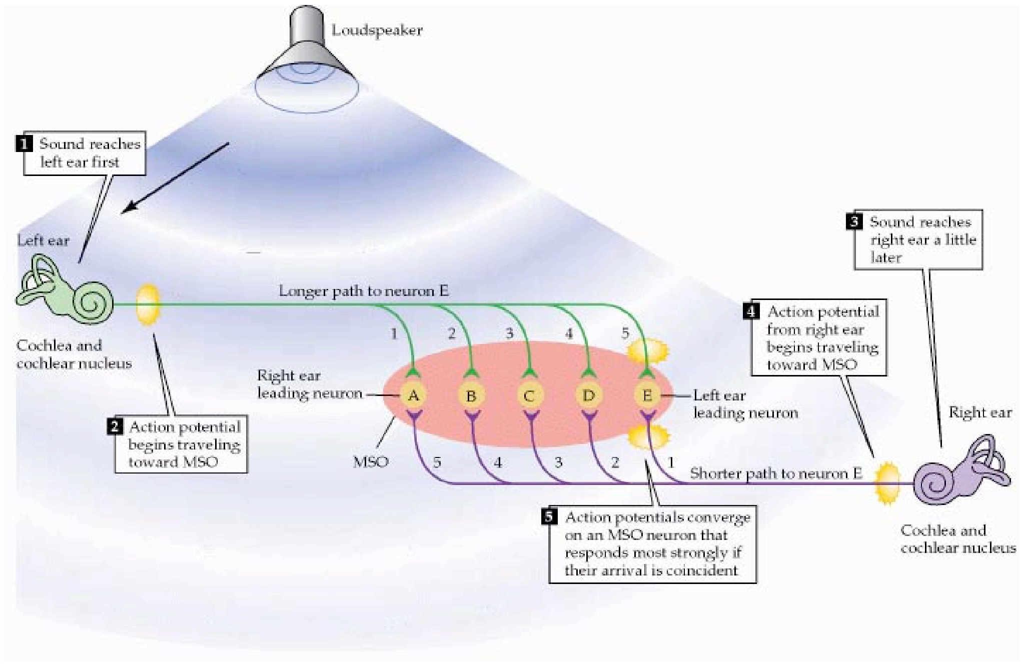

Loudspeaker
1 Sound reaches left ear first
Left ear
Cochlea and cochlear nucleus
2 Action potential begins traveling toward MSO
Longer path to neuron E
1 2 3 4 5
Right ear leading neuron
A B C D E
Left ear leading neuron
MSO
5 4 3 2 1
Shorter path to neuron E
3 Sound reaches right ear a little later
4 Action potential from right ear begins traveling toward MSO
Right ear
Cochlea and cochlear nucleus
5 Action potentials converge on an MSO neuron that responds most strongly if their arrival is coincident

Figure 1: Overview of the TD based localisation mechanism (from [3])

one ear gets a head start on the signal from the other ear, the two signals will meet not in the middle, but in a more periphere neuron. Thus, each neuron responds to sounds arriving with a certain interaural time difference, i.e. from a certain horizontal position. This way, the ingenious circuitry in the MSO accomplishes a resolution of 10 μs, even though its neural components operate in the millisecond range.

Although the transmitted nerve signals are phase-locked up to 3 kHz, interaural phase difference is used for localisation only up to 1.9 kHz, approximately. For wavelengths shorter than the distance between the ears (ca. 18 cm, on average), phase difference is ambiguous - one can no longer determine whether a phase difference of, say, 1/3 of a period represents a delay of 1/3 of a period, or a lead of 2/3 of a period.

Above this critical frequency, the sound wave's envelope is used for localisation of a sound source. Basically, interaural time differences are calculated from when the sound wave first arrives at the two ears, rather than interaural phase differences. However, the performance of this method is assumed to be limited compared to phase- and intensity-based methods[4].

3.1.2 Intensity difference

When the frequency exceeds 3 kHz[2], signals sent from the ears are no longer phase-locked, but follow the sound wave's envelope. Thus TD provides no cue to localisation, and another strategy is required for localisation of sound sources.

To sound waves of wavelengths shorter than the diameter of the head, the head functions as an obstacle. Hence a high-frequency sound coming from the side is muffled before reaching the opposite ear. The resulting intensity difference (ID) is used as a clue to determine the direction of an incoming sound wave. For the above-mentioned, rather large-headed person with a head diameter of 20 cm, ID occures starting at approximately 1.7 kHz. A person with a more average-size head (18 cm) experiences ID from about 1.9 kHz[3].

Intensity difference is sensored in the lateral superiour olives (LSO) in cooperation with the medial nucleus of the trapezoid bodies (MNTB), see figure 2. Each LSO - there are two of them, one on each side - receives input from both cochlei, and responds corresponding to its stimulus. However, input from the LSO's contralateral cochlea is inverted on its way through the MNTB, and counteracts LSO activity. Hence two scenarios:

1. An incoming sound from the side stimulates both the LSO and the

[2]This depends not on the size of the head, but the hair cells in the cochlea.

[3]While humans use interaural intensity differences to localise high frequency sounds in the horizontal plane, barn owls use them in the vertical plane. This is possible for several reasons:

- Their ability to follow the phase of sound waves up to 9 kHz enables them to use TD for localisation even at high frequencies.
- One of their ear canals points upwards, the other one downwards. This asymetry provides a cue to vertical interaural differences not available to symmetrically positioned (e.g. human) ears.

From an evolutionary point of view, this seems reasonable. A high-resolution vertical localisation system is crucial to barn owls, because they navigate in all three dimensions while flying and hunting from above. Humans traditionally stay on the ground, along with other humans, animals and almost every other sound-making item of interest, and are better served by precise horizontal navigational abilities.[5]

MNTB. Because of the head's muffling effect, the MNTB receives less stimulus than the LSO, resulting in a net excitation of the LSO. On the opposite side, the LSO is throttled by the MNTB input.

2. An incoming sound wave in the midplane provides equal stimulus to the LSO and the MNTB, thus LSO activity is muted.

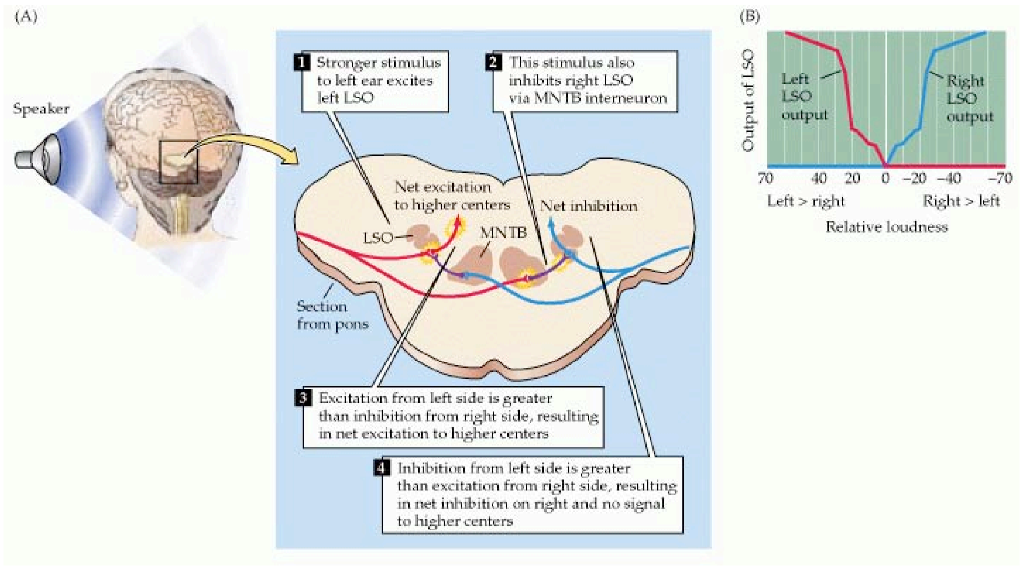

(A)
Speaker
1 Stronger stimulus to left ear excites left LSO
2 This stimulus also inhibits right LSO via MNTB interneuron
Net excitation to higher centers
Net inhibition
LSO
MNTB
Section from pons
3 Excitation from left side is greater than inhibition from right side, resulting in net excitation to higher centers
4 Inhibition from left side is greater than excitation from right side, resulting in net inhibition on right and no signal to higher centers
(B)
Output of LSO
Left LSO output
Right LSO output
70 40 20 0 −20 −40 −70
Left > right
Right > left
Relative loudness

Figure 2: Overview of the ID based localisation mechanism (from [3])

For any perceived sound, only one LSO transmits information on the sound source's location. Furthermore, the source's angular position can be derived from the magnitude of the output.

Interaural ID increases as frequency increases, because the head becomes a more and more prominent obstacle as wavelengths become shorter. Just above the critical frequency at about 1.9 kHz, differences are small. For higher frequencies, differences of 10-20 dB between the ears may occur[6].

3.1.3 Spectral analysis

Neither TD nor ID provides any information on the elevation of a sound source. Because the ears are identical (though mirrored), both in position and shape, no time or intensity difference can be retraced to the vertical position of the source. Nevertheless, a sound's origin can be told from its spectral content, although the vertical resolution is lower than the horizontal (9°at best [7]). This effect is neither examined nor exploited in this project, thus the thorough explanation is omitted. Yet it is mentioned for a complete understanding of the localisation mechanisms of human hearing.

Tests have been made which show that sounds are perceived as coming from the front, above or behind depending on which frequencies are the most prominent. White noise, containing every frequency, sent towards a person from every possible angle, would undergo some spectral modifications before reaching the ear (individual differences apply)[2]:

- Sounds from the front add up with reflections from the shoulders to gain a boost at approximately 3 kHz.
- Sounds from above receive a boost at about 8 kHz due to resonances in the pinnae.
- Sounds from behind are amplified at about 1 kHz.

Consequently, a sound centered around i.e. 8 kHz is likely to be perceived as coming from above. This matter has been examined in e.g. [2] and [8].

3.2 Source position simulated by loudspeakers

Both time and intensity differences are easily emulated by loudspeakers. Exploiting the two mechanisms, a variety of positions can be reproduced even in a normal two-speaker system - sources may be put not only in the actual positions of the loudspeakers, phantom sources may be positioned at any position between the speakers[4].

Both time and intensity differences are used in stereo recordings to position phantom sources. Time difference in recordings is produced by a two-microphone setup, where microphones are situated within a distance of ca. 20 cm, probably corresponding to the distance between the ears[7].

Far more commonly used are intensity differences between the speakers. ID can be produced using two directive microphones, situated at the same spot, but recording in different directions. The microphones are often cardioids with an angle of 110°between their axes[7]. ID can also be simulated by the panning control, which exists on any mixing console. The panning control makes the intensity of one channel stronger than the other, while preserving the total intensity.

Intensity difference is the more common positioning method not only for practical reasons, there are also psychoacoustical aspects which make it favourable. At low frequencies, which is the more important range to how sounds are perceived, the source's position is determined by interaural time differences. And - surprisingly - interaural time differences occur when the two loudspeakers differ only in intensity! In short, ID at the speakers makes TD at the ears.

This is true for wavelengths so long that the extra travel distance to the farther ear (*dt* in figure 3) only makes a small phase difference between the ears.

A sound wave sent from one speaker arrives slightly later at the farther ear, simply because the travel distance is longer. If two speakers simultaneously

[4]In theory, phantom sources may be put at virtually any position - even behind the listener - only using loudspeakers in front. Techniques have been known for decades, but are normally practically infeasible - they often require an excessive amount of equipment (e.g. wavefield synthesis), or require the listener to stay in one exact position at any time(e.g. ambisonics). Spectral analysis is used to some extent (e.g. by QSound). Still, regular two-speaker stereo is by far the most common.

emit sound waves that differ in intensity, the two ears experience different scenarios; at one ear, the large wave adds up with the smaller, delayed wave. At the other ear, the small wave adds up with the larger, delayed wave. Adding two waves with different amplitudes produces a resulting wave time-shifted towards the larger of its constituents, as shown in figure 3.

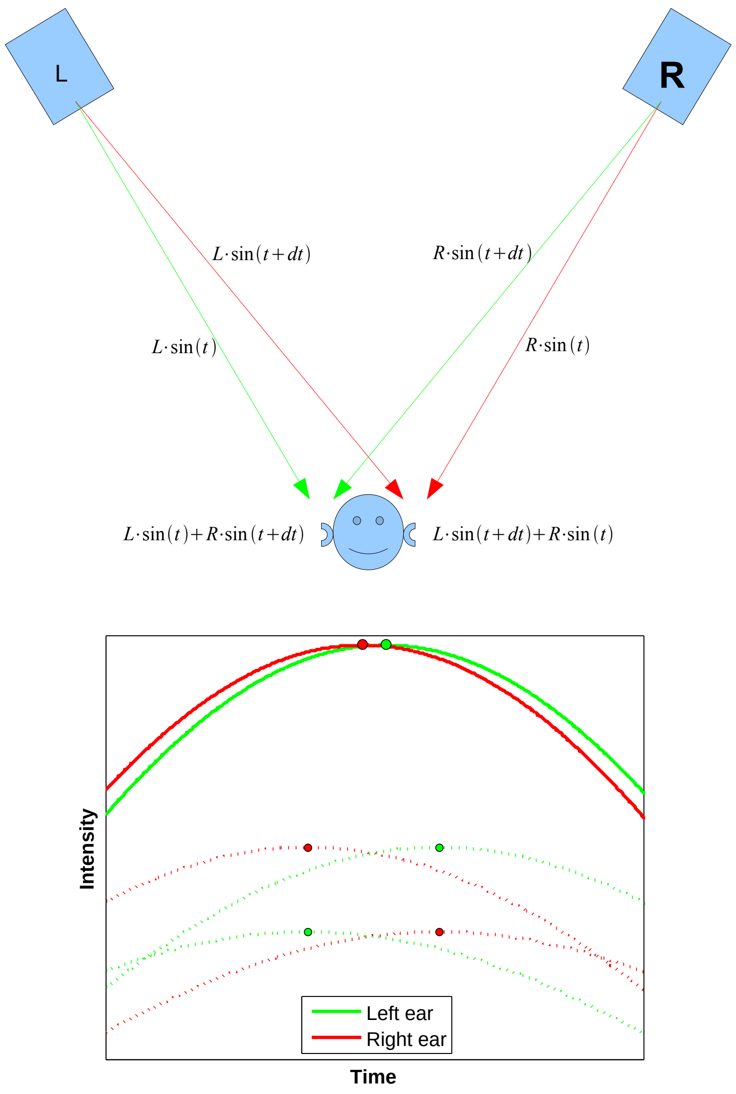


Figure 3: Addition of waves with different intensities. The green, dotted lines represent the waves arriving at the left ear, the red at the right. The dotted lines sum up at the ears, resulting in the solid lines - which differ only in phase.

4 Method

4.1 Simulation experiments

A significant part of this project turns upon the simulation of an ensemble of instruments based on the recording of one single instrument, as illustrated in figure 4. Such a simulation can not be done just by multiplying an instrument sample, the outcome of that would sound innatural. Basically, each instance of the instrument must be manipulated to sound differently, i.e. a clarinet sample must be manipulated to sound like a different clarinet - of course while still clearly sounding like a clarinet. Several properties can be altered in order to achieve this:

- Time delays
- Spectral profile
- Intensity profile
- Phase

A goal was defined: *Based on a tune played by one single flute, make the tune sound as if it were played by a flute quintet.*[5]

4.1.1 Time delaying

In an ensemble of music instruments playing together, the timing between individual instruments is never completely accurate. Inevitably, there are small deviations in impact and release of each note. These deviations are audible, and although a listener may not necessarily be able to point them out directly, they do influence the musical experience.

In an electronically generated ensemble, such deviations are absent unless intentionally added. Inaccuracy is emulated by time-shifting each tone of each recording a random number of samples. Two methods were tried:

[5]Even though this was the exact goal set for this effort, all scripts produced work with any number of any instrument through an automated process.

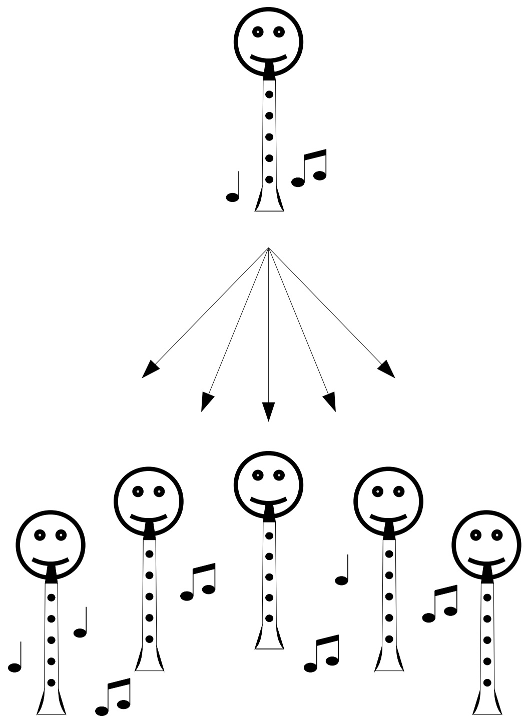


Figure 4: A quintet made from one single instrument recording

1. The original recording was divided into overlapping Tukey windows, and each window time-shifted a random number of samples. The Tukey window was chosen for its rather flat-topped temporal profile, see figure 5. The relation between the maximum shift and the window overlap must be set to avoid pauses in an originally continuous tone, i.e. the overlap must be larger than twice the maximum shift.

   When correctly done, the beginning and the end of a tone are likely to be shifted, without any gaps appearing in the tone. The temporal profile is somewhat manipulated as well, due to the summing of overlapping windows, and the windows' fading temporal profile.

2. Manually indicated in advance, each note was time-shifted a random number of samples. This is a time-demanding method, but on short pieces of low complexity it may offer a higher precision level, simplify correlating shifts (as there is supposedly some correlation between delays in performed music; if you're quick on one tone you might over-compensate to be late on the next etc.), and ensure that tones

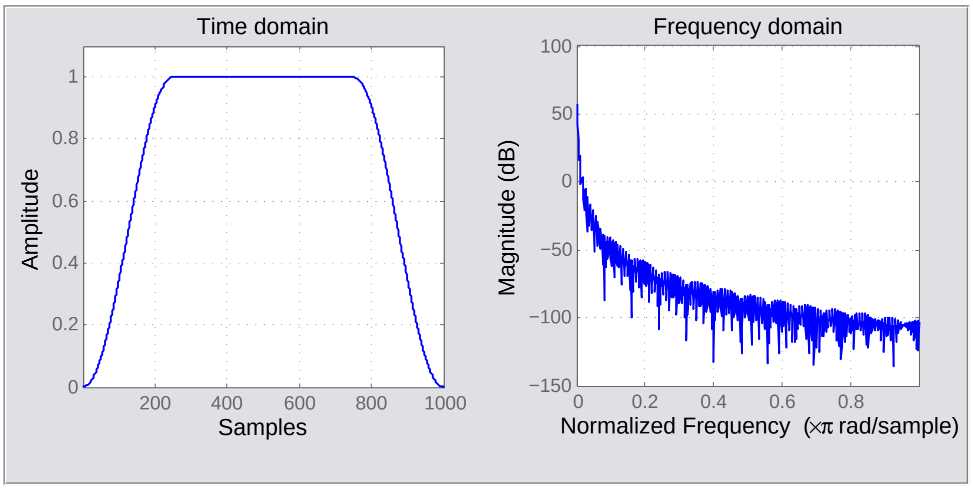


Figure 5: Time domain and frequency domain plot of a Tukey window

have the same length and amplitude profile even when delayed.

Alas, stocastic delays between notes led to innatural transitions, presumably particularly audibly on notes sliding away from each other. Correlated time shifts might help on this matter, but was not implemented in this project. Some experiments were made by this method, but it was eventually abandoned.

4.1.2 Spectral manipulation

No matter how fine-tuned, each specimen of an instrument has its own spectral profile - overtones are more or less dominant from one instrument to another, for example. This effect was emulated by a small tilt of each instrument recording's spectrum.

For this simulation, the input signal was filtered by a linearly increasing or decreasing filter, ranging from 20 Hz to half the sampling frequency of the input signal - often 22.05 kHz. The magnitude of the fundamental, or the most dominant overtone, was used as reference, and thus kept at its level. Consequently, as lower frequencies were decreased, higher frequencies were increased and vice versa.

As the human ear works in a logarithmic fashion[6], a numerically small augmentation is relatively more significant than a large one, with respect to its magnitude. Likewise, a certain augmentation of a low-level sound is more significant than the (numerically) same augmentation of a high-level sound. Hence a numerically linear increase in sound pressure level is actually heard as a logarithmic increase.

Thus, whilst the numerically linear filter applied gives a significant increase/decrease to the primary and secondary overtones, higher order overtones are only slightly further enhanced/diminished. In contrast, the spectral tilt described in section 4.2 is logarithmic, q.v.

4.1.3 Intensity variation

When playing a woodwind or string instrument, small variations in intensity are inevitable. Blowing with a constant pressure is unrealistic, as is stroking the bow with constant velocity. In a music ensemble, uncorrelated variations in each musician's level of intensity lead to inter-instrumental fluctuations as well, as - for instance - one instrument might dominate in one moment, before over-compensating in the next to become slightly drowned out by the other instruments.

For emulating this effect, each instrument was multiplied by a vector fluctuating around 1. The creation of this amplitude variance vector followed the algorithm:

1. Creation of a vector with as many elements as there are seconds in the sound file (rounded - for instance, a 9.7 second instrument sample implies a 10 element vector).
2. Assignment of a random value to each element, values symmetrically distributed around 1.
3. Resampling of the vector to as many elements as the instrument signal, with a reference point approximately every second. Reference points are maxima, minima, or saddle points.

The result may be seen in figure 6.

[6]Hence the introduction of the logarithmic dB scale.

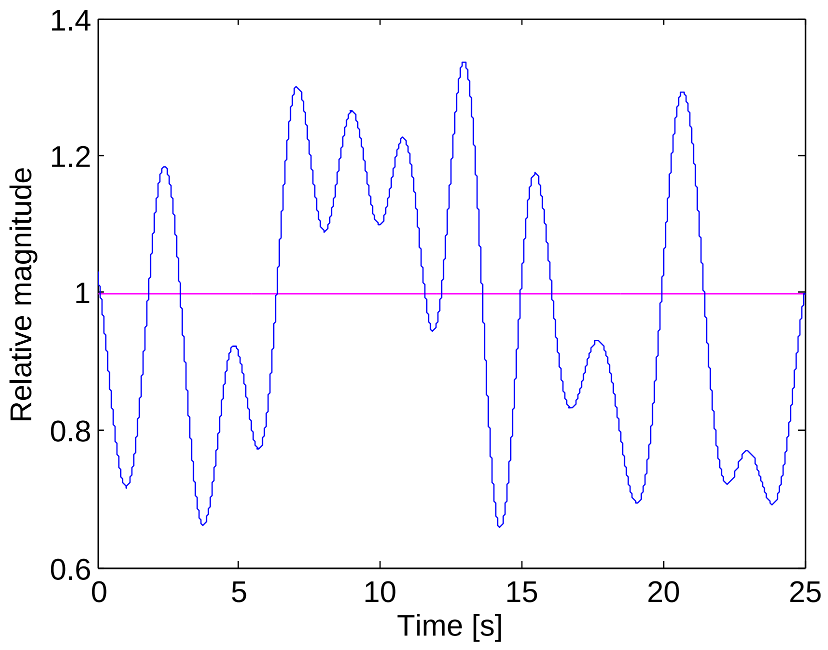


Figure 6: Example of an amplitude variation vector, fluctuating around 1, with a potential maximum deviation of 0.7

Subsequently, each value of the instrument sample was multiplied by its corresponding value in the amplitude variance vector. As each instrument sample had its own stochastic amplitude variance vector, inter-instrumental variations were emulated as well as the fluctuations of each individual instrument.

4.1.4 Phase shift

If an instrument sample is multiplied, all samples are naturally correlated. Real instruments playing together act, however, incorrelatedly, as even if they happen to play the same frequency, they are not likely to be in phase.

The difference between correlated and incorrelated signals is noticeable. The addition of two correlated sounds results in an increase of 6 dB, while the addition of two incorrelated signals only leads to a 3 dB increase. Thus the instrument samples must be decorrelated in order for the simulation to be credible.

Decorrelating the instruments is done by shifting their phase randomly. The shift is done within one single period and leads to a time shift which is inaudible, but still sufficient to decorrelate the signals. Three identical, but decorrelated sound files are shown in figure 7.

4.1.5 Parameters

Numerous pilot tests were run, and suggested the following variable parameters:

1. The maximum time shift, either backwards or forwards in time, was set to 130 milliseconds. Shifts are randomly distributed between 0 and this value. Note that the maximum delay between notes is twice this value, as windows may slide away from each other.
2. Intensity variations were set to fluctuate between 0.3 and 1.7, relative to the original value. These values correspond to a maximum increase of 2.3 dB, and a maximum decrease of 5.2 dB. However, during resampling of the amplitude variation function (section 4.1.3), the function is smoothed, making realistic extreme values more moderate. Figure 6 shows an example function.
3. The maximum magnitude of the spectral tilts was set to ± 3 dB, i.e. a difference of 6 dB between low frequencies and high frequencies.

Source code is presented in appendix B.

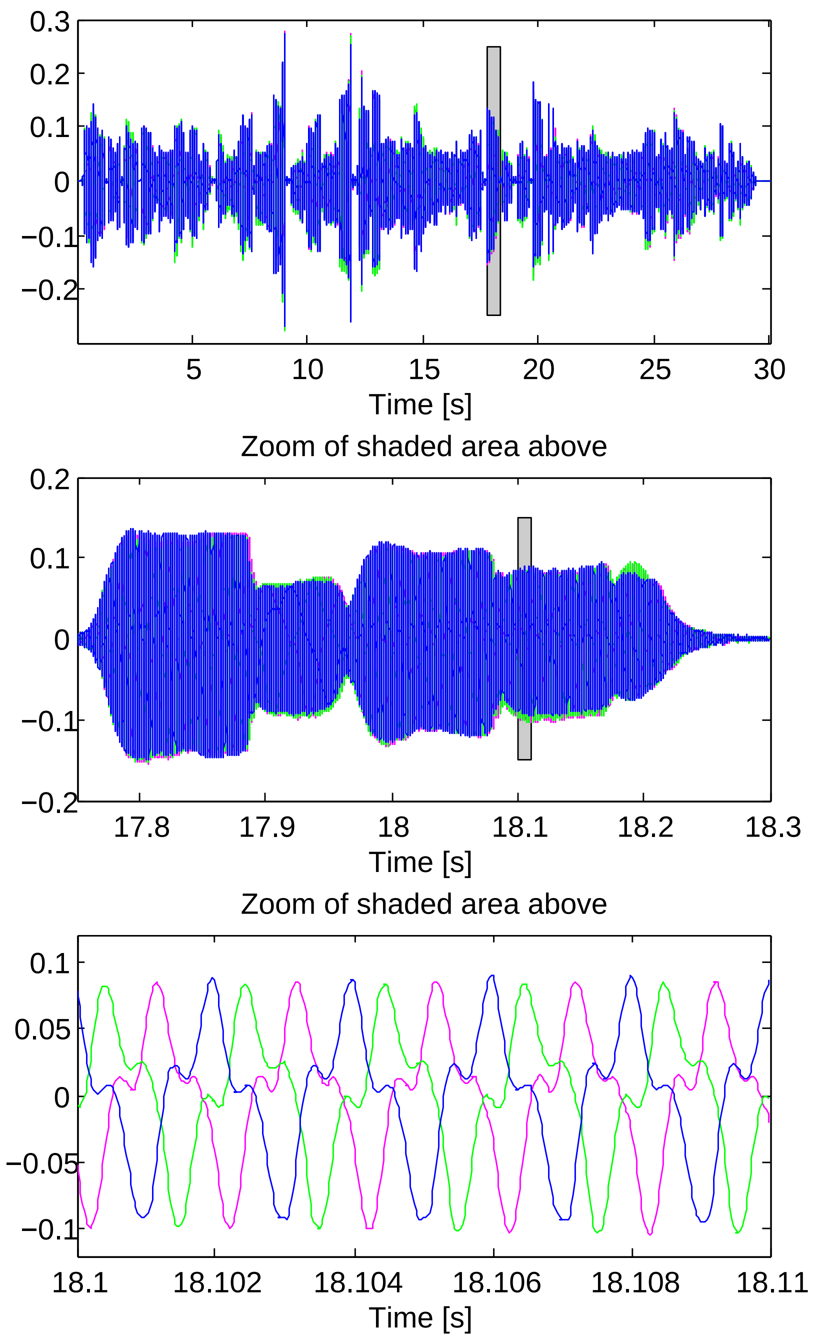


Figure 7: 3 identical sound files after decorrelation. 1st figure: The sound files aren't audibly changed. 2nd figure: A zoom still barely shows evidence of manipulation. 3rd figure: Further zooming proves that the files are uncorrelated.

4.2 Spectral tilt

The issue of the project was, principally, to investigate an instrument's spectral components' influence on our ability of horizontal localisation. A relevant scenario is an ensemble of music instruments of which the spectral properties of one of them make it stand out from the rest. Hence a test was designed for the investigation: *In a flute quintet, one of the flutes becomes increasingly spectrally distinctive. Engage a test panel to determine at which level the distinctive flute can be localised.* The distinction consisted of a spectral tilt, in which high frequencies were *increased* and low frequencies *decreased*.

The spectral tilt was achieved by the means of the graphic equaliser shown in figure 8, and presented in appendix C. It imitates the equaliser common in home stereo systems (although not a real-time equaliser), and works by applying a filter to the input file according to pre-adjusted sliders. Each slider works in a 1/1 octave band; hence the 10 sliders adjust the following octave bands:

31.5 Hz - 63 Hz - 125 Hz - 250 Hz - 500 Hz - 1 kHz - 2 kHz - 4 kHz - 8 kHz - 16 kHz

Five different tilts were defined as described below. Note that a "4 dB tilt" does not necessarily mean that high frequencies are amplified by 4 dB with respect to low frequencies. The filters are named by the position of the sliders when the filters were designed. Because the filter function is always continuous, each slider's setting influences the adjacent octave bands. Thus all sliders are correlated, even though each of them is individually adjustable, and their names do not necessarily correspond to their real magnitudes. Filter magnitudes are listed in table 1.

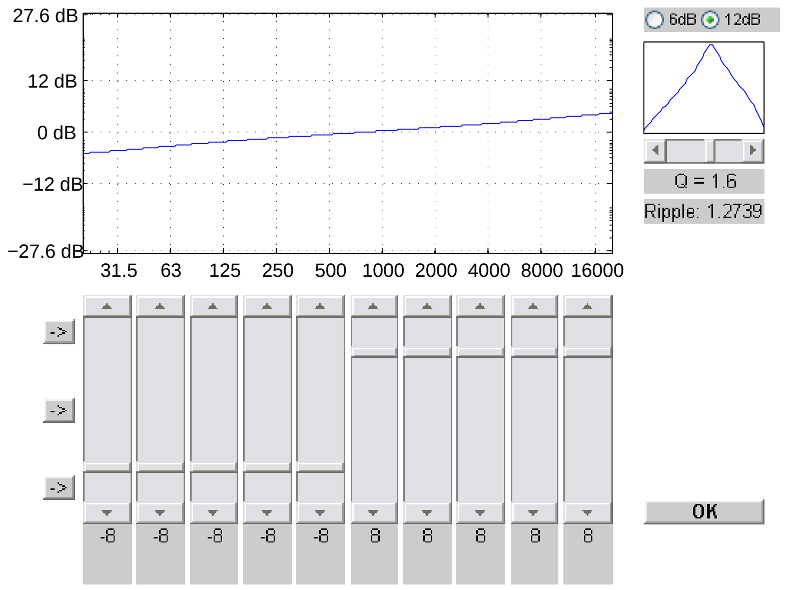
4 dB - Applies the filter shown in figure 8, only with a $\pm$ 4 dB slider setting and hence a flatter filter profile.

6 dB - Applies the filter shown in figure 8, only with a $\pm$ 6 dB slider setting.

8 dB - Applies the filter shown in figure 8.

10 dB - Applies a slightly steeper version of the filter shown in figure 8.

12 dB - Applies a rather extreme version of the filter in figure 8.



6dB [✓ 12dB]

Q = 1.6

Ripple: 1.2739

-8 -8 -8 -8 -8 8 8 8 8 8

Figure 8: Interface of the graphical equaliser utilised to apply the spectral tilt.

Further, the terms ”x dB tilt” refers to these filter profiles.

Note that an extensive spectral tilt actually influences the total intensity of the sound. The defined filters increase frequencies above ca. 750 Hz, decreasing the components below. In the case of a flute, there are almost no low frequency components, hence a HF increase accompanied by a LF decrease increases the total intensity.

	125 Hz	250 Hz	500 Hz	1 kHz	2 kHz	4 kHz	8 kHz
4 dB	-0.8	-0.4	-0.1	+0.1	+0.4	+0.8	+1.2
6 dB	-1.3	-0.7	-0.2	+0.2	+0.7	+1.3	+2
8 dB	-2.1	-1.2	-0.4	+0.4	+1.2	+2.1	+3.1
10 dB	-3.6	-2.14	-0.7	+0.7	+2.1	+3.6	+5.2
12 dB	-9.5	-6.4	-3	+3	+6.4	+9.5	+12.5

Table 1: Filter magnitudes at selected frequencies for the different spectral tilts. All values are in dB.

4.3 Listening tests

To evaluate the efforts made during the project, listening tests were run on 8 test subjects, all adults with normal hearing. Two tests were run; one for the simulated music ensemble of section 4.1, one for the issue of localisation of a distinctive sound. Prior to both tests, each subject was thoroughly explained the methods behind test objects and references, and examples were played for the subjects to know what to listen for.

4.3.1 Listening test 1 - simulated ensemble

As the simulation was made to resemble a music ensemble, a listening test was arranged against a reference in order to determine its credibility. The reference was made in an anechoic room by a professional flautist. The flautist was presented two short musical pieces, which she played and recorded:

1. Charles-Marie Widor: Suite for flute and piano, 2. movement: Scherzo
2. Ketil Bjørnstad: Sommernatt ved fjorden

Listening to the first recording, she then played each of the pieces seven more times. From these seven, five were picked and combined to a unisonous quintet.

For the simulation test, Widor's scherzo was picked as test object. During the test, the reference and a simulated quintet were played successively to the test subject, followed by a simple question: *Which quintet is simulated?* Only two options were offered, representing the reference and the simulation, there was no blank option. Subjects were demanded to tick one of the options.

Both quintets were played back through an ordinary stereo speaker setup, i.e. only two speakers were used. The five flutes were evenly panned, using intensity difference (section 3.2), between the speakers.

4.3.2 Listening test 2 - localisation experiment

Further, another test was performed to investigate the test subjects' ability of localising one spectrally distinctive (tilted, see section 4.2) instrument in a flute quintet. The aim was to get a clue of the threshold value at which a spectral tilt of one single flute becomes audible. Thus the tilt magnitude started at an inaudible level, gradually increasing until noticeable.

This time the recording of Bjørnstad's "Sommernatt ved fjorden" was selected reference. Each flute was played back through one dedicated speaker; thus 5 speakers were employed. The speakers, or flutes, were positioned as in figure 9, on a circle around the listener, with a 22.5°angle between them; at 315°, 337.5°, 0°, 22.5°, and 45°.

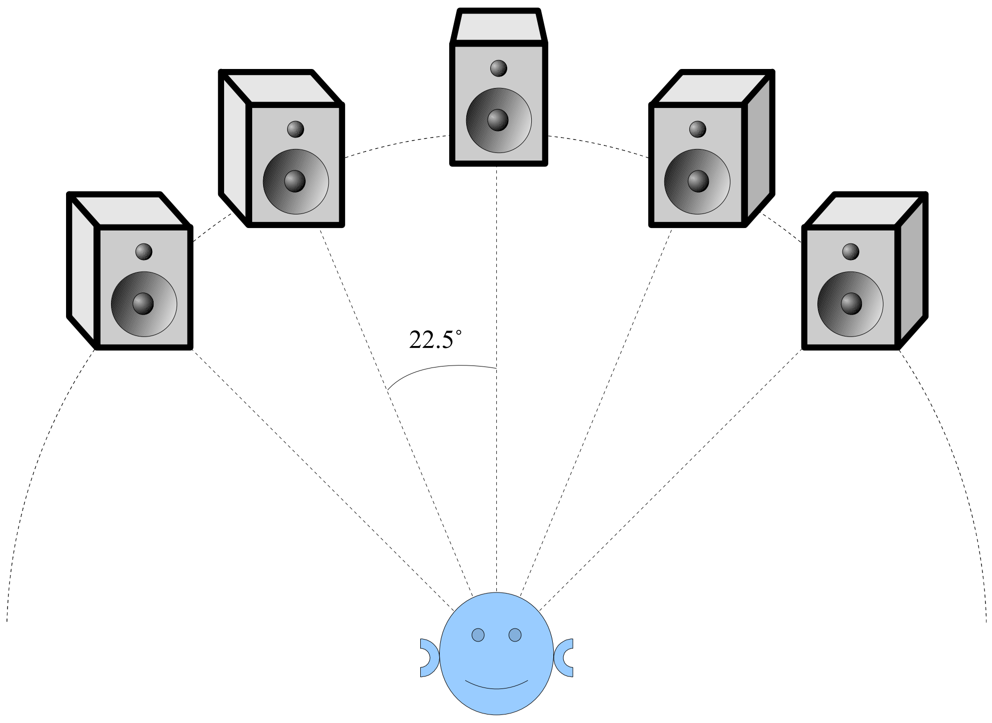


Figure 9: Sketch of listening test setup

The piece was played back 5 times in a row. Each time, one of the flutes was manipulated according to the description in section 4.2. For each run, high frequencies were further increased and low frequencies further decreased, as shown in table 2.

Because of the spectral content of the flute, octave bands up to 250 Hz are almost irrelevant. The fundamental of a flute is normally at 500-2000 Hz, depending on the tone, and spectral components below the fundamental are minimal.

	Spectral tilt
1st run	4 dB
2nd run	6 dB
3rd run	8 dB
4th run	10 dB
5th run	12 dB

Table 2: Spectral tilt in listening test

The tilted flute was not randomly picked; flute no. 4 (at 22.5°) was the tilted one in every test run. Partly for practical reasons, but also for eliminating a major source of error: Localising a periphere, tilted speaker may be easier than a central one, an aspect potentially greatly influencing the test. On the other hand, a non-random speaker requires one assumption to be made: *For every test iteration, localisation is easier than on any previous iteration.* This was assumed throughout the test, and was not further investigated.

Following every run, test subjects were asked to tick one flute which appeared manipulated to them, or a random flute if none appeared manipulated. Hence, presumably, random results would be produced if manipulation was at a non-audible level. The subjects were allowed to turn their head as they wished during the test, as looking towards the source significantly improves the localisation ability[7].

5 Results

This section presents the outcome of the listening tests performed to evaluate the simulation effort and the localisation test. The two subsections summarise the results from their respective tests. No interpretation is done, only plain results from the listening tests are presented.

5.1 Listening test 1

The task of the first listening test was to identify a simulated quintet from a reference. The outcome of the test was as listed below, and as visualised in figure 10.

1. 25% managed to identify the simulation from the reference.
2. 75% failed to identify the simulation from the reference.

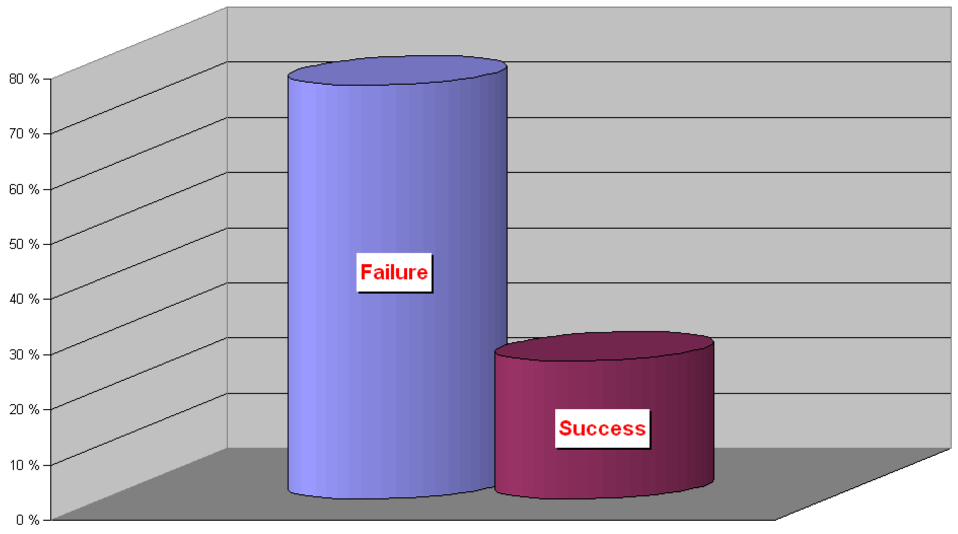


Figure 10: Graphical results of quintet simulation listening test

5.2 Listening test 2

The test subjects' second task was to determine which flute had been subject to a spectral tilt in a unisonous quintet. Results are presented in table 3, and visualised in figure 11. As mentioned in section 4.3.2, flute no. 4 was consistently the manipulated one.

	Flute 1	**Flute 2**	**Flute 3**	**Flute 4**	**Flute 5**
4 dB	12.5	12.5	12.5	**25**	37.5
6 dB	25	25	12.5	**25**	12.5
8 dB	25	12.5	25	**0**	37.5
10 dB	0	37.5	25	**12.5**	25
12 dB	12.5	12.5	0	**62.5**	12.5

Table 3: Listening test results of localisation experiment. Results are given in %.

Considering figure 11, there is a *noise floor* at the highest value which is likely to appear accidentally, given a uniform probability density. A simulation of 100.000 iterations shows that in a group of 8 people picking one random of 5 flutes, 3 or more people would pick the same flute in about 84 of 100 instances. 4 or more would pick the same flute in only 28 of 100. Thus a reasonable position of the noise floor is between these values.

A quick summary of the second listening test:

- Answers are fairly evenly distributed at 6 dB and 8 dB spectral tilts.
- At 4 dB, answers actually tend towards the tilted speaker. However, no values surpass the noise floor, and feedback from test subjects as well as absense of the tendency at 6 and 8 dB imply that the tendency at 4 dB is accidental.
- The 10 dB tilt was claimed by many, and the 8 dB tilt by some, to be audible but difficult to localise. In the former's case, answers tend slightly to the right, i.e. the side of the tilted speaker - though not enough to surpass the noise floor.
- At 12 dB, the majority of the test subjects successfully localised the tilted speaker.

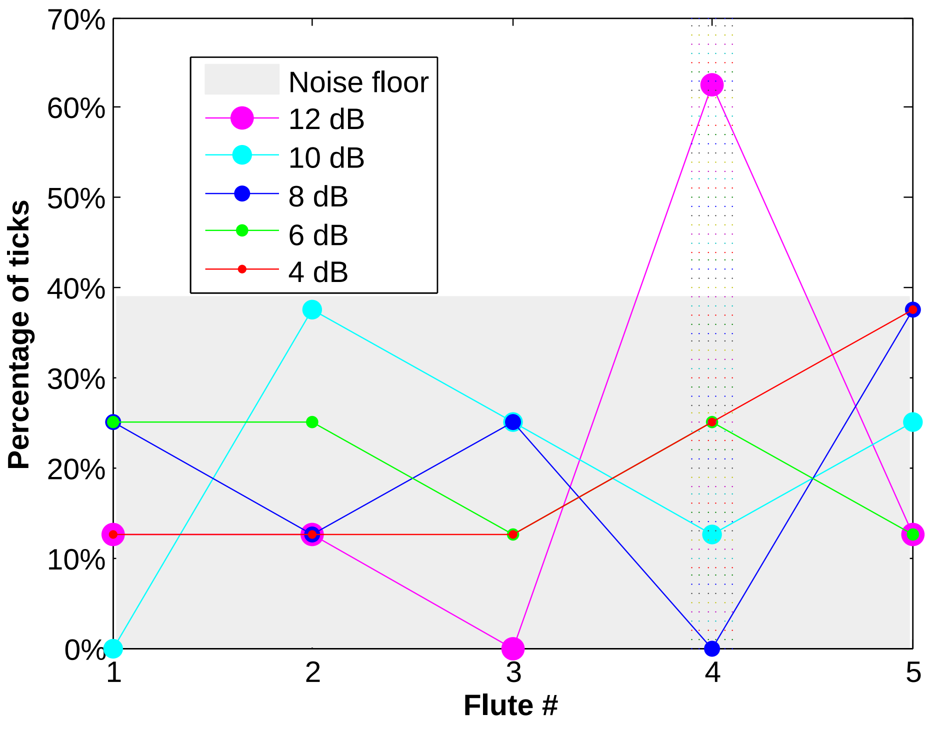


Figure 11: Graphic representation of table 3

- There was no appearant connection between the answers and whether the test subject had a musical background or not.

6 Discussion

6.1 Listening test 1

The outcome of the first listening test is rather surprising. A poor simulation would generate a high percentage of success among the test subjects, as most subjects would recognise the simulation from the reference. However, an ideal simulation would generate a 50% success rate; as noone could tell the simulation from the reference, answers would be random. Thus a success rate as low as 25% suggests that there is something amiss with the test. Reasons could be, for instance:

- High failure rates could appear due to a coincidence, despite a good simulation (section 6.1.1).
- The reference could be inadequate (section 6.1.2).
- Other reasons could apply.

6.1.1 Coincidence?

Even if the simulation and the reference were both flawless, the distribution of answers could be skewed due to a sheer coincidence. By using statistics, the probability of the current skew (or a more extreme one) occuring by accident can be calculated.

Because of the question's simple nature, the only possible answers being "First" and "Second", answers are binomially distributed. Thus probability may be determined by (1) [9].

$$P(x) = \binom{n}{x} (p)^x (1-p)^{n-x} \tag{1}$$

n and x are the total number of answers and correct answers, respectively. Assuming the simulation be ideal, probability of success would be 50% for each test subject, thus $p = 0.5$. Hence

$$P\left(X \leq 25\%\right) = \binom{8}{2} (0.5)^2 \left(1 - 0.5\right)^{8-2} \tag{2}$$

$$P\left(X \leq 25\%\right) = 0.145 \tag{3}$$

Moreover, in (3) the simulation is assumed to be ideal. Such an assumption is unrealistic, in any simulation some minor flaws must be expected. Consequently,

$$\begin{aligned} p &< 0.5 \qquad (4) \\ \Rightarrow P(X \leq 25\%) &< 0.145 \qquad (5) \end{aligned}$$

Thus it is possible, but certainly not probable, that the listening test's outcome be result of a coincidence.

6.1.2 Inadequate reference?

In section 4.3 the procedure for making the reference is described. The flautist's feedback during recording was that it felt unnatural, for one reason in particular: She had no visual contact with her co-player. Visual interaction between the players is a significant aspect of a quintet, highly influent on its sound. But in a quintet made from five individual recordings, there is no visual interaction whatsoever.

Several other disadvantages of such an imitation to a real quintet may also be proposed:

- Each player, and each flute as well, has its own individual character when playing. In the imitated quintet, each part is played by the same player on the same instrument.
- When five players play simultaneously, every player's time- and intensity imprecision is correlated to all the other players. In individual recordings, any imprecision is only correlated to the initial recording, to which the player is playing along.

Considering the potential audibility of these aspects, the peculiar result might be due to an imperfect reference.

6.1.3 Summary

Nevertheless, such a results suggests that the simulation works to satisfaction. Those who managed to distiguish the simulation from the reference stated the following aspects as clues:

- Richer, more dynamic sound
- More natural breathing

Both arguments make sense, as summing overlapping windows slightly alters, and often smooths, the signal's temporal profile. But this effect appears to be rather inconspicuous, as only a few of the test subjects were able to point it out. Besides, feedback from most subjects claimed that both pieces were rather credible.

The principal objection against a simulated quintet, is the same as posed by the flautist against the reference: There is no interaction between the musicians. In a musical ensemble populated by human players, every player makes tiny timing mistakes throughout the piece, each instrument sounds slightly different from the others, and no musician is able to play with constant intensity. All these little errors can be implemented in a simulation to make it sound human. What is harder to implement is the correlation between errors, the effect of the musicians adapting to the others as they play along. In the current simulation, all errors are stochastic, hence uncorrelated.

Additionally, other effects are hard to simulate, such as

- A player's individual character, or sound, which is unique to every musician.
- Improvisation[7].

Thus a simulated ensemble is hardly suited to replace a real musical ensemble when it comes to musical performance. However, the technique might be interesting for quickly and easily producing background music, like accompaniment to a singer or a band. In such a case precision and syncronism is crucial, whereas each musician's individual character is less important. Moreover, a minor lack of dynamic is not really a problem, as background accompaniment is supposed to be just that - in the background.

Even if it works on a woodwind instrument like a flute, that doesn't necessarily mean that it works on another type of instrument - a piano, a guitar,

[7]Music based on Markov chains may actually sound both improvised and fairly human-like, even though the Markov chain is a random process. Markov-based music generators exist e.g. in the computer music programs CSound and MAX.

or a violin, for example. Flutes, and woodwind instruments in general, have simple, neatly structured spectra, whereas string instruments are far more complex. In an early stage of the project, experiments made with a clarinet seemed to work quite well. Experiments made with a guitar didn't work well at all. However, this might also be due to the guitar recording not being anechoic.

In short, results are valid for an anechoic flute recording, and most likely to other woodwinds as well. Whether the same method may be applied to other instruments, remains unanswered.

6.2 Listening test 2

Compared to the first listening test, the second one turned out more as expected. The subjects' responses at 4 dB, 6 dB, 8 dB, and 10 dB tilts are random, as they are unable to localise the tilts. At 12 dB, a clear peak arises on the tilted speaker (figure 11).

The same simulation as in section 5.2 indicates that 62.5 % or more pick the same flute by chance in about 5.2 of 100 instances. Flute no. 4 is picked by 62.5 % or more in 1 of 100 instances. Thus, statistically, it can be stated with *99 % confidence* that the result of the 12 dB tilt in figure 11 is not accidental. Feedback from test subjects supports this conclusion, the majority claimed to easily localise the 12 dB tilt. Still, 37.5 % failed to localise it.

The most interesting result of the second test is not that people were able to localise the 12 dB tilt, nor that the most moderate tilts were inaudible. The mid-range tilts, however, were expected to be more distinctive. Pilot tests suggested audibility at 8 dB and localisation at 10 dB. However, results at both levels failed to surpass the noise floor.

Appearantly, even a clearly audible tilt can not necessarily be localised. The 10 dB tilt and, according to some of the test persons, the 8 dB tilt are audible, although their origin is hard to point out. At the 10 dB tilt, several test subjects announced to be a little annoyed that even though they were able to detect the tilt, they were unable to determine its source.

One possible conclusion may be drawn from this. All test subjects had normal hearing, and thus weren't trained in using spectral distinctiveness for horizontal localisation[8]. Horizontal localisation is normally based on time difference or intensity difference, whereas the difference in this case was mainly spectral. The results imply that the *spectral content means little to horizontal localisation.* At the 12 dB tilt, localisation is most likely achieved by focusing on high frequency components and exploiting their intensity difference.

Obviously, in the case of a music ensemble, where one of the instruments has a distinctive spectral characteristic, *it is difficult to point out the instrument.*

A principal question arises from this conclusion: Is it easier to localise one individual instrument in a live orchestra, as stated in the problem description? Probably yes, because localisation is to a large extent about recognition. Especially spectral localisation benefits from previous knowledge of a sound source - the actual sound is compared to what the sound usually sounds like to determine its direction. Because listening to an orchestra is familiar, any anomality imidiately draws attention. On the other hand, the listening test was a new situation to the test subjects, except for the brief introduction prior to the test.

The results achieved in this test apply mainly to the exact setup employed:

- Increasing the amount of flutes makes the tilt less audible.
- Narrowing the angle between speakers makes localisation more difficult.
- A tilt of a peripheral flute might be easier to localise than a tilt of a central flute.

Note that adjusting the playback volume would probably have no impact on the test results. Increasing the volume does not improve speech intelligibility, as long as the volume is well beyond the background noise in the first place. It is reasonable to assume that this principle applies to a musical context as well.

[8]Some hearing impaired, with normal hearing on only one ear, train themselves to find the source of a sound from its spectrum, even in the horizontal plane.

7 Conclusion

As the effort has been concentrated on two independent, though related, areas, several inferences may be drawn from the results. Two principal questions were posed in the introduction, and experiments were made in an attempt to respond them.

1. By combining timing errors, spectral manipulation, intensity deviations and phase shifts, one musical instrument can be copied into several in a way that sounds fairly natural. The other side of the picture is that such a manipulation goes on the expence of the sound's dynamic and liveliness. Nevertheless, such a method may be useful in certain cases; if a music ensemble is desired in the background of a singer or band, one single instrument can be recorded and multiplied into an ensemble. Naturally a secondary solution to a real ensemble, it may facilitate production in cases where time, money, space etc. makes it difficult to make a full-scale recording.

2. In an ensemble of flutes, the spectral content of one individual flute is probably not a main clue to our ability of localising it. In short, a spectral manipulation of an individual source must be rather extensive to make a difference easy to localise. It turned out that there is a long way to go from hearing that one flute is different, to being able to tell which flute it is.

7.1 Further work

Alongside the questions that have been answered during this work, several others have arisen. Further experimenting on the simulated quintet might be interesting - principally: *Does it work with any other instruments?*

As well, some aspects could be implemented to make the simulation more human-like:

- Correlation between time delays, both auto-correlation and correlation between instruments. On this point, Markov chains might be an interesting study.

- Pitch deviation.

Likewise for the localisation experiment, questions remain unanswered, or arose on the way.

- Is a peripheral flute easier to localise than a central one?
- Would it be easier to localise a distinctive flute if it were moving?
- To what extent does the instrument affect the test results? Would it be easier to localise an instrument with more low-frequency components?

Performing the same experiment by adjusting the pitch of one instrument instead of tilting its spectrum could also be an interesting study.

References

[1] Daniel Kahlin and Sten Ternstrom. The chorus effect revisited: Experiments in frequency-domain analysis and simulation of ensemble sounds, 1999.

[2] Jens Blauert. *Spatial Hearing - Revised Edition: The Psychophysics of Human Localization.* MIT Press, 1996.

[3] Dale Purves et al. *Neuroscience.* Sinnauer Assiciates, Inc., 1997.

[4] J.C. Middlebrooks and D.M. Green. Sound localization by human listeners. *Annual Review of Psychology*, 1991.

[5] Durand R. Begault. *3-D sound for Virtual Reality and Multimedia.* AP Professional, 1994.

[6] Peter Svensson. 3d sound and multimedia: Lecture notes.

[7] Asbjørn Krokstad. *Akustikk for ingeniører.* Institutt for teleteknikk, NTNU, 1999.

[8] A.D. Musicant and R.A. Butler. Influence of monaural spectral cues on binaural localization. *The Journal of the Acoustical Society of America*, 77:202, 1985.

[9] Jan Terje Kvaløy and Håkon Tjelmeland. *Tabeller og formler i statistikk, 2. edition.* Tapir akademisk forlag, 2000.

A Enclosed files

Attached data files:

1. Reference flute quintet (Widor), 2-channel stereo, wave format
2. Simulated flute quintet (Widor), 2-channel stereo, wave format
3. Listening test 2 (Bjørnstad)
   4 dB tilt, mono, wave format
   6 dB tilt, mono, wave format
   8 dB tilt, mono, wave format
   10 dB tilt, mono, wave format
   12 dB tilt, mono, wave format
   refbj.mat: Reference quintet, 5 channel, MATLAB data file
   playxdbtilt.m: MATLAB script to run listening test 2
4. Source code: Ensemble simulation
   main.m
   timedelay.m
   spectilt.m
   ampvar.m
   decorrIIR.m
   adj.m
   play.m
   stereotize.m
5. Source code: Graphical equaliser
   ecg.m
   ec.m
   ex.m
   frc.m
   riple.m
   tfinv.m
   zcalc.m

Note that the localisation test requires a setup of minimum 5 speakers to be run. If conditions are satisfied, run the file ”playxdbtilt.m” in MATLAB to reproduce the test (playxdbtilt.m was not the original file used, but it should work similarly).

B Source code for the ensemble simulation

main.m

```
% main.m
%
% Simulation of an ensemble based on one single instrument.
% Initialisation file.
%
% Torbjørn Ursin, 2007

clear all
global fs noS duration t operations verbosityLevel wb

% Adjustable parameters
noS=2;                   % Number of speakers
instrument='fast4';      % Instrument to play
verbosityLevel=1;        % 0 for no figures, 2 for all figures
aif='asio';              % Audio interface ('win' or 'asio')
dev=1;                   % Device no. (pa_wavplay for an overview)

%%
[snd,fs]=wavread(['instruments/' instrument]);
duration=length(snd)/fs;

if min(size(snd))==2
    snd=.5*(snd(:,1)+snd(:,2));
end
snd=snd(1:round(duration*fs));

orig=snd;
rand('state',0);
randn('state',0);

operations=5;
wb=waitbar(0,'Manipulating timing...');
snd=timedelay3(snd);
snd=spectilt(snd);
snd=ampvar(snd);
snd=decorrIIR(snd,rand(128,2*noS));
snd=adj(snd);
waitbar(1,wb,'Complete.')
```

```matlab
close(wb)

%%
orig=[orig;zeros(length(snd)-length(orig),1)];
if verbosityLevel==1||verbosityLevel==2
    colors=['r';'g';'b';'c';'m';'y';'k';...
            'r';'g';'b';'c';'m';'y';'k';...
            'r';'g';'b';'c';'m';'y';'k'];
    figure(1)
    subplot 211
    t=linspace(0,duration,length(orig));
    plot(t,orig,'k')
    ylim([-1.2*max(orig) 1.2*max(orig)])
    title('Original sound file')
    drawnow
end

%%
% save simulfast snd
% play(snd,dev,aif)
stereotize(snd,fs,dev,aif)
% load sndwi;
% stereotize(snd,fs,dev,aif)

```

timedelay.m

```
function out=timedelay(in)
% timedelay.m time-shifts sounds in an input sound file.
%
% Torbjørn Ursin, 2007

global noS fs duration operations verbosityLevel wb

windowWidth=.6; % Seconds
overlap=.4; % Seconds
maxDeviation=130; % Milliseconds

out=zeros(length(in)+windowWidth*fs-overlap*fs,noS);
% out=zeros(size(in));
window=tukeywin(windowWidth*fs,.25);
for x=1:noS
    waitbar(1/operations/noS*x,wb,'Manipulating timing...');
        for y=1:windowWidth*fs-overlap*fs:length(in)
            devtn=floor(maxDeviation*fs/1000*rand(1)-...
                maxDeviation*fs/1000/2);
            if y+devtn<0
                devtn=0;
            end
            if y+windowWidth*fs+devtn>length(in)
                out(y:length(in),x)=out(y:length(in),x)+...
                    in(y:length(in)).*window(1:length(in)-y+1);
            else
                out(y:y+windowWidth*fs-1,x)=...
                    out(y:y+windowWidth*fs-1,x)+...
                    in(y+devtn:y+windowWidth*fs+devtn-1).*...
                    window;
            end
        end
    out(:,x)=out(:,x)/max(max(out(:,x)))*max(max(in));
    if verbosityLevel==2
        t=linspace(0,duration,length(out(:,x)));
        figure(2)
        subplot(noS,operations,(x-1)*operations+1)
        plot(hann(windowWidth*fs))
        figure(3)
        subplot(noS,operations,(x-1)*operations+1)
        plot(t,out(:,x))
        drawnow
    end
end
```

spectilt.m

```
function out=spectilt(in)
% spectilt.m tilts the spectrum of an input sound file by
% applying a linear filter.
%
% Torbjørn Ursin, 2007

global fs noS t duration operations verbosityLevel wb

maxDeviation=1; % Maximum deviation upwards or downwards

n=128;
f=linspace(0,1,fs/2);
out=[];
for x=1:noS
    waitbar(1/operations+1/operations/noS*x,wb,...
        'Manipulating spectrum...');
    deviation=2*maxDeviation*rand(1)-maxDeviation;
    [mag fund]=max(abs(fft(in(:,x),fs)));
    m=[ones(1,fund) linspace(1,1+deviation,length(f)-fund)];
    b=fir2(n,f,m);
    out(:,x)=fftfilt(b,in(:,x));
    if verbosityLevel==2
        figure(2)
        subplot(noS,operations,(x-1)*operations+2)
        semilogy(f*(fs/2),m)
        axis([0 fs/2 0 max(m)+.1])
        figure(3)
        subplot(noS,operations,(x-1)*operations+2)
        t=linspace(0,duration,length(out(:,x)));
        plot(t,out(:,x))
        figure(4)
        loglog(f*fs/2,m)
        axis([0 fs/2 0 max(m)+.1])
    end
end

```

ampvar.m

```
function out=ampvar(in)
% ampvar.m manipulates the amplitude of an input sound file.
%
% Torbjørn Ursin, 2007

global noS fs duration operations verbosityLevel wb
hops=floor(length(in)/fs);
amp=.7;

t=linspace(0,duration,length(in));
out=[];
for x=1:noS
    waitbar(2/operations+1/operations/noS*x,wb,...
        'Manipulating intensity...');
    av=amp*rand(hops,1)-amp/2;
    av=resample(av,length(in),hops);
    av=av+1;
    out(:,x)=av.*in(:,x);
%     clear('av')
%     load('avm');
%     avmtmp=avm(:,1:noS);
%     clear('avm');
%     out(:,x)=avmtmp(:,x).*in(:,x);
    if verbosityLevel==2
        figure(2)
        subplot(noS,operations,(x-1)*operations+3)
        plot(t,av)
        figure(3)
        subplot(noS,operations,(x-1)*operations+3)
        plot(t,out(:,x))
    end
end
%
% for x=1:noS
%     av=amp*rand(hops,1)-amp/2;
%     av=resample(av,length(in),hops);
%     av=av+1;
%     avm(:,x)=av;
%     disp(x)
% end
% save('avm')
```

```
%
% %%
% load('avm');
% avmtmp=avm(:,1:noS);
% clear('avm');
%
% out=avmtmp.*in;
```

decorrIIR.m

```
function y=decorrIIR(y,Noise)
%input is a signal y with size [length,nchann] and Noise of size
%[filterlength,nchann] wich normally distributed between 0 and 1
%The channels of the noise matrix must be twice the number of y
%channels. The filtering will lead to a sample delay that is twice the
%length of filterlength
% -  Audun Solvang 2006 -
%   Uses the functions POL2CART, CONJ, POLY, FLIPLR, FILTER.
%
% Modified: Torbjørn Ursin, 2007

global operations noS verbosityLevel wb
[lengde,nchann]=size(y);
%the radius of the poles
N1=Noise(:,1:nchann);
%the phase of the poles
N2=Noise(:,nchann+1:2*nchann);
%Ensures that none of the poles will be at the circle of unity. The
%tweakfactor may be varied
N1=N1*0.9;
%The phase of the poles must be in the range -pi to pi
N2=(N2-0.5)*2*pi;
%Generate the radius for the zeros
N11=N1.^-1;

%N is the order of the filter:
[N,nchann]=size(N2);
%Transforms the polar notation of zeros to cartesian notation
[re,im]=pol2cart(N2,N11);
compz=re+j*im;
%filters channel by channel
for jj=1:nchann
    %run the filtering in cascade.
    for ii=1:N
        % creates the complex conjugate
        medz=[compz(ii,jj);conj(compz(ii,jj))];
        % find the polynomial that corresponds to the filter
        % coefficients of the given zero
        b=poly(medz);
        %find the corresponding filtercoeffiecients for the pole
        a=fliplr(b);
```

```
            y(:,jj)=filter(b,a,y(:,jj));
            waitbar(3/operations+.5/operations/nchann*(jj-1) ...
                    +.5/operations/N*ii,wb,'Manipulating phase...');
        end
    if verbosityLevel==2
        figure(2)
        subplot(noS,operations,(jj-1)*operations+4)
        plot(b)
        figure(3)
        subplot(noS,operations,(jj-1)*operations+4)
        plot(y(:,jj))
    end
    end
```

adj.m

```
function out=adj(in);
% adj.m Rearranges the pa_wavplay input file to put sounds in front.
% Example: Channels 1,2,3,4 become channels 15,16,1,2.
%
% Torbjørn Ursin, 2007

global wb operations

noS=min(size(in));
out=[];
mv=floor(noS/2);
for x=1:noS
    if(operations<0)
        waitbar(4/operations+1/operations/noS*x,wb,...
            'Adjusting position...');
    end
    if x-mv<1
        out(:,16-mv+x)=in(:,x);
    else
        out(:,x-mv)=in(:,x);
    end
end
```

play.m

```
function play(in,dev,aif)
% play.m plays an input sound file through an arbitrary number of
% speakers.
%
%  dev - is the device id to use for output.
%  aif - Audio interface. Determines which sound driver to use
%        'win'      Windows Multimedia Device
%        'dx'       DirectX DirectSound driver
%        'asio'     ASIO Driver
%  Input arguments are to be set in main.m.
%
% Torbjørn Ursin, 2007

global verbosityLevel fs noS

colors=['r';'g';'b';'c';'m';'y';'k';...
    'r';'g';'b';'c';'m';'y';'k';...
    'r';'g';'b';'c';'m';'y';'k'];
t=linspace(0,length(in)/fs,length(in));
if strcmp(aif,'win')
%  wavplay(in,fs)
  for x=1:noS
    if verbosityLevel==1||verbosityLevel==2
      figure(1)
      hold on
      subplot 212
      plot(t,in(:,x),colors(x))
      ylim([-1.2*max(max(abs(in))) 1.2*max(max(abs(in)))])
      title('Manipulated sound file')
      drawnow
    end
    wavplay(in(:,x),fs)
  end
elseif strcmp(aif,'asio')
  if verbosityLevel==1||verbosityLevel==2
    figure(1)
    hold on
    subplot 212
    plot(t,in)
    ylim([-1.2*max(max(abs(in))) 1.2*max(max(abs(in)))])
    title('Manipulated sound file')
```

```
        drawnow
    end
    pa_wavplay(in,fs,dev,aif)
else
    disp('Audio interface error. Specify win or asio.')
end
```

stereotize.m

```
function stereotize(in,fs,dev,aif)
% stereotize.m plays an input sound file through two speakers,
% panning input channels evenly between the speakers.
%
%  fs  - sampling frequency
%  dev - is the device id to use for output.
%  aif - Audio interface. Determines which sound driver to use
%          'win'      Windows Multimedia Device
%          'dx'       DirectX DirectSound driver
%          'asio'     ASIO Driver
%  Input arguments are to be set in main.m.
%
% Torbjørn Ursin, 2007

ch=min(size(in));
out=zeros(length(in),16);
for x=1:ch
    out(:,16)=out(:,16)+in(:,x)*(ch-x)/(ch-1);
    out(:,2)=out(:,2)+in(:,x)*(x-1)/(ch-1);
end
wavwrite(out,fs,'SimulatedQuintet')
pa_wavplay(out,fs,dev,aif);
```

C Source code for the graphic equaliser

ecg.m

```
% ECG, ecualizador grafico
%
%
%
% espen moberg, 2006
%
% Modified: Torbjørn Ursin, 2007

function ecg (va)
global s p q hax hgrf hqgrf hsli htxt hqsli hqtxt hriple rango...
    frecs bandas
input='instruments\slow4';
% input='instruments\fast4';

if nargin < 1,
    va = 'dn1012';
    s = tf('s');
end

switch (va(1))
    case 'd' % inicializar valores y dibujar la ventana,
             % el programa
        % valores
        bandas = eval(va(3:4));
        rango = eval(va(5:end));
        switch bandas
            case 31
                frecs = [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];
            otherwise
                frecs = [31.5 63 125 250 500 1000 2000 4000 ...
                    8000 16000];
        end
        bandas = length(frecs);
        p = zeros(1,bandas); % matriz con los valores del
                             % los pots (en dB)
        q = 1.6;

        %ventana
        close all
        figure;

        % grafico principal
        ga = 0.6; % ancha del graf
        axpos = [0.1 0.55 ga*1.1 0.4];
        hax = axes('position', axpos); % handle de los axes
        h_min = ec(zeros(1, bandas), frecs, 0.5);
        h_max = ec(ones(1, bandas), frecs, 0.5);
        [h f] = ec(p/(2*rango)+0.5, frecs, q);
        hgrf=plot(f,abs(h)); % handle del plot
        axis(hax,[min(f) max(f) min(abs(h_min))*...textcolorcomment
            0.9 max(abs(h_max))*1.1]);
        set(hgrf, 'parent', hax);
        set(hax, 'xscale', 'log');
        set(hax, 'yscale', 'log');
        set(hax, 'xtick', frecs);
        yt = sort([1 max(abs(h_max)) min(abs(h_min))...
                   10^(rango/10) 10^(-rango/10)]);
        set(hax, 'ytick', yt);
        set(hax, 'yticklabel', [num2str((10 * log10(yt))',3)...
                                repmat(' dB',size(yt,2),1)]);
        set(hax, 'ygrid', 'on');
        set(hax, 'yminorgrid', 'off');
        set(hax, 'xgrid', 'on');
        set(hax, 'xminorgrid', 'off');
        set(hax, 'fontsize', 10);

        % grafico Q
        hqax = axes('position', [0.8 0.75 0.15 0.15]);
        [z f] = zcalc(q,1000);
        hqgrf = plot(f,1./abs(z));
        set(hqax, 'xscale', 'log');
        set(hqax, 'yscale', 'log');
        axis(hqax,[min(f) max(f) ...
                   min(1./abs(z))*0.9 max(abs(1./z))*1.1]);
        set(hqax, 'ytick', []);
        set(hqax, 'xtick', []);
        set(hqax, 'fontsize', 10)

```

```
        % sliders
        for i = 1:bandas
            hsli(i) = uicontrol('style', 'slider', ...
                'value', (round(100*p(i)))/100, ...
                'min', -rango, 'max', rango, ...
                'sliderstep', [round(1/(2*rango)) ...
                round(6/(2*rango))], ...
                'unit', 'normalized', ...
                'position', ...
                    [0.13+(ga/(bandas-1)*(i-1))-ga/bandas/2 ...
                    0.1 ga/bandas 0.38], ...
                'tooltipstring', ...
                    [num2str((round(p(i)*100))/100) ' dB'], ...
                'callback', ['ecg ss' num2str(i)], ...
                'fontsize', 10 );
            htxt(i) = uicontrol('style', 'text', ...
                'string', (round(100*p(i)))/100, ...
                'backgroundcolor', [0.8 0.8 0.8], ...
                 'unit', 'normalized', ...
                 'position', ...
                     [0.13+(ga/(bandas-1)*(i-1))-ga/bandas/2 0 ...
                     ga/bandas 0.1], ...
                 'fontsize', 10);
         end

         % q-slider
         hqsli = uicontrol('style', 'slider', ...
             'value', log10(q), ...
             'min', log10(0.1), 'max', log10(10), ...
             'sliderstep', [0.005 0.1], ...
             'unit', 'normalized', ...
             'position', [0.8 0.7 0.15 0.04], ...
             'tooltipstring', ['Q = ' num2str(q)], ...
             'callback', 'ecg qs', ...
             'fontsize', 10 );

         % q-texto
         hqtxt = uicontrol('style', 'text', ...
             'string', ['Q = ' num2str(q)], ...
             'backgroundcolor', [0.8 0.8 0.8], ...
             'unit', 'normalized', ...
             'position', [0.8 0.65 0.15 0.04], ...
             'fontsize', 10);

         % ripple-texto
         hriple = uicontrol('style', 'text', ...
             'string', 'Ripple = ', ...
             'backgroundcolor', [0.8 0.8 0.8], ...
             'unit', 'normalized', ...
             'position', [0.8 0.6 0.15 0.04], ...
             'fontsize', 10);

         % salto-botones
         alts = [0.4 0.27 0.14];
         for i=1:3
             hsalto(i) = uicontrol('style', 'pushbutton', ...
                 'string', '->', ...
                 'backgroundcolor', [0.8 0.8 0.8], ...
                 'unit', 'normalized', ...
                 'position', [0.05 alts(i) 0.04 0.04], ...
                 'callback', ['ecg Sn' num2str(i)], ...
                 'fontsize', 10 );
         end

         % escuchar-boton
         heschuchar = uicontrol('style', 'pushbutton', ...
             'string', 'OK', ...
             'backgroundcolor', [0.8 0.8 0.8], ...
             'unit', 'normalized', ...
             'position', [0.8 0.1 0.15 0.04], ...
             'callback', 'ecg en', ...
             'fontsize', 10, ...
             'fontweight', 'b' );

         % elegir entre 6 y 12 dB
         dbvalores = [6 12];
         dbpos = [0.8 0.87];
         for i=1:length(dbvalores)
             hdbel(i) = uicontrol('style', 'radiobutton', ...
                 'string', [num2str(dbvalores(i)) 'dB'], ...
                 'value', (dbvalores(i) == rango), ...
                 'backgroundcolor', [0.8 0.8 0.8], ...
                 'unit', 'normalized', ...
                 'position', [dbpos(i) 0.92 0.1 0.04], ...
                 'callback', ['ecg dn' num2str(bandas) ...
                                 num2str(dbvalores(i))], ...
                 'fontsize', 8 );
         end

     case 'q' % cambiar? el valor de q
         q=10^(get(hqsli,'value'));
         set(hqsli, 'tooltipstring', ['Q = ' num2str(q)]);
         set(hqtxt, 'string', ['Q = ' num2str(q)]);
         [z f] = zcalc(q,1000);
         set(hqgrf,'xdata',f, 'ydata',abs(1./z));
     case 's' % cambiar? un slider
         ind = eval(va(3:end));
         p(ind)=get(hsli(ind),'value');
         set(hsli(ind), 'tooltipstring', [num2str(p(ind)) ' dB']);
         set(htxt(ind), 'string', p(ind));
     case 'S' % cambiar todos los sliders
         pres(1,:) = ones(1,bandas);
         pres(2,:) = 0.5 * ones(1,bandas);
         pres(3,:) = zeros(1,bandas);
         pres(4,:) = 0.5 * ones(1,bandas); pres(4,1) = 0;
             pres(4,2) = 1;
         pres(5,:) = 0.5 * ones(1,bandas);
             pres(5,ceil(bandas/2)+1) = 1;
         pres(6,:) = 0.5 * ones(1,bandas);
             pres(6,ceil(bandas/2)+1) = 0;
         pres(7,:) = zeros(1,bandas);
             pres(7,ceil(bandas/2)+1) = 1;
         pres(8,:) = ones(1,bandas);
             pres(8,ceil(bandas/2)+1) = 0;
         switch bandas
             case 10
             case 31
         end
         pre = rango * 2 * (pres(eval(va(3:end)),:) - 0.5);
         for ind=1:bandas
             set(hsli(ind), 'value', pre(ind));
             ecg(['sn' num2str(ind)]);
         end
         ecg('as');
     case 'e' % escuchar
         [h f H] = ec(p/(2*rango)+0.5, frecs, q, rango);
         [x,fs]=wavread(input);
 %         [x,fs]=wavread('utopia.wav');
         t = 0:1/fs:length(x)/fs-1/fs;
         y = lsim(H,x,t);
 %         sound(y/4,fs);
         eqout=y/4;
         save eqoutbj3 eqout
 end
 if (va(2) ~= 'n')
     [h f] = ec(p/(2*rango)+...
         0.5,frecs,q,rango); % dar el vector con valores
                             % entre 0 y 1
 % [h f] = ec(sign(p).*real(log10(abs(p/rango*9)+1))/2+...
 % 0.5,frecs,q,rango);...
                             % dar el vector con valores
                             % entre 0 y 1
     f_intersant = [20 125 250 500 700 1000 2000 3993 ...
                   7973 16000 20000];
     for counter=1:11
         i = find(f>=f_intersant(counter),1);
         disp([num2str(round(f(i))) ': ' ...
             num2str(10*log10(abs(h(i)))) ])
     end
     disp(' ')
 %     f(i) % nesten internast
     set(hgrf,'xdata',f, 'ydata',abs(h));
     set(hriple,'string',['Ripple: ' ...
         num2str(10*log(riple(h, f, frecs))) ' dB']);
 end
```

ec.m

```
function [h, f, H] = ec(a, frecs, q, rango, acura)
if nargin < 1, a = [0.0 1.0 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5]; end
% if nargin < 1, a = [0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]; end
if nargin < 2, frecs = [31.5 63 125 250 500 1000 ...
        2000 4000 8000 16000]; end
if nargin < 3, q = 1; end
if nargin < 4, rango = 12; end
if nargin < 5, acura = 1000; end
if size(frecs) == size(a)
    bandas = length(a);
else
    error(['dimension de a (' num2str(length(a)) ...
        ') no es igual a la dimension de frecs (' ...
        num2str(length(frecs)) ')']);
end
global s;
if nargout < 1,
    s = tf('s');
end
gb = 15e6;
rs = 100;
k = 20;
% frecs = logspace(log10(31.5),log10(16128),bandas)
f = logspace(log10(20), log10(20000), acura);
for i = 1:length(frecs)
    f0 = frecs(i);
    [z(i,:), f, Z(i,:), c1(i), c2(i)] = ...
        zcalc(q, f0, f, rs, k, 1);
end

aa = a';
a = repmat(a',[1,1000]);

switch (rango)
    case 12
        r = 20000;
        rp = 20000;
    case 6
        r = 2000;
        rp = 2000;
end
```

```

% la curva
h = (1 + sum(a .* rp ./ (z + a .* (1 - a) * r))) ./ ...
    (1 + sum((1 - a) .* rp ./ (z + a .* (1 - a) .* rp)));

% v2ms = ruido(a,f,r,rp,rs,k,c1,c2,z)

% funcion transferencia...
if (nargout > 2) | (nargin < 1) ,
    h1 = diag(aa * rp) * tfinv(Z + aa .* (1 - aa) * r);
    h2 = diag((1 - aa) * rp) * tfinv(Z + aa .* (1 - aa) .* rp);
    hh1 = tf(0);
    hh2 = tf(0);
    for i = 1:length(frecs)
        hh1 = hh1 + h1(i);
        hh2 = hh2 + h2(i);
    end

    H = (1 + hh1) * tfinv(1 + hh2);

    [xxx,fff] = bode(H,f*2*pi);
    xx = 1:size(xxx,3);
    ff = 1:size(fff,3);
    xx(:) = xxx(1,1,:);
    ff(:) = fff(1,1,:);
    h = xx;
end

if nargout < 1,
    close all;
    figure(1);
    ax = axes;
    grf=plot(f,abs(h),'b');
    hold on;
    plot(f,abs(xx),'k');
    hold off;
    set(ax, 'xscale', 'log');
    set(ax, 'yscale', 'log');
%     axis(ax,[min(f) max(f) min(min(abs(h))) max(max(abs(h)))]);
    figure(2);
    bode(H, f*2*pi)
    h = 0;
    f = 0;
```

```
else
    return
end
```

ex.m

```
%Equalizer
%Vikas Sahdev
%Rajesh Samudrala
%Rajani Sadasivam
%
[x,fs]=wavread('utopia.wav');
Wn = .20;
N = 62;
%These are the gains on each of the 3 bands
gLP = 0.4;
gBP = 1.5;
gHP = 1.5;

LP = fir1(N,Wn);
Wn1 = [.20, .50];
BP = fir1(N,Wn1);
Wn2 = .50;
HP = fir1(N,Wn2,'high');
figure(1)
freqz(LP);
figure(2);
freqz(BP);
figure(3);
freqz(HP);
y1 = conv(LP,x);
y2 = conv(BP,x);
y3 = conv(HP,x);
yA= gHP * y3;
wavwrite(yA,fs,'Equalizer3');
yB= gLP * y1;
wavwrite(yB,fs,'Equalizer1');
yC= gBP * y2;
wavwrite(yC,fs,'Equalizer2');
yD = yA + yB + yC;
wavwrite(yD,fs,'Equalizer4');
```

frc.m

```
function [Y, f] = frc(x, fs, c)
if nargin < 3, c = 'b'; end
% T = 2;  % duracion
% fs = 500;
% x = rand(1,T*fs);
f = linspace(0,fs/2,fs/2);
Y = fft(x,fs);
Pyy = Y.* conj(Y);
Y = Y(5:length(Pyy)/2+4);
plot(f,abs(Y),c)
% % f = 1000*(0:256)/512;
% % plot(f,Pyy(1:257))
% title('Frequency content of y')
% xlabel('frequency (Hz)')
```

riple.m

```
function r = riple(x, y, y_picos)

if nargin < 1, % para testing...
    [x y] = ec(ones(1,10));
    y_picos = [31.5 63 125 250 500 1000 2000 4000 8000 16000];
end

x=abs(x);
bandas = length(y_picos);
r = [];
p = [];
% encontrar indices para los picos
for i = 1:bandas
    il = find((y_picos(i)*0.95 < y & y < y_picos(i)*1.05));
    yl = y(il);
    xl = x(il);
    [xl_max il_max] = max(xl);
    il_max = il(1) + il_max;
    p = [p il_max];
end

% la difencia entre los picos y el minimo entre los picos
for i = 2:(bandas-2)
    r(i) = (x(p(i)) + x(p(i+1))) / 2 / min(x(p(i):p(i+1)));
end

r = max(r);


```

tfinv.m

```
function hinv = tfinv(h)
[a b] = tfdata(h);
hinv = tf(b,a);
return
```

zcalc.m

```
function [z, f, Z, c1, c2] = zcalc(q, f0, f, rs, k, metodo)
if nargin < 1, q = 1; end
if nargin < 2, f0 = 500; end
if nargin < 3, f = logspace(log10(20), log10(20000)); end
if nargin < 4, rs = 500; end
if nargin < 5, k = 20; end
if nargin < 6, metodo = 4; end

global s;
if nargout < 1,
    s = tf(’s’);
end

gb = 15e6;
w0 = 2 * pi * f0;
w = 2 * pi * f;
switch (metodo)
case 1
    r1 = k * rs;
    r2 = rs;
    c1 = 1 / (q * rs * w0);
    c2 = q / (k * rs * w0);
    z = 1 ./ (c1 * j .* w) + r2 + r1 * r2 * c2 * j .* w;
    Z = 1 / (c1 * s) + r2 + r1 * r2 * c2 * s;
case 2
    r1 = k * rs;
    r2 = rs;
    c1 = 1 / (q * rs * w0);
    c2 = q / (k * rs * w0);
    z = 1./(c1*j.*w)+(r2.*(1 + r1 * c2 * j .* w).*...
        (1 + j .* w / gb) ./ (1 + (r2 * c2 + 1 / gb) * ...
        j .* w + (r1 + r2) * c2 / gb .* (j .* w).^2));
    Z = 1 / (c1 * s) + (r2 * (1 + r1 * c2 * s) * (1 + s / gb) ...
        / (1 + (r2 * c2 + 1 / gb) * s + (r1 + r2) * c2 / gb * s^2));
case 3
    r = rs / 2;
    c1 = 1 / (q * rs * w0);
    c2 = 4 * q / (rs * w0);
    z = 1 ./ (c1 * j .* w) + 2 * r + r^2 + r^2 * c2 * j .* w;
    Z = 1 / (c1 * s) + 2 * r + r^2 + r^2 * c2 * s;
case 4
```

```
    r = rs / 2;
    c1 = 1 / (q * rs * w0);
    c2 = 4 * q / (rs * w0);
    z = 1./(c1 * j .* w)+r.*(2 + r * c2 * j .* w) .* ...
        (1 + j .* w / gb) ./ (1 + j .* w / gb + r * c2 ...
        / gb .* (j .* w).^2);
    Z = 1 / (c1 * s) + r * (2 + r * c2 * s) * (1 + s / gb) / ...
        (1 + s / gb + r * c2 / gb * s^2);
end

if nargout < 1,
    close all;
    figure(1);
    ax = axes;
    grf=plot(f,abs(z));
    hold on;
    [xxx,fff] = bode(Z,f*2*pi)
    xx = 1:size(xxx,3);
    ff = 1:size(fff,3);
    xx(:) = xxx(1,1,:)
    ff(:) = fff(1,1,:)
    plot(f,abs(xx),'k');
    hold off;
    set(ax, 'xscale', 'log');
    set(ax, 'yscale', 'log');
    axis(ax,[min(f) max(f) min(min(abs(z))) max(max(abs(z)))]);
    h = 0;
    f = 0;
else
    return
end
```