INTO SOUND

On the practical analysis and processing of sound

Ton Wempe

Printed in The Netherlands

CONTENTS

Preface

As an electronics engineer having worked at the language department of the University of Amsterdam for many years, I assisted researchers and students in many aspects of processing of sound, such as various signal analysis methods and automation of analysis and manipulation handling procedures. A great deal of their needs for assistance fell into the category spectral analysis, a field which most language researchers experienced as 'difficult'. Generally, people with a background in linguistics have not had much training in mathematical and technical subjects. Sometimes, in the planning of a research, the focus might be shifted so as to avoid signal analysis techniques. Often, those analyses that could not be avoided will be carried out applying ´blindly´ the default parameter settings of a suitable program like "Praat" [2]. These may work perfectly in many cases but will need to be adapted when specific sound material is analyzed. Also, adaptations need to be made depending on the aim of the analysis. This lack of technical background might also cause serious errors in the interpretation of the results of the analyses. Language researchers from other countries than my own (The Netherlands) have told me similar stories, so it seems that this problem might be quite widespread.

There are numerous books about signal analysis, varying from simple introductions to highly advanced mathematical works. Why then this book when there are so many already? It seems that the books available are either highly theoretical and technical and thus only suited for people skilled in mathematics, or oversimplified so that the background is not explained clearly and cannot be understood sufficiently. This applies especially to researchers in the humanities who have not had an extensive training in mathematics, which limits a person's ability to understand most books on signal analysis. In addition, many books concentrate solely on digital signal processing and microprocessor applications, which would be a too limited area for the audience for which this book is intended.

In this book, therefore, I will try to explain the principles of manipulations, analyses and productions of sound that researchers who work in this area generally come across, rather than to try to cover all aspects of signal processing. The tools that will be used are first of all, the reader's own common sense, and secondly, the versatile free program "Praat" (see the **Introduction** for accessing and using Praat). The mathematics used here hardly exceed secondary school levels. Possibly, this kind of simplification might be at odds with the mathematical correctness now and then. However, because the purpose of this book is getting insight in the signal processing mechanisms, that is what should have the highest priority. As long as the mathematical incorrectness does not influence the general truth of the underlying principles, the complications of a strictly mathematical approach are avoided in favor of clarity.

The book focuses on speech research. However, the basic principles of analysis used are valid for all kinds of sound analyses, or more generally, 'low frequency' waves analysis. Therefore, sounds ranging from animal sounds, musical instruments, singing voices, machine noise and heart beats to earth quakes, can be analyzed using the same principles. This is provided that the microphone or transducer used, and the sound input of the computer and other electronics used are able to process the specific signals adequately, and that the above-mentioned parameter settings are adapted to the specific properties of these sounds.

Some properties of speech sounds and musical sounds are dealt with in some detail in special sections.

Acknowledgements

A great number of people have had some influence, directly or indirectly, on the contents of this book. During the many years I worked at the Institute of Phonetics of the University of Amsterdam I had the great opportunity to learn from all of my colleagues there. In particular, the discussions with Paul Boersma (who, being a great scientist, gave always highly valuable answers), David Weenink, Louis Pols and, in embryonic times, Toni Rietveld, have formed my interests greatly about the subject of sound signal analysis. I owe special thanks to Dirk Jan Vet who made a great job in correcting my mistakes in the previous versions of the book, all from his solid knowledge and ability to think very logically. Naturally, the remaining mistakes (which cannot be eliminated completely, due to the exponential behavior of its number function) are completely on my account.

The readers' group, led by Mirjam Ernestus of the University of Nijmegen, had the patience to point out a lot of passages in my explanations which were too cryptic formulated for 'normal people' like linguists and thus guided me to make many clarifying modifications.

In addition, the questions of all kind from the students of the Institute gave me some insight into the subjects which should get special attention.

The program Praat, created by Paul Boersma and David Weenink, provided me with an ideal tool for generation of sound examples, all signal processing and *all graphic pictures* (except two) I needed for the book.

Finally, I would express many thanks to Marije van Wieringen who corrected the many mistakes in my, Dutch biased, English writing.

Introduction

This book consists of two main parts:

Part A deals with the basic principles of the analysis and manipulation of sound. The focus is on explanation through logical reasoning. Why do signals behave as they do, and what are the consequences of the various methods of analysis and manipulation? This part of the book tries to answer your fundamental questions about many aspects of signal processing. This is the theoretical part of the book, but all of it is embedded in a practical context.

Generally, for most studies in *exact sciences* a next step in learning a subject can only be taken when the earlier steps are understood: the new step is built on those that came before. Of course, it would have been possible to organize the book in such a way that the reader might choose a topic at random. However, the book would then have ended up containing a huge number of references to other parts. Instead, I chose the first option: new sections are based on the knowledge of the earlier ones. Nevertheless, many references to an earlier section are included.

The section on *discrete time signal processing* (aspects of sampling signals for computer processing) is postponed to the end of part A. Although most books on signal processing deal with sampled signals from the beginning, I will discuss the properties and analysis of the analog signals prior to the discretization, in order to avoid confusion of the sampling effects with the inherent signal's properties.

Part B contains practical notes that are important for knowing how to make good sound recordings, avoiding signal distortions or background noise effects, choice of equipment, avoiding mistakes, etc. An important ingredient of this part B is the explanation of the effects of parameter settings in programs for the most commonly used signal transforms, like ‘'Praat''. Using the wrong settings for analyses can sometimes lead to inaccurate results! An especially tricky part is that the graphs may look accurate even when the output could still be misleading. It is important to know how to measure correctly.

Part B relies heavily on the information given in Part A and I would advise you to read Part A first. If you know enough already about basic signal processing principles you could skip this part, but, why miss all the fun?

Finally, in the Appendices a slightly more mathematical approach of some signal transforms and representations is presented. Here, the mathematics used is not very complicated but of course there are a few formulas there! For people with allergic reactions to formulas with e.g. integrals there is no need to suffer: they can skip them. Part A without appendices offers sufficient information to understand the basic

principles. For other people the appendices may serve as a step between the popularizing part of this book and other more mathematical books on the subject. These appendices may even offer you some more insights in the 'nature' of this sort of mathematics.

The program "Praat"

I would strongly advise you to download and use the free program 'Praat' [2] with which you can see and hear the effects described in the book. To run the demos in this book I made scripts for this program. The scripts are small programs that contain commands for Praat, and programming features, which make it possible to show effects of signal processing mechanisms and to play signals automatically.

The scripts can be found in the self-extracting file “scriptsIntoSound.exe” on the internet pages [www.fon.hum.uva.nl/wempe], or [www.auditon.nl]. It is highly probable that the demos in the book will work as well with Praat versions that are newer than the one I used for testing, because the program is always kept downward-compatible. Instead of downloading the newest version of the program from the Praat site mentioned in the reference you could download the version from one of my pages to ensure that the demos and descriptions in the book apply to the same Praat version that I used to test all scripts.

The installation of Praat on your computer is very simple. The self-extracting program “Praatxxxx winxx.zip” (x for version number and 32 or 64 bit) produces only one executable file (Praat.exe). You can put it on your desktop or anywhere else on your hard disk.

Praat is a versatile program for signal analysis. Originally designed as a basic speech analysis tool, it has been developed to a universal signal research platform where all kinds of analyses and manipulations can be made on signals, with the flexibility to adjust all kinds of parameters. The possibility to produce high quality graphs and the intuitive script language make this program tailor-made for researchers in this field.

There is no need to *learn* using Praat for running the demos in this book. It is used only as a tool together with the demo scripts. The basics for getting the scripts ready to run will be explained, however.

There are Praat versions for Windows, Mac and Linux but the demo scripts for this book are only tested in the Windows version. Possibly, for the other platforms some minor modifications of the scripts may be needed, and if you need help with that, there is a Praat user group (http://uk.groups.yahoo.com/group/praat-users/) where you can ask questions about scripting in general. There are some beginner’s manuals downloadable from Praat's home page (http://www.fon.hum.uva.nl/praat).

You do not need to understand the contents of the demo scripts. For people who are interested in how the scripts work: although the structures are quite simple and are clarified by explanatory comments, the scripts sometimes contain lines that will not be clear during the stage the reader has reached at the moment the demo is being used. That will not cause any problems as the purpose of the demos is the demonstration of the effects, not an explanation of the way the scripts work. (Nevertheless, it can be interesting to make your own scripts, of course!)

For listening to the demo sounds you should connect loudspeakers or headphones to your computer's sound output. For laptops or tablets I would advise to use headphones/earphones or external speakers instead of the built-in speakers because of the highly limited frequency range of these tiny built-in speakers. Make sure that the sound mixer (Windows Volume Control) or your special sound card software is adjusted for playing sounds from the Wave input. So, adjust the (Windows) Volume Control and Wave sliders to proper levels to avoid overloading, and be sure that eventual corresponding mute boxes are unmarked.

For using the demos, it is convenient to alter some of Praat's default settings, to avoid confusion by displaying more than what you are working on at that moment. This is taken care of in the special script for this purpose, called INIT.script. These new settings are preserved by Praat so that there is no need to adjust the settings after the next time you start the program. When the settings happen to be accidentally overruled, you could run the INIT script again so that the preferred settings are restored. The script also ensures the proper sound playing settings for all demo scripts (apart from the Windows settings mentioned above: they depend on the hardware used).

How to run scripts in Praat

If you have not downloaded and unpacked the scripts yet, please do it now and put them either in a new directory, or in the directory where Praat can be found. It is better not to use the desktop for all these scripts to avoid cluttering your screen with icons. When you have done that, run the Praat program. You will see two windows. The left window is called ‘Praat Objects’; the right window is called ‘Praat Picture’. In the Object window under ‘Praat’ select ‘Open Praat script...’. browse to the directory where you put all the demo scripts, select the demo script mentioned in the book and then click ’Open’. In the window that appears you will see the text of the script. Under its menu ‘Run’ select ‘Run Ctrl-R’.

This is the way to run all demo scripts. When in the book the text DEMOx.x appears (x.x stands for an index number) you should open the Praat script in the way described above and run it. Sometimes a window appears wherein you can enter some parameter values mentioned in the text or continue after the script has interrupted itself: many scripts produce a window where the user can select options or give data. When you no

longer need a certain script, you can close its window, but you can leave it alone as well: the program does not limit the number of scripts in memory. When the script has generated one or more windows you can close them in the normal way (by clicking the X in their upper right corner). However, you should not close the window named "Praat picture": if you want to get rid of the pictures drawn earlier you can erase them by selecting ‘Erase all’ under the ‘Edit’ menu of the ‘Praat picture’ window.

(The *sound editor* window in Praat is displayed slightly different from its default settings. The INIT script will automatically adjust the settings needed for our purposes. It has to be run only once.)

The scripts will produce certain objects in the Object window. When you quit Praat, the program will ask whether or not you want to save them. As the scripts produce the necessary objects anew when you run them there is no need to save them.

All this sounds more complicated than it is: during your actual Praat sessions you will find that it is all very straightforward.

Part A. Understanding basic signal analysis

1. The nature of sound

Most readers of this book probably know that sound in air consists of quickly varying local alterations of air pressure. All sound sources, i.e. all things or beings that make sound, do that by vibrating mechanical parts that move the air back and forth locally (or up and down or left and right or a combination of directions). These air movements cause local air pressure alterations which push and pull their neighboring areas and they push and pull the next areas in turn. In this way, waves of locally vibrating air travel in all directions, away from the sound source. There is a small delay before neighboring areas start to move: the propagation of sound in air takes time. At room temperature this speed is 340 m/s. Therefore, the sound wave propagates through air but the *air molecules themselves* remain where they are. Think about the waves that occur when you throw a stone in a pond: the waves travel along the surface of the water away from the position where the stone hit the water (the *source*) but the duck nearby moves only up and down. It remains at the same distance from the source. Of course, in air the waves travel in all three dimensions instead of two, as is the case with water waves.

Apart from air, sound can be propagated in all kinds of media: gases, liquids and solid matter. The propagation velocity depends on both the compressibility and the density of the medium. The higher the compressibility ("elasticity") the lower the propagation velocity will be. Also, the higher the density, the lower the propagation velocity will be. For helium gas, for example, the compressibility is quite the same as for air but the density is lower so that the propagation velocity is higher. For metal, the density is much higher than that of air but the compressibility is very, very much lower than that of air so that, as a result, the propagation velocity of sound in metal is about 5000 m/s which is 15 times the velocity in air. The propagation *velocity* is practically *not* dependent on wind; for sound propagation in air, the air is needed as a medium; the *type* of medium is not changed by wind, and the moving of the medium is much slower than the velocity of sound.

The spreading of the sound in all directions causes the sound **intensity**, which is expressed as power per unit of surface area, to decrease as the distance from the source increases: the **spreading loss**. If we think of a sphere with a certain radius r_1 around the sound source, all sound power is divided equally over the entire surface of the sphere so that everywhere on the sphere's surface the sound intensity has the same value.

The surface of the sphere with a radius r_1 is equal to $4\pi r_1^2$. If we alter the distance to r_2 the total surface is $4\pi r_2^2$. The intensity anywhere on the surface will have changed inversely proportional to the ratio of the *squares* of the radii. Consequently, if we define

one distance from the sound source as d_1 and one other distance as d_2, and the intensity found at distance d_1 as I_1 and the intensity at distance d_2 as I_2, we can write it as the following formula:

$$\frac{I_2}{I_1} = \frac{d_1^2}{d_2^2} \tag{1.1}$$

This formula is only valid for **point sources** which means that, theoretically, the sound source is supposed to be concentrated in one point. In practice, many sound sources can be considered as point sources if the size of the sound source is small compared to the measuring distances.

Another restriction on the validity of the formula is that the sound waves occur in the **free field** which means that there are no reflections by walls or furniture: all sound waves travel away from the sound source unhindered[1]. In a room the sound loss depends highly on the sound reflection, absorption and dimensions of the room. (For example, in a typical office room without special acoustic adaptations the intensity at 2 meters from the source can be about 2 or 3 times as high as the intensity of the same source in the free field at the same distance, due to the reflections in the room. When the microphone is placed near a wall this factor must roughly be doubled, caused by 'standing wave' effects, which will be discussed later.)

Naturally, when the sound waves travel away from the source there is some loss of energy during the takeover of the movements by adjacent parts of air: the **atmospheric absorption.** This type of sound loss must be distinguished from the spreading loss as described above. This atmospheric absorption is dependent on the *viscosity* (toughness) of the molecular structure of the air. Generally, this energy loss increases when the sound contains higher tones (or frequencies, which is explained in section 3.). This type of energy loss is only important in cases of high distances: higher than about 1 km or so. In cases of shorter distances, the formula 1.1 will suffice.

The intensity being a measure of power per unit of area, this is not what our microphones register. Inside the microphone its diaphragm is activated by the **sound pressure** (p). The sound pressure is commonly called **SPL** (Sound Pressure Level) and must be seen as a force acting on a surface. It is expressed in pascal. 1 pascal is defined as 1 Newton per square meter. This SPL is what our microphones transform into proportional electrical voltages. Do not mix up the sound pressure with the intensity: *pressure* is force per unit of surface area, measured in Newtons per square meter, and *intensity* is power per unit of surface area, measured in Watts per square meter. The intensity is the acoustic power that the sound source emits through the unit of area; the pressure is the *effect* of this emission. This means that the intensity is proportional to

[1] The term *free field* refers to a theoretical space without sound reflections. Out-of-doors, there always will be some sound reflections, so one should not take the term literally.

the SPL *squared*. In the box called **OHM's LAW** this relation is explained in a bit more detail.

Now our formula 1.1 can be rewritten for SPL values instead of intensity values:

$$\frac{p_2^2}{p_1^2} = \frac{d_1^2}{d_2^2} \quad \text{or} \quad \frac{p_2}{p_1} = \frac{d_1}{d_2} \tag{1.2}$$

If there is more than one sound source active at the same time, the air pressure at a particular position will be the result of all wave interactions. Therefore, at each particular point in time the voltage level of the microphone is proportional to the resulting air pressure level of all sound wave interactions at that point in time.

Our ear drums are put in motion in exactly the same way. Our brains are able to distinguish different sounds quite well from the resulting movements of the eardrums caused by different sound waves. Using both ears we can even hear the directions where the sounds come from, based on the little time delay differences of the sound wave

OHM'S LAW

The relation between pressure and intensity can be explained by using an analogy in electricity. You have *voltage* (the 230 volts of the mains outlet, for instance), and *current* (the amperes that flow through, say, a light bulb). If no lamp is connected with the outlet there is no current but there is a voltage. When a lamp is connected, the amount of current depends on the lamp type: a 100 Watt lamp causes a higher current than a 40 Watt lamp. The difference between the lamp types is determined by their *resistance*. Ohm's law defines the simple relation: the current is the voltage divided by the resistance ($I = V/R$). The resistance is a property of the lamp and has nothing to do with the outlet voltage. Consequently, $R = V/I$ and is a constant for a particular lamp which implies that, for example, doubling the voltage causes also doubling of the current.

Now, the amount of light that the lamp produces is proportional to the *power* (P, in Watts), which is the voltage multiplied with the current ($P = V \text{x} I$). Substituting I by V/R gives: $P = V^2/R$. The power is proportional to the *square* of the voltage!

Accordingly, this can be applied to the subject of sound. Just as, for a certain lamp, the voltage causes a proportional current, the acoustical pressure level causes a proportional particle velocity, for a certain medium. 'Ohm's law' then is: $v = p/z$, where v is the particle velocity, p the pressure level (SPL) and z the acoustic **impedance** of the medium (in analogy with current, voltage and resistance, respectively). Here, the power is called *intensity* (in Watts/m^2) and can be expressed as: $Int = vp = p^2/z$. The intensity is proportional to the square of the pressure!

For z the more general term *impedance* is used instead of resistance because of the possible frequency dependent phase shift between p and v. But that's another story!

paths between source and different ears, the *phase difference*. Likewise, the diaphragms of two microphones placed in similar positions as our ears, produce the complete information that our ears receive. Listening through earphones to recordings made in this manner gives the realistic sensation as if you are present in the room in which the sound was produced (*artificial head stereophony*). Even the awareness of sounds coming from behind or above is preserved. The reason why I am emphasizing this subject is that there are people who think that for more realistic spatial sound properties you need more microphones, which obviously is a mistake. On the contrary, simply

mixing more than two microphone signals into the stereo signal creates the opposite effect. Sometimes certain small signal delays are applied to the extra channels to get some artificial spatial sensation (‘surround sound’) but the real direction information is then disturbed.

In the practical part of the book we will go into the *acoustic* implications of the sound recording in some more detail, when we deal with the strategy to use microphones in rooms.

2. Decibel

The impression of sound intensity ('volume') is not linear: when the intensity of a sound, for example, doubles in magnitude, the impression is a certain increase of the volume. When the intensity doubles again, the *same increasing step* is perceived, and so on. Each doubling of the intensity we perceive as the same increase. Thus, the impression of the intensity behaves as if our hearing works logarithmically. This phenomenon should not be very surprising. Generally, our perception behaves not in an absolute but in a relative way. If you lift an object that weighs 1 kg, put it down and then lift an object of 1.5 kg, you will have some impression of the difference. If you do the same with objects of, say, 4 kg and 4.5 kg, the impression of the difference is much smaller. Only when the second set of objects weigh 4 kg and 6 kg respectively, the impression of the difference is the same compared with the first set. Apparently, for equal impressions of differences, the *percentage of increase* has to be constant. This general property of our perception is expressed by **Weber's law**:

$$\frac{\Delta I}{I} = C \tag{2.1}$$

where I is the physical value, ΔI the change of value which causes a certain impression and C a constant. Of course, the formula does not hold for very low (and very high) ΔI values. If the difference of the values is so small that it has reached the discrimination threshold or the *Just Notable Difference* (JND), the constant C becomes C_W: the *Weber's fraction*. Obviously, this JND too is a constant percentage of the absolute physical values.

For the representation of sound intensity, the logarithmic measuring unit **decibel** (dB) is used. In principle, it is nothing more than an exponential measuring unit for a *ratio* of two quantities. Its name implies that it is derived from the *bel* (named after Alexander Graham Bell) and, indeed, the decibel is 0.1 bel. One bel simply means a factor 10^1. Two bels mean 10^2 and so on. Likewise: -1 bel $= 10^{-1} = 0.1$. Then an increase by 1 decibel $= 10^{0.1}$, which is about a factor 1.26 (or 26 % increase). The other way goes accordingly: for example, a factor 1000 means 10^3; its log is 3 which is 30 decibels. A factor 1/1000 means 10^{-3} which has a log of -3 which is -30 decibels. In general:

$$\#dB = 10\log(INTratio) \tag{2.2}$$

$$INTratio = 10^{\#dB/10} \tag{2.3}$$

Now, this decibel is NOT the dB that is used for the expression of sound pressure! It is the norm to use the dB for the acoustical *intensity* which is the SPL *squared*. The acoustic intensity is the power per unit of area, and the SPL is the force per unit of area (see the box **OHM'S LAW**). As an example, when the SPL is raised by a factor 3, the intensity is raised by a factor 9. That implies that we have to *multiply* the number of dBs *by two* in formula 2.2 when dealing with SPLs because the dBs *refer to intensity*! In this way, the general formulas for SPL dBs become:

$$\#dB = 20\log(SPLratio) \tag{2.4}$$

$$SPLratio = 10^{\#dB/20} \tag{2.5}$$

For example, when the SPL ratio doubles, the number of dBs is increased by 20 log (2) which is about 6 dB, as log (2) = 0.30103.

So, when we want to express one *intensity* value with respect to a second *intensity* value, the formulas 2.2 and 2.3 should be used. When we want to express one sound *pressure* value with respect to a second sound *pressure* value, the formulas 2.4 and 2.5 should be used. This fact is emphasized because it seems that many people are unsure when to use the factor 10 and when the factor 20. It stems from the (historical) decision to interpret sound pressure in terms of the effect it has on the intensity. (The same applies to electrical voltages: the voltage ratio is also expressed in terms of the effect on the electrical power.)

Now what, for example, is meant by "the sound level is 80 dB"? As the dBs imply a *ratio* of SPL levels why are they used to define an *absolute* sound level? In other words, what level is 0 dB? The answer is that the average lowest hearing threshold is used as a reference. The weakest sound that is just noticeable for the average person is 20 µPa (micropascal: a millionth of a pascal) and defined as 0 dB. So, according to formula 2.5, 80 dB means a factor $10^{(80/20)} = 10000$. The SPL then is 10000 x 0.00002 Pa = 0.2 Pa. For indication of this reference, dB_{SPL} is sometimes used.

The strongest SPL that people can bear is about 120 dB which corresponds to 20 Pa. Levels like that cause pain and can damage the nerve cells of the inner ear. We may conclude from this that our ears can manage a pressure range ratio of $10^6 = 1$ million! As you may know, the atmospheric air pressure is about 10^5 Pa. So, the atmospheric air pressure is $10^5/20 = 5000$ times as high as the highest sound pressure we can bear! Luckily this atmospheric pressure is present at both sides of our eardrums...

Another measuring unit of sound levels is quite common: the **dB_A**. Our ear's sensitivity is dependent on frequency. An attempt to take this into account is the application of a generalized frequency dependent function: the '**A-weighting**' of sound. The dB_A values are adapted to this standardized frequency dependence. In Part B we will learn more about this method (and its limitations) to compensate for our ear's properties.

A consequence of logarithmic measuring is that if there is no sound at all (zero *SPL*) this *cannot* be expressed on a log scale. In that case the *SPLratio* is 0/20 µPa which is 0. And log (0) amounts to minus infinity. Therefore, please do not make the mistake that zero sound pressure corresponds with 0 dB! In practice, in our atmosphere there is always sound with *some* level, no matter how small, so that it always can be expressed in dB.

3. Waveform, frequency, spectrum

When you give a swing a single push from its position at rest, it moves forward and backward in a smooth, periodical way. It goes on like that, while the greatest deviations from its rest position slowly decrease until it finally comes to rest again. Were it not for the air resistance and the friction of the bending rope or the hinges, the movement would go on forever. In that case it would move exactly like a **sine wave** (provided that the deviations remain small). This type of movement forms the most basic periodic "vibration" there is. When a graph is plotted of the swing movement as a function of time, a graph like the example in fig. 3.1 will emerge, called the **waveform**. The maximum absolute value of this sine function is the **amplitude**.

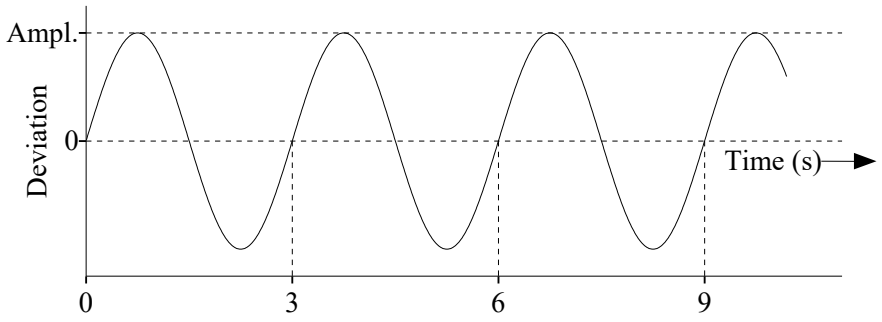


Fig. 3.1. Movement of a swing as a function of time.

If the air pressure is changing according to this sine waveform (although at a much faster pace), a so-called pure tone can be heard. Run DEMO3.1.script in the program Praat to play an example of a tone and displaying its graph. (Switch on your sound playing device or put on your headphones.)

As already mentioned, the signal which the microphone registers is the SPL of the sound. The vertical amplitude axis, therefore, refers to the SPL.

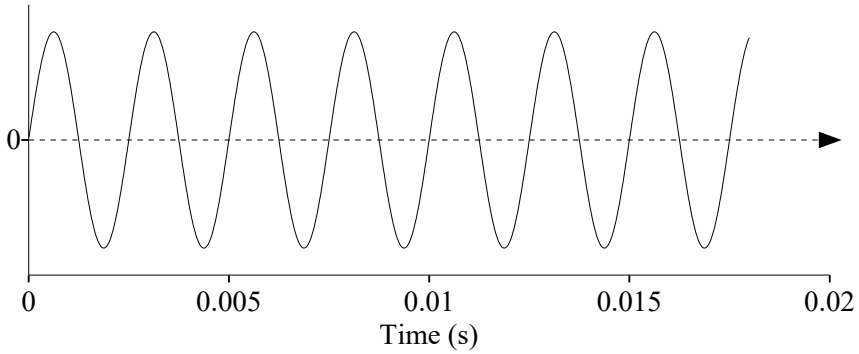


Fig. 3.2. Waveform of 18 ms of a sine wave.

In this example there are 400 complete cycles per second. Its **frequency** is 400 hertz. This can be read from the waveform as follows: the time of one complete "cycle" (= the time between two similar positions in the waveform, let's say two corresponding "zero crossings"), is 2.5 milliseconds (ms), so the number of cycles per second is 1/0.0025 or 1000/2.5 which is 400 hertz. (See also Fig. 3.2.)

In section 1 the propagation of sound in air was mentioned, being 340 m/s. So, one complete period of this 400 Hz tone spreads out in space over a length of 0.0025 x 340 m = 85 cm. Therefore, a tone of 400 Hz has a **wavelength** of 85 cm. Because our ears remain in the same position, we do not perceive these wavelengths, only the air pressure alterations. (In fact, if we would move towards the sound source, this would increase the speed with which the pressure alterations reach our ears and this increase of the propagation speed simulates a shorter wavelength and, therefore, a higher tone. The reverse, if we would move away from the sound source, the tone would be heard at a lower frequency. This is the well-known *doppler effect.*) The relation between wavelength, propagation speed and frequency is expressed by the formula:

$$\lambda = \frac{c}{f} \tag{3.1}$$

where λ is the wavelength, c the propagation speed and f the frequency.

It's a boring sound, isn't it? Let's make it a bit more interesting. Running DEMO3.2 will play a chord that is generated by the script but it could also have been produced by some electronic musical instrument. You see that its waveform looks quite complicated already (see also fig. 3.3). And yet it consists of only 3 pure tones added together. It is almost impossible to distinguish these individual sine waves from its waveform by looking at it. However, we can display this sound in a different way: we can create a graph that displays the amplitudes of the individual pure tones as a *function of frequency*.

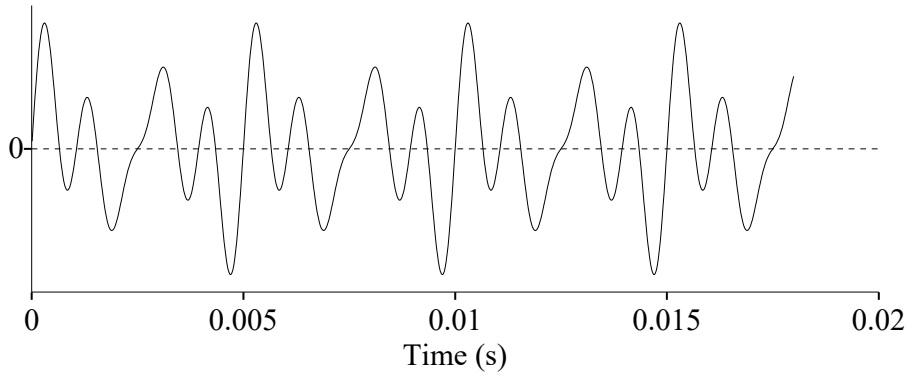


Fig. 3.3. Waveform of three added sine waves.

Fig. 3.4 shows this graph which is called **spectrum**, and it looks extremely simple. Basically, this spectrum consists of only three points, the horizontal position representing the frequency and the vertical position the amplitude. It is customary to draw vertical lines from each point to the horizontal axis: the *spectral lines*.

The vertical axis is scaled in dBs, which is used normally in spectra because the auditory impression of the intensity is logarithmic (see section 2). Another reason is that weak but audible components could be invisible if a linear scale was used. For example, the spectral line of a component that is 40 dB lower than the maximum would have a length of 1/100th of this maximum which is very near to zero on a linear scale. On a logarithmic scale with a customary range of, say, 80 dB this component reaches half of the maximum height in the graph!

The 0-dB reference in Praat spectra is a (one second) sound at the level of our hearing threshold (20 μP SPL). (Although these values suggest a volume calibration, you must not rely on it because the volume of a sound played by the computer depends completely on the positions of the computer volume adjustment, the sound card amplification, the speaker or headphone properties, etc.) It is simply impossible to calibrate the level because of all these unknown variables. (See also the Praat manual about SPL calibration.) The *relative* values, however, can be read perfectly well from the graphs.

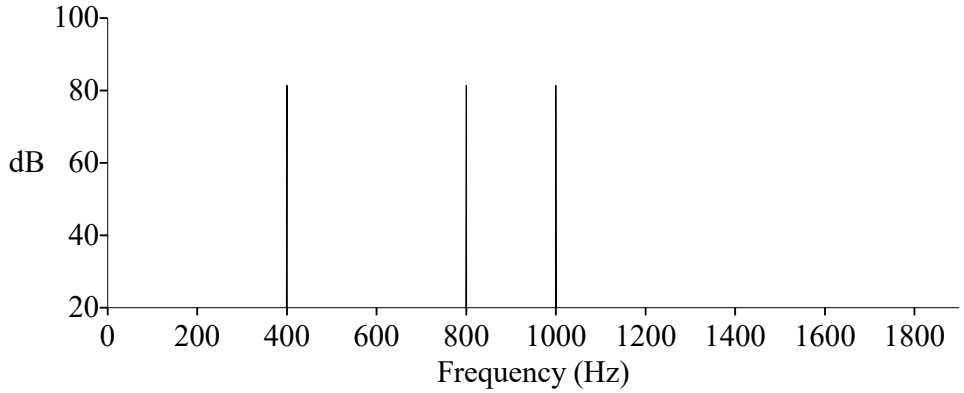


Fig. 3.4. Spectrum of three added sine waves.

Of course, we could construct this spectrum directly from our knowledge of the signal's properties: it consists of three sine waves with equal amplitudes. The Praat program, however, did not 'know' this. Nevertheless, it can accurately find the three sine components one way or another. Somehow it analyses the waveform and produces the sine wave components. The explanation of how this is made possible comprises the main part of the book.

For now, we can conclude that the *waveform* is the way the sound is represented in the **time domain**, and the *spectrum* is the way the sound is represented in the **frequency domain**, as is determined by the nature of the horizontal axes.

A commonly used name is **amplitude spectrum**, as it refers to the amplitudes of the frequency components. Strictly speaking, in the program Praat this is not correct, as this program, like many signal analysis programs, displays *density spectra* where the dB values depend on the time length. For the moment we would not bother about the difference as here it is only a matter of the way of scaling the vertical axis. In section 15 about noise the difference is explained. In Part B we will look at Praat's practical spectrum properties in some more detail.

It is also quite common to use the term **power spectrum**. Although we mean *SPLs* in the time and frequency representations of sound, the values in dB *refer to the power or intensity*, which is the amplitude *squared*, as you will remember from section 2. On a logarithmic scale, therefore, the values are doubled. Naturally, the *shapes* of the spectra remain unaltered when only the numbers along the axis are doubled.

4. Fourier series

When we look at the waveform of DEMO3.2 (fig. 3.3) we can see on the scaled time axis that a specific pattern is repeated each 5 ms during the entire signal. No matter where you define the start of this 5 ms section, the pattern is always repeated after 5 ms. In other words: this repetition period occurs 200 times per second. Therefore, the repetition has a **frequency** of 200 Hz (hertz), which is called the **fundamental** frequency. The relation between fundamental frequency and repeated period is represented by this simple formula:

$$F_0 = \frac{1}{T_0} \tag{4.1}$$

where F_0 and T_0 represent the fundamental frequency and the period, respectively. Frequency can be expressed in Hz, or *cycles per second*, which speaks for itself.

One repetition period of the waveform contains all information about this signal as all periods are the same. The complete data of all 3 sine waves are present in each single period. Now, a French mathematician and physicist, Jean Baptiste Joseph **Fourier** (1768-1830) proved that all *periodic* signals, i.e. signals that consist of repetitive parts of constant lengths and forms, can be represented by just combinations of sine waves of specific amplitudes and with frequencies that are *multiples only* of this fundamental frequency (F_0), including F_0 itself. The general form of a periodic signal is:

$$x(t) = x(t + T) \tag{4.2}$$

Here $x(t)$ stands for the varying sound amplitude as a function of time. We can conclude that any periodic signal can be represented by a set of sine waves of specific amplitudes and with frequencies of F_0, $2F_0$, $3F_0$, $4F_0$, etc. In the example above, we have the following frequency components or **harmonics**:

1st harmonic:	F_0:	amplitude: 0	frequency: 200 Hz
2nd harmonic:	$2F_0$:	amplitude: 1/3	frequency: 400 Hz
3rd harmonic:	$3F_0$:	amplitude: 0	frequency: 600 Hz
4th harmonic:	$4F_0$:	amplitude: 1/3	frequency: 800 Hz
5th harmonic:	$5F_0$:	amplitude: 1/3	frequency: 1000 Hz

next multiples all have amplitude 0.

Obviously, this is a simple example. Nevertheless, it teaches us that the amplitude of the frequency F_0 itself can be zero whereas the corresponding period (here 5 ms) is clearly present. The fundamental frequency only depends on the length of the repeated part which is determined by the *greatest common divisor* of the frequency components! Only then the frequency components will be multiples of a common fundamental frequency. In the part B section on frequency range it will also be shown that even for a sound with a clearly audible fundamental frequency this frequency component *need not necessarily be present in the spectrum*!

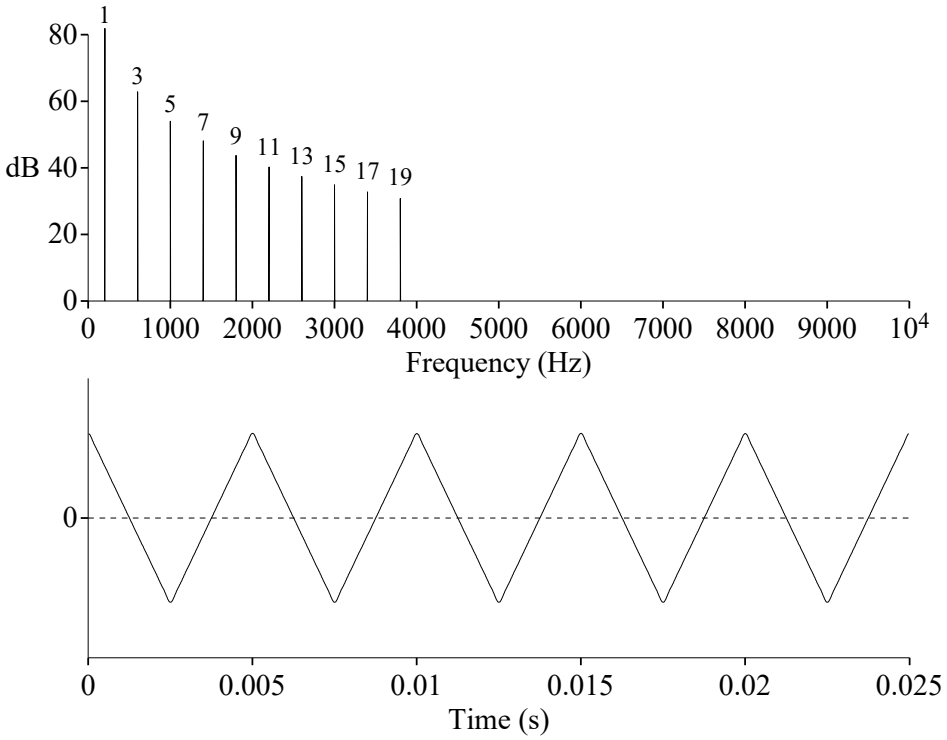


Fig. 4.1. Triangle waveform approximation with 10 odd harmonics.

Now, let's add some other sine wave components together. DEMO4.1 asks you to type the number of harmonics to generate. Try the default (already filled-in) value first.

A part of the resulting waveform is displayed and it looks like a *triangle* wave. (See also fig. 4.1.) The corresponding spectrum is displayed as well which shows the 10 harmonics. (You will see that there are only odd-numbered harmonics; the even harmonics are absent. This is always the case when the signal is symmetrical in the

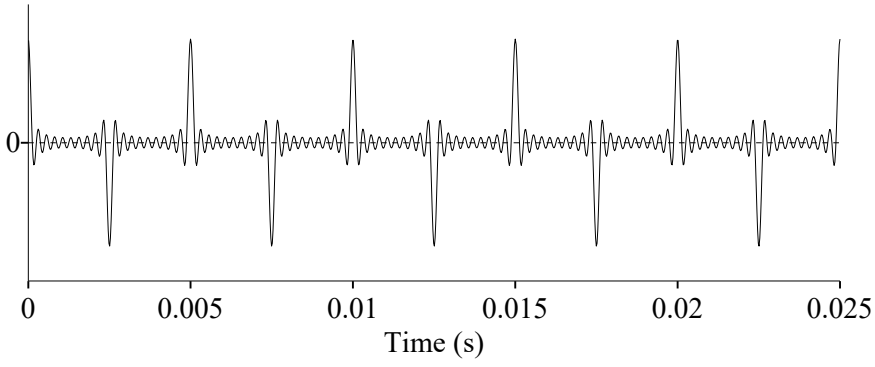


Fig. 4.2. Waveform from 10 odd harmonics with equal amplitude.

sense that the lower halves of the waveform are mirror images of the upper halves. Later, in section 6, we will learn more about symmetry properties.)

Of course, this triangular wave example is a theoretical one: in practice the waveforms are usually much smoother. This example is merely used for analytical purposes.

The spectrum shows that the demo script is designed in such a way that the amplitude of the harmonics decreases with increasing frequency. This *roll-off* is very 'natural': almost all waveforms of natural sounds can be synthesized with sets of harmonics that have some roll-off. As a contrast see fig. 4.2 for the waveform which arises when all 10 harmonics of the triangle example of fig. 4.1 have equal amplitude. The relatively shallow parts of the triangle have disappeared which indicates to the effect that the high frequency components are 'emphasized' at the cost of the low frequency components.

If you run the script again with 3 harmonics instead of 10, you will see that the triangle waveform is no longer perfect. You can try any number of harmonics. (Obviously, if you run the script with only one harmonic a perfect sine wave will emerge.)

Now run DEMO4.2 to approximate a *sawtooth* waveform (see also fig. 4.3.). You may experiment with different numbers of harmonics again. What you see is that the required number of harmonics to approximate the waveforms with some degree of

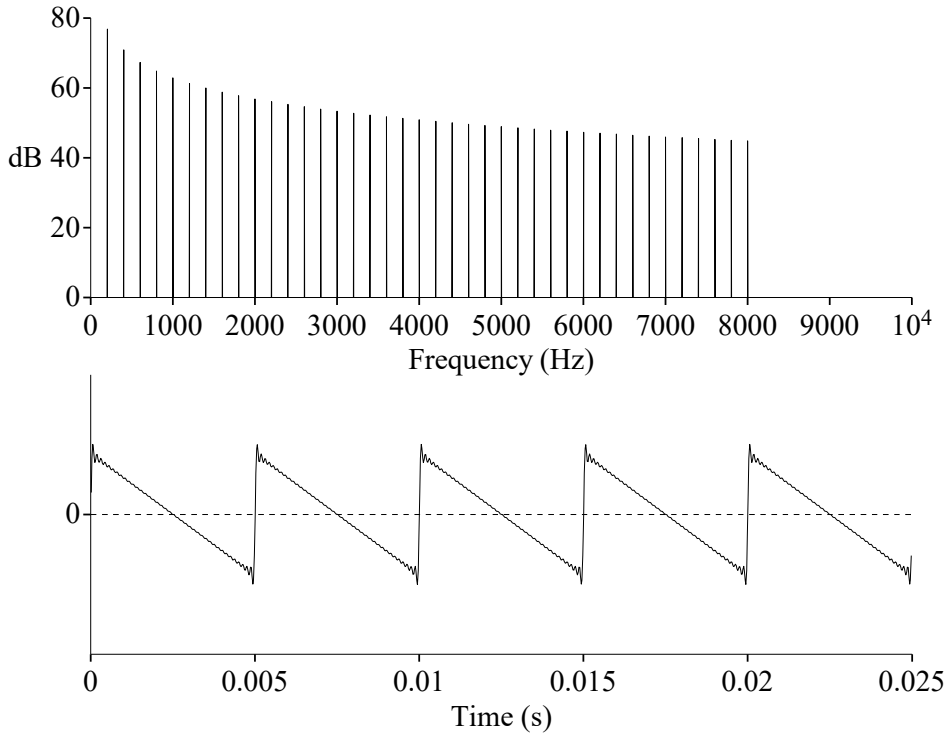


Fig. 4.3. Sawtooth waveform approximation with 40 harmonics.

accuracy is higher for the sawtooth wave than for the triangular wave (even when you include in your count the missing even harmonics of the triangular wave).

Theoretically, for signals with sudden changes or *discontinuities* in the waveform, we would need an infinite number of harmonics to acquire the perfect waveform. The sawtooth wave has a very big step in the waveform which needs strong high harmonics compared with those for the triangular wave. In practice, these waveforms do not exist in perfect form: their discontinuities always take some time which means that they are not pure mathematical discontinuities. Nevertheless, in practice it is always possible to approximate the practical waveform almost perfectly with a limited number of harmonics.

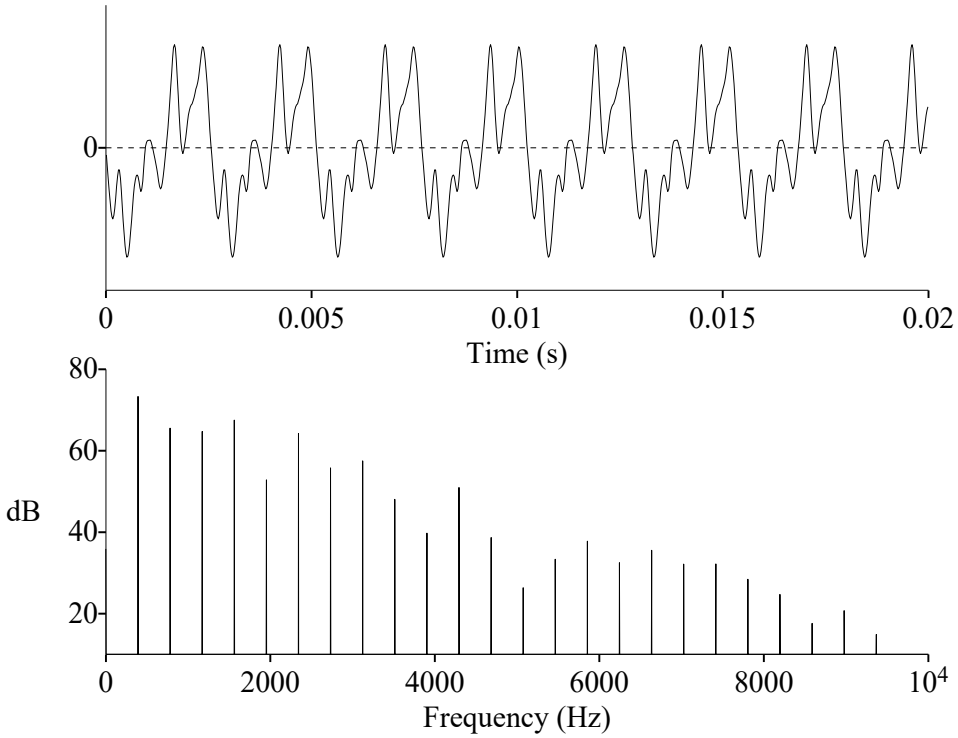


Fig. 4.4. Waveform of steady part of a violin sound, and its spectrum.

In these examples we have applied a Fourier *synthesis*. Let's do it the other way around. In DEMO4.3 we have a constant steady tone of a violin (see also fig. 4.4). The repeated part (T_0) is about 2.56 ms (milliseconds) so the F_0 is 390 Hz.

After you select *Continue* its spectrum is displayed (also in fig. 4.4). You will see that many multiples of F_0 exist and... in between there is nothing! Theoretically, all frequencies between the spectral lines are zero. (See the box **NO SILENCE ON dB SCALE** about the consequences of the logarithmic scale.)

NO SILENCE ON dB SCALE

It is not possible to properly display zero level on a logarithmic scale, like the dB scale. This is caused by the fact that the power of the logarithmic base number (say 10) must be minus infinity to get zero: $0 = 10^{-\infty}$. When generating signals with a computer program, the zero values (absent harmonics, for example), could jump to minus 300 dB or so, depending on the precision of the numbers used in your computer. Therefore, the dB range displayed in practice one mostly limits to 60 or 90 dB. In "Praat" you can adjust this range to any value. To see what is present at these extreme low levels you could run DEMO4.3 again and at "dB range" you can temporarily set it to 400 dB for example. Now you see something emerge like a "noise floor". It is caused by the limited number precision in the computer. In practice this is not relevant: ratios of 100 dB or so are more than enough as this covers already an intensity ratio of 1 to 100000.

At the high frequency end of the spectrum the amplitudes of the harmonics decrease and gradually approach *negligible* values. The fact that the spectrum is empty between the spectral lines may seem strange and some people attribute this to the Fourier math peculiarities. However, it can easily be seen that this is a fundamental property: when the frequency of a sine component is a multiple of the fundamental frequency, its contribution to *each period* of the waveform is exactly the same. But, in the case of a sine frequency that is, for example, 2.5 F_0, its contribution to adjacent periods of the waveform is different: the phase of the sine component is not the same for each waveform period. (In this case it takes 2 periods of the waveform to reach the original phase; when the sine component is 2.3 F_0, for example, it takes 10 waveform periods before the phase has returned again to its initial value.)

Therefore, the requirement for a sine component to have the *same contribution* to all different periods of the waveform implies that it must have a frequency that is a multiple of the fundamental frequency.

ABOUT PERIODICITY

What if we add, say, a sine wave of 100 Hz and a sine wave of 103 Hz? Obviously, the fundamental frequency is 1 Hz (not 3 Hz!). So, strictly speaking, all multiples of 1 Hz are zero, except number 100 and number 103. Only if the ratio of the two frequencies is *exactly* 100/103 this is valid. For example, a ratio that is not exactly 100/103 but, say, 100.01/103 would produce a fundamental frequency as low as 0.01 Hz and does not make sense. In practice, we can fulfill this accuracy requirement for the Fourier synthesis very easily when we use a computer or other electronic device. It will be clear that, strictly spoken, in practice it is impossible to get this exact ratio when you add the sine waves of two *separate* sine sound sources together. Even when you use two separate computers that can generate sine waves with highly accurate frequencies, there will always exist some inaccuracy of the frequencies. Obviously, the concept of harmonics is only valid when the frequency components are *related* to each other, as is the case when they stem from one sound source only (or, one sound source *synchronizing* another one).

All these measured frequency components of a specific periodic signal together form its **Fourier analysis**. It's the opposite of Fourier synthesis. To be more precise, the Fourier analysis of a *periodical* signal is called **Fourier series**, just for the reason that the spectrum consists of a series of single frequency components, the harmonics.

In musical terms one speaks of **overtones** instead of harmonics. There is a small incompatibility: the *second* harmonic ($2F_0$) is the *first* overtone, and so on. The F_0 is called the **fundamental**.

5. Sine wave basics

Why use sine waves? Why not use other forms like, for example, triangular or rectangular waves or short pulses as frequency components? When we look at the swing once more we see that it is moving in a natural way. It turns out that all natural-vibrating masses mostly do so in a sinusoidal form which in a manner of speaking is the most 'lazy' way. There must be a reason for the sine tone from DEMO3.1 to sound so boring! The added harmonics made the tone more interesting.

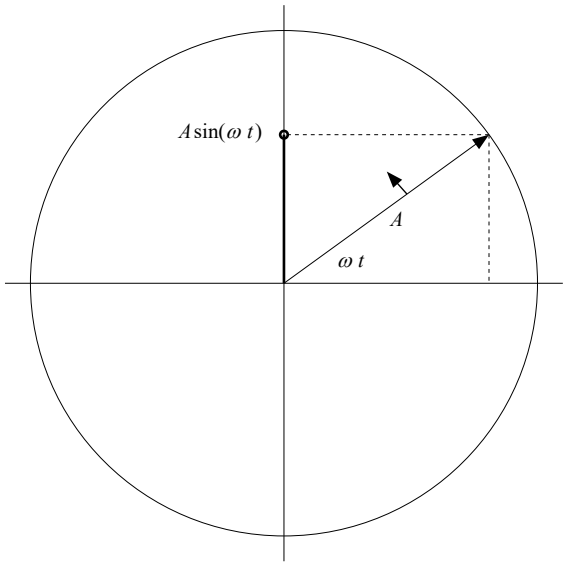


Fig. 5.1. Construction of sine wave by projection of rotating vector on y-axis.

Another reason to use sinusoidal waves is that these do not change shape when they are processed by filters or amplifiers: only their amplitude and phase (time shift) can be influenced. Other waveforms do not have this property. Spectrally, they are the most basic components. (This property of filters and amplifiers is described in the box called **LINEAR SYSTEMS** in section 8.) Fourier showed us that all periodic waveforms can be seen as an addition of a number of these 'basic' sinusoidal waves.

How can we express a sine wave with a formula? DEMO5.1 shows the construction of a sine wave period from a rotating *vector* (see also fig. 5.1). The length of the rotating vector (its amplitude) remains constant and its angle increases proportionally with time. The sine of the angle is the projection on the vertical axis.

To express a sine wave with a formula it should have a variable for the amplitude and a variable for the angle which is dependent on the time. So, the next formula should do:

$$y(t) = A\sin(360ft) \tag{5.1}$$

where $y(t)$ means a function of time (t), usually more generally expressed as $f(t)$, A its amplitude and f its frequency. In this formula we see that only the angle of the sine, here expressed in degrees, increases proportionally with time; everything else remains constant. As an example, if f is 1000 Hz, the angle increases to 1000 times a whole 360 degrees of a circle within 1 second. In other words, the angle increases during 1 second from 0 to 360000 degrees.

Now, in math it is common to define angles in *radians*. When the radius of a circle is paced out on its circumference it selects a sector with an angle of 1 radian. As a consequence, there are 2π radians in a complete round. The formula then becomes:

$$f(t) = A \sin 2\pi f t \tag{5.2}$$

For this purpose, the "angular frequency" (ω) is used instead of f, defining $\omega = 2\pi f$. (See the box **THE RADIAN** about this definition.) Consequently, the formula for a sine wave now becomes:

$$f(t) = A \sin \omega t \tag{5.3}$$

The formula implies that the function 'goes on forever': t runs from -∞ to ∞. At t=0 the sine value is zero but that is arbitrary, as we could just define the zero time point somewhere else in the wave. The next function

$$f(t) = A \cos \omega t \tag{5.4}$$

defines the time origin at the position of one of the peaks of the waveform (cos 0 = 1). It remains the same sinusoidal wave but it is only shifted backward in time one quarter of a period. This is important when two or more sine waves are added together, which is always relevant when dealing with spectra. DEMO5.2 shows our sawtooth wave synthesis of DEMO4.2 but now with alternating sine and cosine (= shifted sine) components (see also fig. 5.2). The waveform has changed dramatically and has nothing to do with a sawtooth any more. Obviously, the phase relations between the Fourier components have much influence on the waveform.

THE RADIAN
Defining angles by using the radius as a measuring unit has a mathematical reason. So far in our story there seems to be no objection against defining the complete circle (360^0), for example, as 1 which gives 1/4 for a right angle (90^0), 1/2 for a 180^0 angle, 1/6 for a 60^0 angle, and so forth. Then we would not have to take into account the 2π in our formulas when transforming to and from the frequency and time domains and we would not need this angular frequency, or even degrees, at all! However, the radian simplifies the formulas to transform sine waves into exponential functions which in turn greatly simplify the math (as used in Appendix II). So, we will come across the ω many times from now on.

To define our sinusoidal wave including the initial phase (the angle at t=0) we should use the formula:

$$f(t) = A \sin(\omega t + \varphi) \tag{5.5}$$

where φ defines the phase angle in radians at t=0.

Instead of defining a phase angle we could use a *sine wave plus a cosine wave*. The result is *one* sinusoidal wave, its phase angle depending on the *amplitudes* of sine and

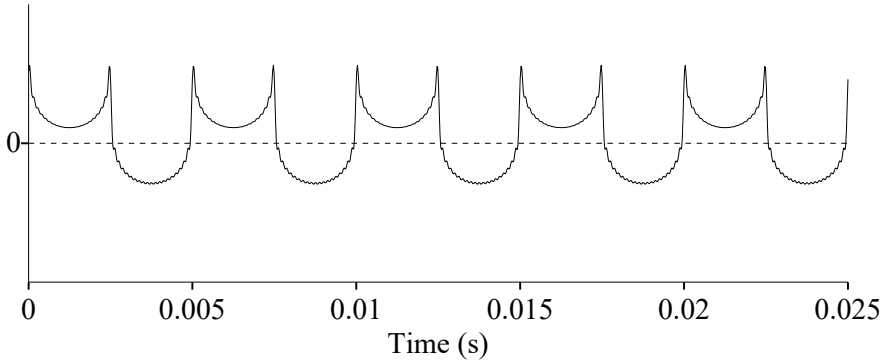


Fig. 5.2. The 40 harmonics of the 'sawtooth waveform' of fig. 4.3, here synthesized with alternating sine and cosine components.

cosine. DEMO5.3 shows what happens to such a wave (displayed from t=0) when the amplitude ratio of sine versus cosine is changed gradually. As you can see, the result is only a phase shift. So instead of formula 5.5 we could just as well use the following formula for a sine wave with phase information:

$$f(t) = A_1 \sin(\omega t) + A_2 \cos(\omega t) \tag{5.6}$$

where A_1 and A_2 define the sine and cosine amplitudes respectively.

For the phase shift range to complete a whole 2π circle these A_1 and A_2 amplitudes should *include negative values* as well. The resulting amplitude (A) of the sine wave made by sine plus cosine is expressed by:

$$A = \sqrt{\left(A_1^2 + A_2^2\right)} \tag{5.7}$$

which coincides exactly Pythagoras' theorem. (In the DEMO5.3 this relation is built-in to keep the resulting amplitude A constant.) This addition of a sine and a cosine *as vectors* with different amplitudes is shown by DEMO5.4.

When a spectrum is displayed, the phase information usually is omitted. Only the amplitudes of the sine wave components are represented. The reason is that our ears are not capable of detecting the phase differences of components of a purely periodic sound (if the period is not extremely long). When you listen to the sounds of DEMO4.2 and DEMO5.2 and select the same number of harmonics, you will hear no difference, whereas the waveforms are very different! (In the scripts I ensured the intensities of both sounds to be equal, to rule out the influence of intensity difference.) Their spectra also look exactly the same, as the phase information is absent in the displays. The formula which defines a complete spectrum, however, must include the phase information. Only then it will contain all information that is present in the signal.

As you will remember from section 2, the waveform of a sound defines the SPL (sound pressure level) which varies all the time. How can we determine the *intensity* of our sine wave? We also read in section 2 that the intensity is proportional to the *square* of the SPL. We cannot square the amplitude A of the sine wave: within the waveform the individual levels all have different values in the range from A to $-A$. Therefore, we have to square the *value in each position* of the waveform, which needs integration as there are infinite positions. To estimate the overall intensity of the sine wave we must take the mean of all these values. Then, to convert the value to an amplitude scale again we have to take the square root of this mean.

RMS

For calculating the power of a sine wave we must square its amplitude function: $P(t) = A^2 sin^2(\omega t)$. To determine the overall power, we must calculate its mean which needs integration over one period:

$P = A^2 \cdot \frac{1}{2\pi} \int_0^{2\pi} sin^2(\varphi) d\varphi$, which can be written as

$P = \frac{A^2}{2\pi} \cdot \frac{1}{2} \int_0^{2\pi} (1 - \cos(2\varphi)) d\varphi$. This integral can be split and the integration of the cosine over two whole periods is zero, which simplifies the formula to:

$P = \frac{A^2}{2\pi} \cdot \frac{1}{2} \int_0^{2\pi} 1 d\varphi$, resulting to $P = 1/2\ A^2$.

So, the equivalent SPL amplitude (i.e. the result of this power) is equal to the square root: $A/\sqrt{2}$ which is the **root mean square** value of a sinusoidal wave with amplitude A.

The name for this value of the sine wave is obvious then: the **root mean square** (rms) value. From the box **RMS** you can see that this rms value of a sine wave is equal to $A/\sqrt{2}$ which is about 0.7 A.

6. Fourier transform

In section 4 it was argued that the spectrum of a periodical sound could only consist of sinusoidal waves with frequencies that are multiples of F_0. The name of the set of Fourier coefficients for these types of signals is called *Fourier Series*. Because of the steps in the frequency range, the Fourier transform is called **Discrete Fourier Transform** (DFT). (For pure periodical signals a more strictly name would have been: 'Discrete Fourier *Series*' but the name DFT is a more universal name, as we will see later.) We will use the term *Fourier Transform* (FT) for the moment.

To calculate the amplitudes (and phases) of these frequency components we need to

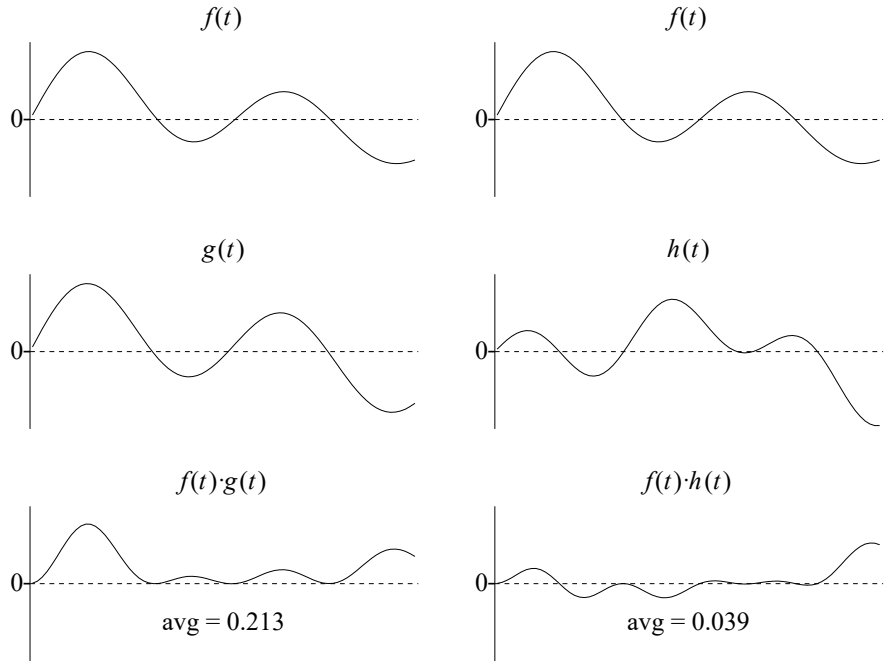


Fig. 6.1. Comparison of signals by multiplication. The function $f(t)$ is better 'imitated' by $g(t)$ than by $h(t)$.

look at only one period of the sound because all periods are the same. How can we calculate these amplitudes and phases? For this, we must go into some detail about *comparing signals*.

In fig. 6.1 some arbitrary functions of time are displayed, called $f(t)$, $g(t)$ and $h(t)$, all with equal duration. The function $f(t)$ looks more like $g(t)$, than $h(t)$, in this example. How can we express this resemblance of functions in an objective manner? Let's assume that the functions we want to examine have the same duration, otherwise the resemblance estimation makes less sense. The answer is to *multiply* the functions and use the mean of the resulting function as a measure for the resemblance. In fig. 6.1 the function $f(t)$ is multiplied by $g(t)$ (left column) or multiplied by $h(t)$ (right column). The

SIGNAL COMPARISON

To compare the functions $f(t)$ and $g(t)$, both having the same duration, we simply multiply them. The mean value of the resulting function then is a measure for the resemblance of the two functions. To take the time interval into account (resemblance over longer time intervals should produce higher factors) the mean is multiplied with the time interval. This is equivalent to the 'surface area' of the graph which means *integration* over the entire time interval. The value achieved in this way is our objective resemblance measure: we have calculated the **cross-correlation factor**. Its formula will now be straightforward:

$$r_0 = \int_{-T/2}^{T/2} f_1(t) \cdot f_2(t) dt$$

Here we selected the centers of the functions as the time origin. This is arbitrary, of course. Obviously, the correlation factor could be negative as well. When one function is the opposite of the other, i.e. when $a(t) = -b(t)$, their correlation has a maximum negative value.

means of the resulting functions then are a measure of resemblance. You can see that this method works for positive *and* negative values of the functions: two negative values at some point cause the resulting function to be positive there. Likewise, if one function is positive and the other negative, the result is *negative* at that point. In addition, when one of the functions is zero at some point, the result is also zero there. In fig. 6.1 for simplicity the mean values of the result are given. The formal way to value the result is explained in box **SIGNAL COMPARISON**. (The only difference is that the mean values are multiplied by the chosen time length of the signal.) We calculated the **cross-correlation factor**.

In the figure it can be seen that the functions of $f(t)$ and $g(t)$ are *correlated higher* than $f(t)$ and $h(t)$, which we already assumed from looking at the shapes of the functions.

Now we can apply this cross-correlation (cc) to the Fourier analysis. We *compare* the period of the signal with each sinusoidal component with amplitude equal to 1. The better the 'fit' the higher the cc or Fourier component will be. We will find all cross-correlation factors of the signal period with all sine and cosine periods that are multiples of the fundamental frequency.

In principle, the Fourier analysis procedure estimates, for each frequency component, which amplitude offers the 'best fit' with the signal period. This is done for sine and cosine periods separately so that the phase information is extracted as well (section 5

with DEMO 5.3 explained how a combination of a sine and a cosine can define a sine wave with any phase angle).
Appendix I explains how the Fourier components are calculated formally. People who do not feel comfortable using some math could simply trust Praat's Fourier transform capabilities to have their spectra presented with a simple mouse click on the button "To Spectrum".

However, the length of the sound or sound part selection is important to bear in mind: only when the length is exactly one period or an exact multiple of one period then the FT graph will show the correct spectrum. In fact, the FT does not 'know' the real period; it regards the sound part you selected as 'the period' and it simply produces frequency components that are multiples of $1/T_S$, where T_S is the length of the *selected* sound, which is not necessarily equal to the real period.

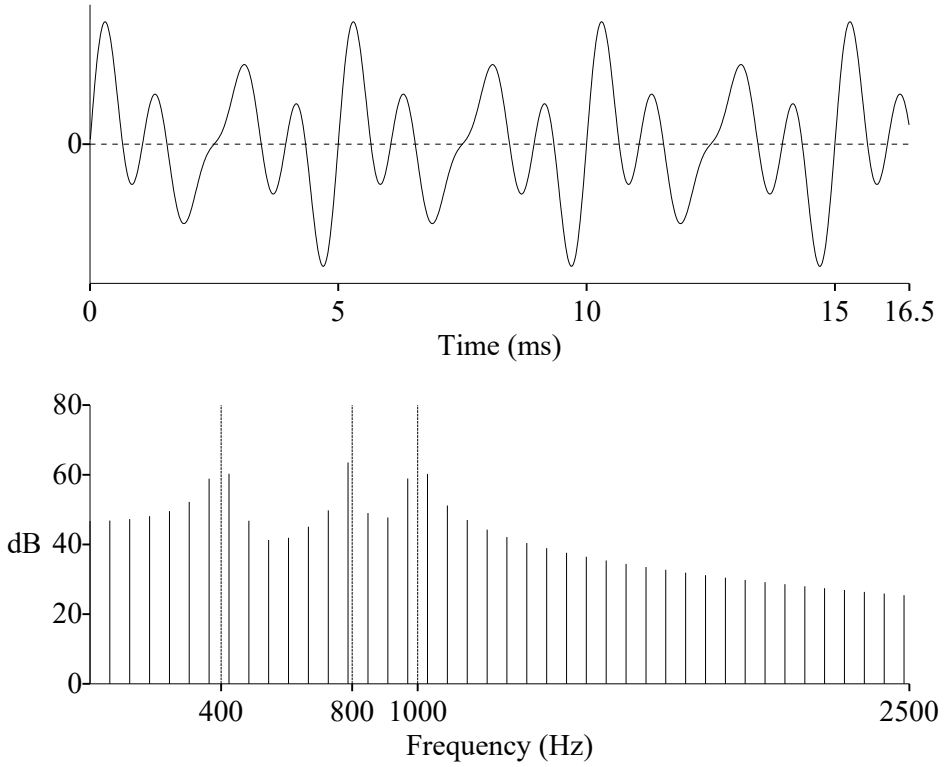


Fig. 6.2. Waveform and spectrum of 3.3 periods of the signal of fig. 3.3 from section 3
The 'real' components (dashed lines) are absent.

In fig. 6.2 we see a spectrum of a sound part that contains 3.3 periods of the periodic signal of fig. 3.3 of 200 Hz fundamental frequency. The length is 16.5 ms (3.3 x 5 ms) so that the FT only calculates frequency components that are multiples of 1000/16.5 Hz or about 60.6 Hz. There are many of them whereas the 'real' frequencies with which the signal was constructed (400, 800 and 1000 Hz) are absent. The 800 Hz component

shows up almost at the right place but only because the 13th multiple is 788 Hz which is very near 800 Hz. The ‘real’ 800 Hz has also gone.

If the periodic signal from which we want to take the FT is much longer (say, 1 second or so) then the FT components are much more ‘closely packed’ and, therefore, take positions much nearer to the ‘real’ frequencies and the ideal spectrum is much better approximated (see fig. 6.3 which displays an FT of about 0.4 second of the same signal). The spectral lines are so close that they form a black background[1]. But even then, most components have frequencies that have nothing to do with the original 3 components

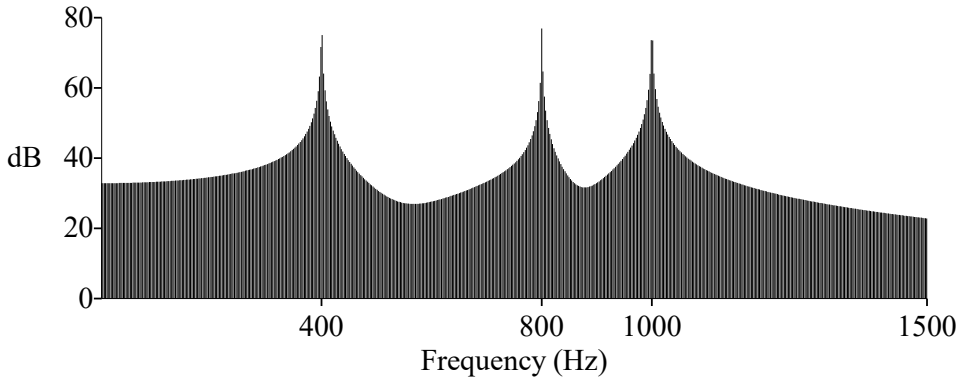


Fig. 6.3. Spectral leakage: spectrum of 0.4015 seconds of the signal of fig. 3.3.

with which the signal was assembled. This phenomenon is often called ‘**spectral leakage**’. The only selections which get rid of these unwanted components have the length of exactly *one period or multiple periods* of the signal. Only then the FT components will fit completely. A more practical method to suppress this spectral leakage and its explanation we will meet in section 13 about windows, which is an essential part of the book.

In the scripts for generating the figures I made sure that Praat uses the DFT instead of the FFT (Fast Fourier Transform). For the FFT it is necessary that the program lengthens the signal with some sound portion of zero amplitude. This could distort the shape of the spectrum. If you want to use the ‘To Spectrum’ button for explorations in Praat, you should NOT select the box ‘Fast’ in the next window, otherwise Praat will calculate the FFT. Unfortunately, the ‘Fast’ option is the default choice. The difference between DFT and FFT is explained in section 26.1 about Praat’s Spectrum in the practical part of the book, together with the way to minimize the unwanted spectral FFT effects.

1 **Remark about the spectrum displays applied.** For explanation purposes the spectra from figs.6.1 and 6.2 look different from Praat's normal spectrum displays, where the spectral points are interconnected with lines (linear interpolation). For clearness of the spectral effect concerned I choose the pure 'line' spectrum display (possible via the ‘Ltas’ in Praat: to be explained in part B).

It is important to realize that the phase information of all frequency components is omitted in all these spectrum graphs. However, when we transform the Spectrum object in Praat back into a Sound object again, the original sound is reconstructed in its entirety. This is possible because Praat's Spectrum object contains the phase information as well. In fact, in the Spectrum object both the sine and cosine values are present, so it forms the complete Fourier transform of the sound. DEMO6.1 shows the spectra of two different sounds, one sound consisting of a smoothed (gradually started and stopped) part of a sine wave of 300 Hz *followed* by a sine wave of 800 Hz (also smoothed in the same way), both during 1 seconds, and the other sound consisting of the *sum* of the same components. The spectra are exactly the same! The two sounds have the same spectral components, only their phase relations are different, which are not contained in the 'power spectrum' displays. (The smoothing avoids the spectral effects that arise when the selected interval is not filled with a whole number of periods of a component, which is the case for the first sound. This smoothing is explained in section 13 about windowing.)

In the example given in fig. 6.2 the spectrum has zero value at 0 Hz. Of course, a frequency component of 0 Hz does not make sense. But from the Fourier transform the value at 0 Hz can be seen as the correlation of the period with a 0 Hz sine wave with an amplitude of 1 which is nothing else than a horizontal line with the value 1. In other words: the value is equal to the *mean* of the wave. In most practical sound waves this

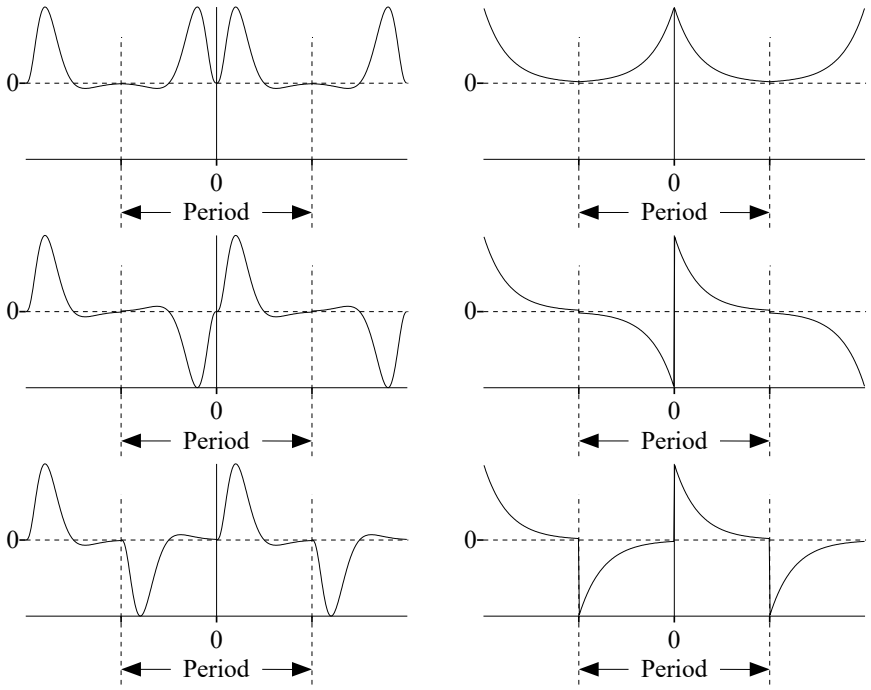


Fig. 6.4. Waveform symmetry. Upper row: even functions. Center row: odd functions. Bottom row: half wave symmetry.

value is zero (or very near zero) as the waveform areas above and below the horizontal axis are mostly in balance. In practice this value is often called the *DC component*. (DC stems from the words Direct Current as opposed to Alternating Current, which originally were electricity concepts.)

Some types of signals have special phase properties because they are *symmetrical* in some way. If the waveform is symmetrical with respect to the y axis it is defined by the formula $f(-t) = f(t)$. Some examples you can see in the upper part of fig. 6.4. (As mentioned before, the choice of the time origin for periodical signals is arbitrary because in theory these signals have no beginning or end. For simplifying the math, it is often convenient to define the *center* of the period of the symmetrical signal as the time origin.) All sine components then, are zero: there are only cosine components. The explanation is as follows: every sine component is computed by multiplying it with the function. Every positive position of the result on the time axis has a value that is the opposite of its corresponding negative position. This multiplication with the symmetrical function therefore causes canceling of all these pairs of points, causing all sine components to be zero. The cosine component pairs multiplied by the symmetrical function do not cancel: they are added. These signals are called **even**.

When the signal has a rotational symmetry of 180^0 with respect to the origin, it can be defined by the formula $f(-t) = -f(t)$, as the center of fig. 6.4 shows. These signals are called **odd**.[1] With the same reasoning as we used above it can be seen that for all odd functions the cosine components are zero: only sine components exist.

A special type of symmetry is called **half wave symmetry**. These functions are defined by $f(t + T/2) = -f(t)$, as shown in the bottom of fig. 6.4. By looking at their waveforms it can be concluded that all products of the signal period and frequency components with an even number of periods will be zero. In other words: for this type of signals *only the odd-numbered harmonics* exist. (In general cases, both sines *and* cosines are present. Of course, when there the signal is *also* even or odd, then, respectively, only odd-numbered cosines or only odd numbered sines exist.)

Now you can understand why the spectrum of the triangle wave of fig. 4.1 of section 4 only contains only odd-numbered harmonics. What's more, due to the symmetry with respect to the y-axis, it contains only cosines. From this symmetry knowledge you can also conclude that the spectrum of the sawtooth wave from fig. 4.3 only contains (odd- and even-numbered) cosines.

The transform from spectrum back to sound again is named '**inverse** Fourier transform'. As you can guess, it simply comes down to an addition of all sine wave components with their individual amplitudes and phases (or all sine and cosine waves).

---

1 The words *even* and *odd* are used because of the fact that functions raised to an even power (x^2, x^4, x^6, etc. and also cos x) are always positive for positive and negative x, and functions raised to an odd power (x, x^3, x^5, etc. and also sin x) always have the same sign as x, thus obeying the formulas for even and odd functions respectively.

To distinguish the transform from sound to spectrum from inverse transform, it is often called '**forward** Fourier transform'.

Understanding the principle of the Fourier transform is a very important step in our understanding of signals. Many things still have to be explained to understand the basic relation between time functions and their spectra but most of them are based on the Fourier analysis principles.

7. Resonator

In section 3 we used the example of a swing and mentioned the fact that after a single push from its position at rest, it swings back and forth with a *constant frequency*, its deviations from its rest position slowly decreasing, until it finally comes to rest again because of air resistance and friction. Therefore, after each cycle, its *amplitude* is decreased by a small percentage. You could say that after some time the amplitude has halved and, after the *same amount of time* the former amplitude has *halved again*. This decrease of amplitude over time occurs conforming to an exponential curve (red line in fig. 7.1). It has a constant *percentage* of decrease per time unit. This can be expressed by the following formula:

$$A(t) = A(0) \cdot 10^{-pt}, \quad t \geq 0, \mathrm{p} > 0 \tag{7.1}$$

Here is $A(t)$ the amplitude at time t, $A(0)$ the amplitude at start and p determines the rate at which the amplitude decays. For example, when p = 1/40 then the amplitude decreases to 0.1 of its initial value after 40 seconds (pt = 1 then). Consequently, after 120 seconds its amplitude is 1/1000 of the start value (perhaps not an unrealistic value for a swing).

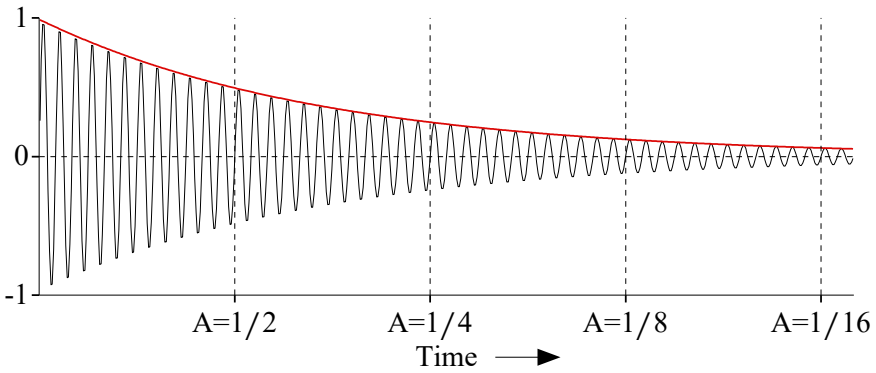


Fig. 7.1. Sine wave with exponential decaying amplitude.

Instead of logarithms with base number 10 it is customary to use the *natural logarithm* (ln): it has base number e which is 2.781828... This e stems from Leonhard Euler, a Swiss mathematician and physicist. As you know, a logarithmic function can be defined using any base number. The reason for using e is its mathematical convenience: differentiation and integration of logarithmic functions with base e offer great simplicity compared to base 10 or other base numbers. And differentiation and integration are often necessary when transforming signals from one domain into the other. Do not bother about the physical meaning of the choice of the base number: there isn't any here! Now, our formula becomes as follows:

$$A(t) = A(0) \cdot e^{-\alpha t}, \quad t \geq 0, \alpha > 0 \tag{7.2}$$

Here the decay rate is defined by α. If it is 0.0576 then the decay rate is the same as mentioned in our swing example above. (Sometimes the decay rate is defined as the time interval wherein the amplitude decreases 50 percent, also called the *half-life*. In our case this time interval is: ln(2)/α, or, in the base 10 log case: log(2)/p which equals 12.04 seconds. This can be calculated by simply substituting $A(t) = 1/2 \cdot A(0)$ in the formulas.)

If we replace the amplitude of the general formula for the sine wave (5.3) with formula 7.2, we have the expression for the single **damped sine** wave:

$$x(t) = A(0) \cdot e^{-\alpha t} \sin(\omega t), \;\; t \geq 0, \alpha > 0 \qquad (7.3)$$

In practice there are many more examples of objects that can produce damped sine waves: they are called **resonators**. Think about popping corks from bottles, tapping glasses, reverberation of rooms, oral cavities for speech and... the sounds of most musical instruments! They all obey the same formula 7.3, each with their own values of their damping α and their **resonance frequency** ω (although the ways the sounds start can vary).

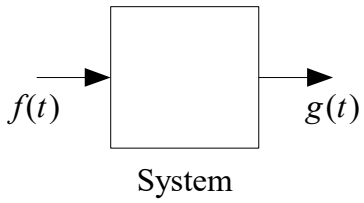


Fig. 7.2. The signal $g(t)$ is the result of the input signal $f(t)$ modified by the system.

Devices like resonators are called **systems**. They do not generate sound on their own but have to be *excited* (activated) by some input signal. The resulting output signal is the response on the input signal, the form of the output depends on the input and the system. This is schematically represented by fig. 7.2 where $f(t)$ is the input signal and $g(t)$ is the output signal.

The resonators are a special type of systems. These resonators have very useful properties, as we will see.

8. Filtering

When, while sitting on a swing, you start aiding it by movement of your body at the 'natural' pace of the swing the deviation increases until it reaches a steady value, whereas moving your body at lower or higher rates the final steady value of the deviation seems to decrease. Apparently, the swing can be seen as a 'filter' which amplifies what we might call the 'input' when its frequency is equal to or near the

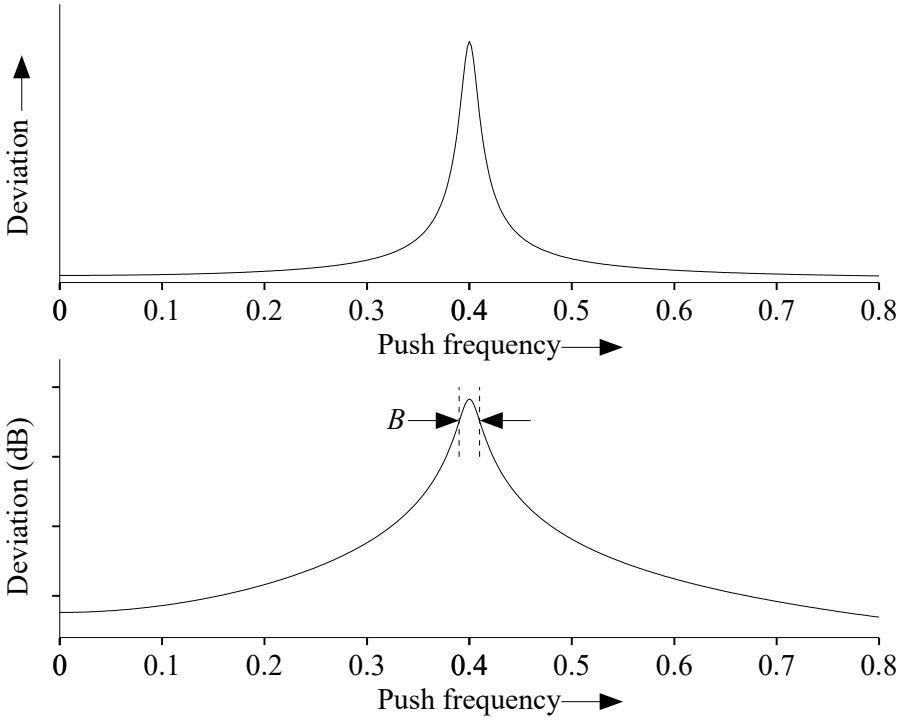


Fig. 8.1. Deviation of swing as function of push frequency. Top: linear amplitude scale; bottom: logarithmic amplitude scale.

natural frequency of the swing and attenuates the input when its frequency is outside this range. This is equivalent to the working of a **band-pass filter**. Indeed, electronic devices can be made that behave like resonators which could be used as band-pass filters.

When we construct a graph that displays the final swing amplitude as a function of "body movement frequency" or input frequency, we end up with something similar to fig. 8.1. The peak of the deviation shows at 0.4 hertz which means that our imaginary swing has its **resonance** frequency at 1 cycle per 2.5 seconds. This graph is nothing less than a band filter function in the frequency domain.

Just like spectra, it is customary to display the amplitude axis of filter functions on a logarithmic scale, as also shown in fig. 8.1.

The filter is a *system*: it *responds* to the input signal. To define its properties the *ratio* of output amplitude and input amplitude is relevant only. Therefore, the value at each frequency of the filter function represents the output amplitude divided by the input amplitude for the frequency concerned.

LINEAR SYSTEMS

Devices like amplifiers, attenuators (like volume controls), and filters, are all *linear systems*. They have two fundamental properties:

1. When the input amplitude is multiplied by a factor *a,* the consequence is that the output amplitude is multiplied by the same factor *a* (homogeneity).
2. When the input can be seen as the sum of different frequency components, the output is the sum of the responses to each component separately (additivity).

Devices like modulators, distorting amplifiers, frequency mixers, etc. all are *non-linear* systems as they produce frequency components not present in the input.

Important to mention here is the fact that a specific sine wave at the input of a filter causes a sine wave at the output with the same frequency: no frequencies are generated by the filter. Only the amplitude and phase of the input sine wave components are altered by the filter (see the box called **LINEAR SYSTEMS** about this property).

The filter function, defined by the *ratios* of output and input amplitudes for each frequency can be expressed by:

$$H(\omega) = \frac{G(\omega)}{F(\omega)} \tag{8.1}$$

where $H(\omega)$ is the filter function or **transfer function**, $G(\omega)$ the output spectrum and $F(\omega)$ the input spectrum. Because of the fact that the system is linear, all input sinusoidal components emerge at the output, only with their amplitudes multiplied by the filter system. Schematically filtering can be visualized as in fig. 8.2 which is the frequency domain version of fig. 7.2. (Later on, in this section we will learn that for estimating the output spectrum the *phase properties* of the filter must be taken into account. Strictly speaking the functions *F*, *H* and *G* here refer to the *complex* spectra.)

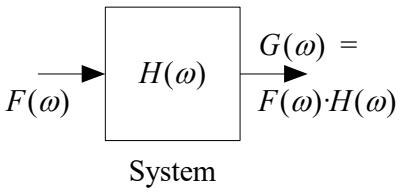


Fig. 8.2. The output spectrum $G(\omega)$ is the result of the input spectrum $F(\omega)$ multiplied by the filter transfer function $H(\omega)$.

The band filter of fig. 8.1 is the simplest type in this category: it is a *second order* band filter (which stems from the fact that its mathematical transfer function consists of terms with powers of a maximum of 2, as described in section 18 and appendix III).

Obviously, this resonator-type of band filter function has, apart from the position of the peak, another variable: the *width* of the graph. The broader the width, the greater the

range of frequencies that will pass through the filter, and vice versa. This variable of the filter is called **bandwidth** (*B*) and is expressed in hertz.

Because the slopes of practical filter functions in general vary gradually, we need some formal procedure how to define bandwidth. To the left and the right of the top of the function we can define points on the frequency axis where the function has decreased to its half-power value with respect to the top power value. The bandwidth, then, is the difference between these frequencies, see fig. 8.1. In filter theory, and generally in the whole field of signal analysis, one deals with amplitude (voltage) instead of power. We know from section 2 that the power is proportional to the square of the voltage. Then the power halved means that the amplitude must be divided by $\sqrt{2}$. So, we must find the frequencies where the function is $1/\sqrt{2}$ times its top value, which is -3 dB if the top value is defined as 0 dB. (Although the graph represents amplitude, the dB values refer to power, as you may remember.) Therefore, this bandwidth is called $B_{\sqrt{2}}$ or $B_{3\mathrm{dB}}$ to distinguish it from possible other definitions. When there is *no subscript*, this $B_{3\mathrm{dB}}$ is meant.

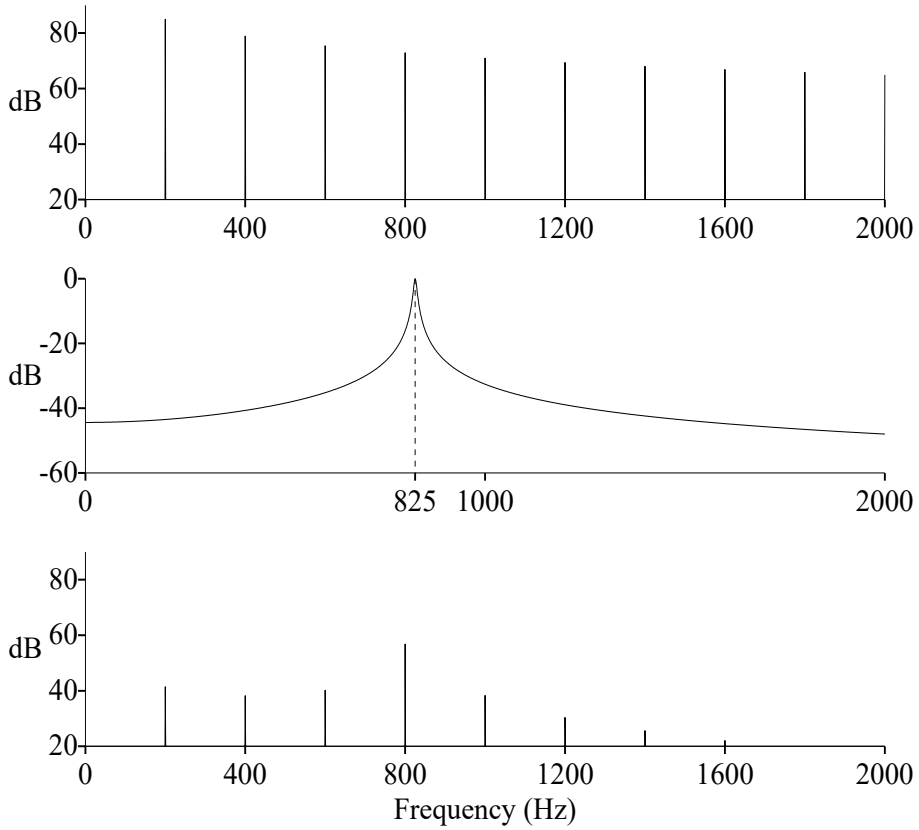


Fig. 8.3. Spectrum of saw tooth wave from fig. 4.3 (top) filtered by resonator filter of 825 Hz (center) results in their multiplication (bottom).

Now, imagine activating a resonator repeatedly, for example by using our sawtooth waveform. In the frequency domain it is easy to understand what will happen: all individual frequency components (which all are pure sine waves) will have *their own* amplification or attenuation (and phase shift) caused by the resonator according to its filter form. As a consequence, the resulting spectrum is the original spectrum multiplied by the filter function (fig. 8.3). This graph is the spectrum of the signal at the output of this resonator band filter.

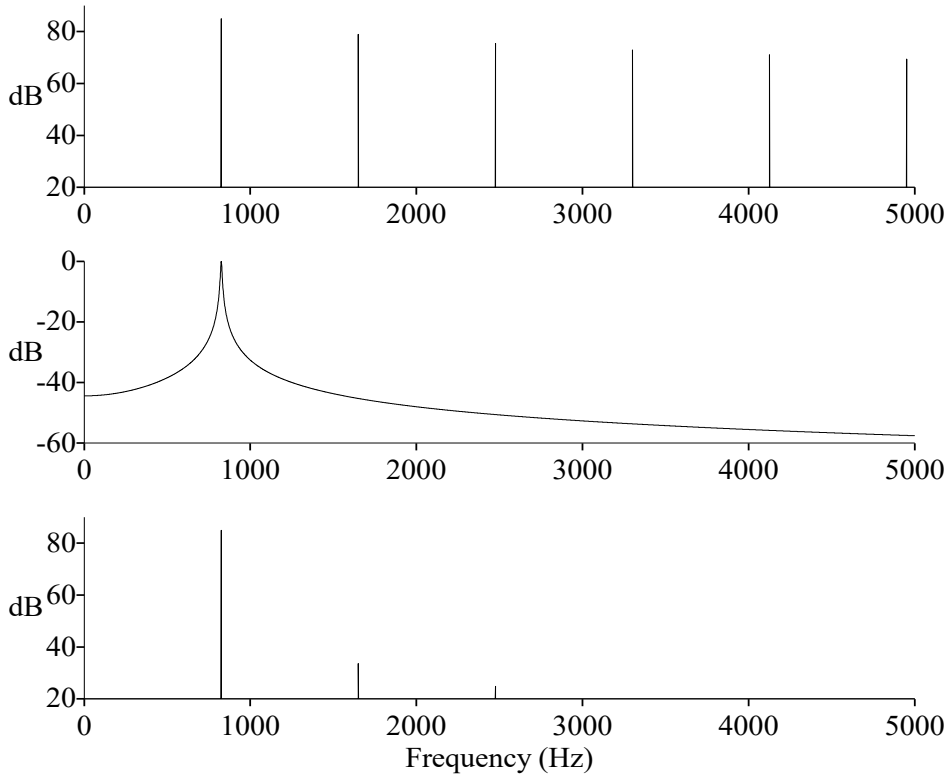


Fig. 8.4. Same as fig. 8.3 but now fundamental frequency of saw tooth 'tuned' to resonator frequency (825 Hz).

Next, we can do the same by activation with a higher frequency: when the repetition rate (= fundamental frequency) is equal to the center frequency of the filter (825 Hz) we get the spectrum of fig. 8.4. We see that the next harmonics of this spectrum fall far from the filter function center and are very much attenuated.

Fig. 8.5 displays the opposite: a very low repetitive rate (8 Hz) is applied so that the spectral lines are very near to each other. (The frequency range is limited to 2000 Hz to zoom in on the area of interest.) The filter function has been approximated almost perfectly in the vicinity of its peak, were it not for the gradual attenuation of the amplitudes as the frequency increases (the 'roll-off') of the saw tooth spectrum which is the reason for the relatively high level of the low frequency components. (In the

picture the overall level of the filtered spectrum is raised to compensate for the low level of the harmonics of the saw tooth in the vicinity of the resonator frequency.)

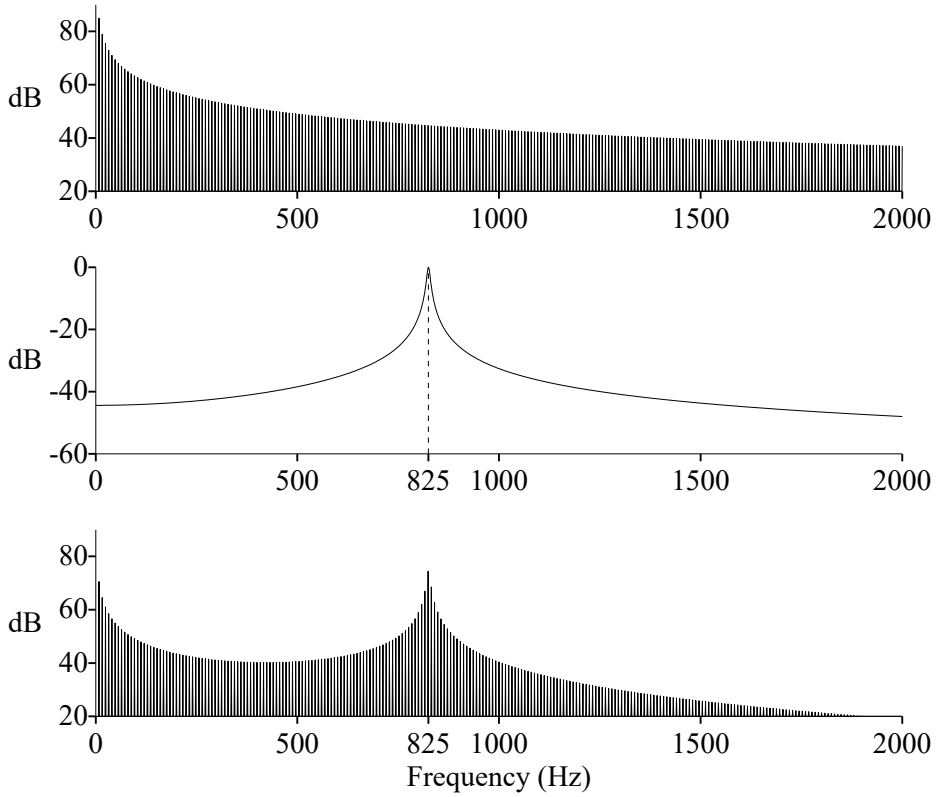


Fig. 8.5. A very low F_0 of the saw tooth wave offers a good approximation of the resonator filter graph near its peak, apart from the gradual decay of increasing frequencies (roll-off) of the saw tooth spectrum. (The level has been raised 30 dB to compensate for the low amplitude of the harmonics.)

What we see is that in all cases the spectral amplitudes at the output form a *sampled version of the spectral band filter function* if we, one way or another, correct for the slope of the saw tooth spectrum. Sampled not in the (usual) time domain but in the *frequency domain*.

When we apply a series of very short pulses instead of the saw tooth for activating the resonator then it will 'run on its own' during the whole period between the pulses, just like the swing after it's been given a push (see section 3). It can be expected that the influence of the excitation source on the resonator is then minimal. DEMO8.1 shows the resonator activated in this way: the resonator filter function has been very well approximated (see also fig. 8.6). After each excitation pulse we can see the damped sine wave of our swing, only on a much faster pace so that we can hear it.

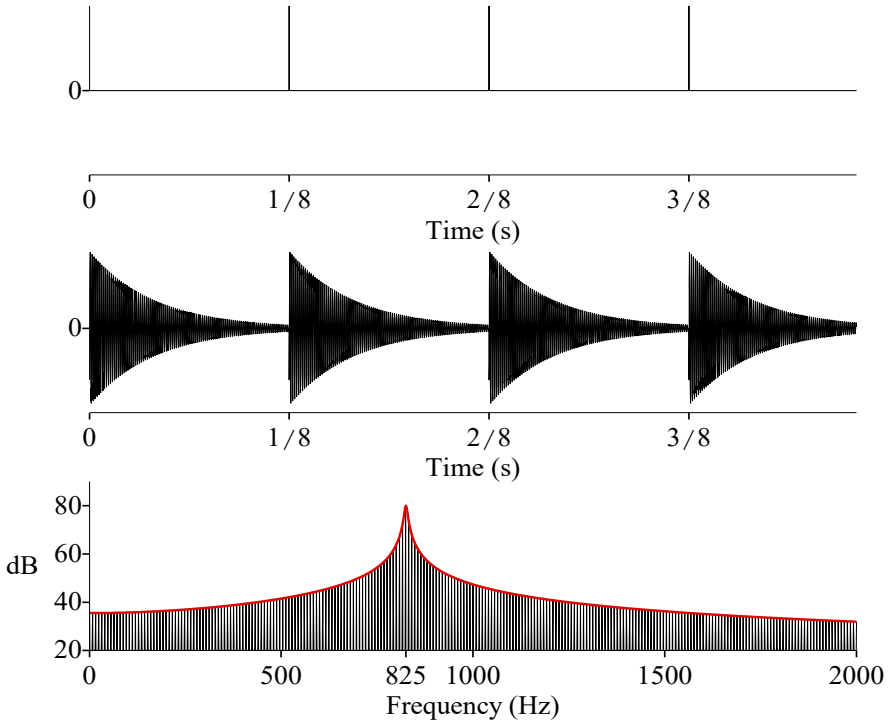


Fig. 8.6. Excitation of resonator with short pulses at low repetition rate (8 Hz). Top: input. Center: output. Bottom: spectrum approximates the filter spectral function (added in red).

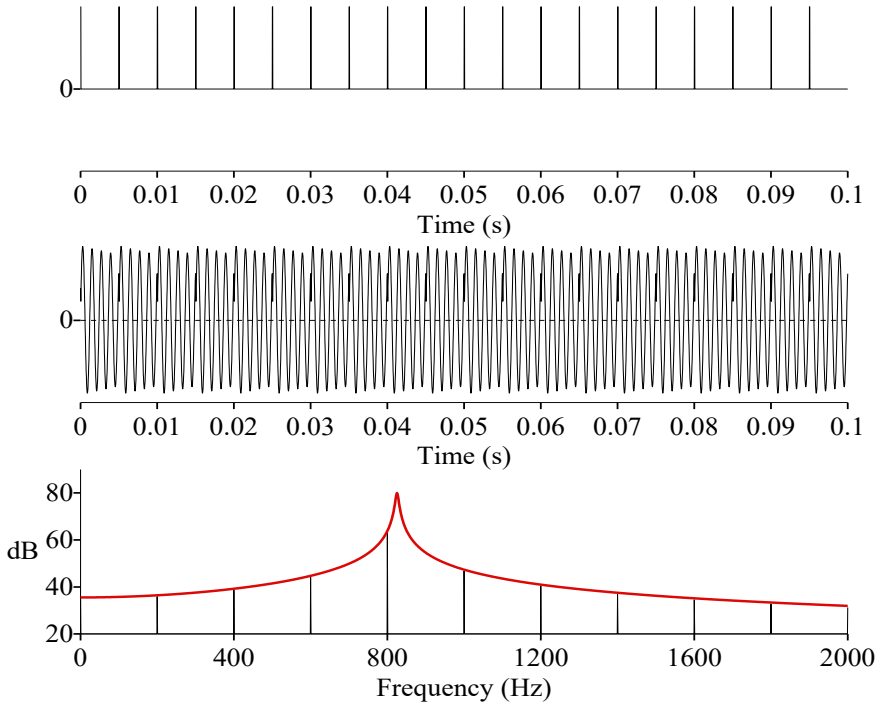


Fig. 8.7. Same as fig. 8.6 but now the resonator activated at high repetition rate (200 Hz). The harmonics in the spectrum (bottom) are so wide apart that the filter function (added in red) cannot be approximated accurately.

The demo also plays the sound when the pulse is repeated with 200 Hz, like we did with the saw tooth. The resonator frequency sounds much less clear and the 200 Hz tone with harmonics dominates, as the spectrum shows (see also fig. 8.7). Like the thought experiment we used before, we can continue to gradually lower the repetition rate until we reach zero: that means activation takes place only once. Now the spectrum is continuous so the approximated filter function form is not sampled any more but is also *continuous*. There is no interaction of a period with the next one because it will never come!

But, to exclude any influence of the excitation pulse on the damped sine, we would also like to activate the resonator with a pulse duration of 0 seconds to let it run on its own completely from $t = 0$. Of course, with a zero length the pulse has no energy left for activation. To handle this problem the concept of the **delta function** or **Dirac pulse** is invented. Its mathematical properties are defined as follows: the duration must be infinitesimally short; its surface area (duration multiplied by amplitude) must be 1. The consequence is that its amplitude must be infinitely high. In practice this is impossible, of course, but mathematics is not subject to the restraints of practical reality. For doing practical experiments we could use pulses that are very short *in relation to* the periods of the spectral components concerned. In fact, this is what happens in the demo. Later-on we will get some insight in the way we can keep the effects of applying a pulse with non-zero length negligible in practice.

In fig. 8.6 and fig. 8.7 we see that the spectral component amplitudes follow the form of the resonator filter *exactly*, only the high repetition rate of fig. 8.7 causes the spectral filter function to be 'sampled' at greater distances. Apparently, the spectral components of the repetitive pulse *itself* all have the same height: there is no 'roll-off'. (This is only theoretically so for infinitesimally short time functions, i.e. delta pulses. Any pulse duration greater than zero will cause some spectral roll-off, as we will see in section 13 about windows.)

Important to mention here, however, is that the output of a filter, activated with a once-occurring Dirac pulse is called the **impulse response**. As we already saw in DEMO8.1 the impulse response of our resonator is a damped sine wave. It defines the filter properties completely.

As a conclusion of all this, when we band filter a specific sound we can find the resulting sound by multiplying the frequency components of its spectrum by the values from the band filter function at the corresponding frequencies, and transform back to sound again, right?

No - wrong! From filter graphs like the red lines in fig. 8.6 and 8.7 the *phase information* cannot be read, just like in the case of the display of a spectrum in section 6. A 'normal' filter, like our resonator-type will have a *phase behavior* as shown by the red line in fig. 8.8. You can see that the phase angle starts with zero radians and

becomes more and more negative as the frequency increases (due to the *delay* in time). The 'fast' phase shift near the resonance frequency is characteristic for this type of filter but its explanation falls outside the scope of this book.

So, if we band filter a sound with a filter like this by spectral multiplication, we must take into account the phases of all spectral components of the sound AND the phase shifts by the filter at all frequencies of the signal's spectrum. (In fact, to the phase angle of each frequency component of the input we must add the phase angle of the filter at the corresponding frequency.)

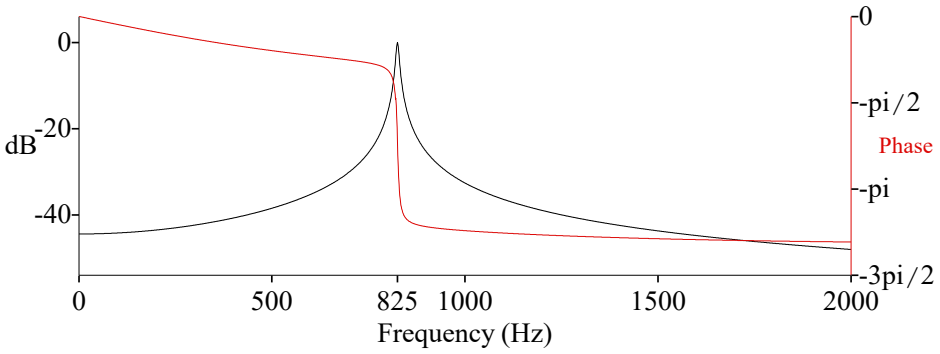


Fig. 8.8. Amplitude graph (black) and phase graph (red) of resonator filter.

In short, we need a *vectorial* multiplication, as the sine components have a magnitude and a direction (phase) as explained in section 5. The vector manipulations can be done adequately by using *complex numbers*. In Appendix II you will find an explanation of the principle behind the complex calculation. For now, it is sufficient to know that complex multiplication of spectra can be done in Praat.

What will happen if we do not bother about the phases of the filter and multiply the (complex) Spectrum object of the sound with the corresponding filter *magnitude* values? The phases of the spectral components remain the same, only the magnitudes alter. So that will not influence the resulting (amplitude) spectrum graph. With DEMO8.2 you can experience the difference. If the repetition frequency is not extremely low (say, above 60 Hz) the filtered signals sound the same. On the time scale, however, we see that the resulting waveforms of the two types of filtering differ substantially (see also fig. 8.9).

This last type of filtering (multiplying with magnitudes only) is called **zero phase filtering**. This filtering is not **causal** which means that the output can start before the signal starts! In fact, the impulse response of this type of filtering is symmetrical around the zero on the time axis. It can only be done in analyzers where a signal can be put in a memory, like a computer. In all 'real world' filtering, however, obviously there cannot be a response before the start of the input signal.

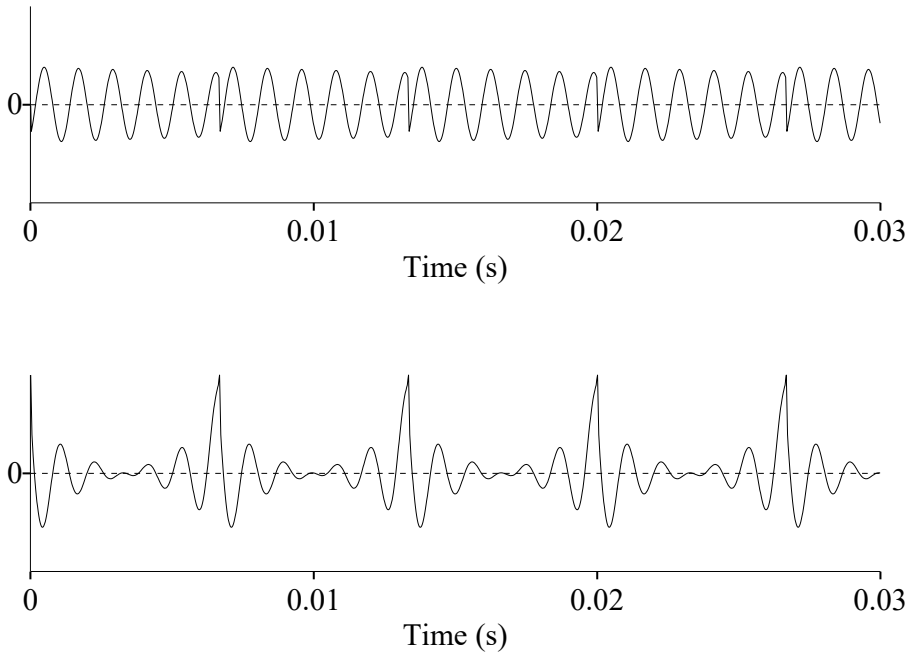


Fig. 8.9. Short pulses repeated with 150 Hz filtered with resonator filter by multiplication of their spectra. Upper graph: output when phase behavior of filter is taken into account. Lower graph: output when phase is NOT taken into account ("zero phase filtering").

Filters exist in a great variety of types. Main types of filters are **low-pass** and **high-pass** filters. The names speak for themselves. Some practical examples of the different types are depicted in fig. 8.10. For low-pass and high-pass filters too, the point where the amplitude is -3 dB relative to the maximum is used as a parameter of the filter. The frequency where this is the case is called the **cross-over frequency** (f_c).

The band-pass example is different from our resonator filter: the pass band is much wider. In this example the 3-dB bandwidth (B) is 1000 Hz with a center frequency of 1400 Hz whereas the B of our resonator example is only 8 Hz with a center frequency of 825 Hz.

This type of wider band-pass filters can be realized by combination of a high-pass and a low-pass filter. Then the output of one of the filters forms the input of the other one (**cascade** configuration, see fig. 8.11). The cross-over frequency of the low-pass filter must be higher than that of the high-pass filter (otherwise there is no output of any importance at all!). The difference between the cross-over frequencies then is the bandwidth. A band suppression filter (sometimes called a **notch** filter) can also be assembled by using a low-pass and a high-pass filter. The configuration then cannot be a cascade: the input signal must be fed to both filters simultaneously and the filter outputs must be added together. Now the cross-over frequency of the low-pass filter must be lower than that of the high-pass filter (**parallel** configuration, see fig. 8.11).

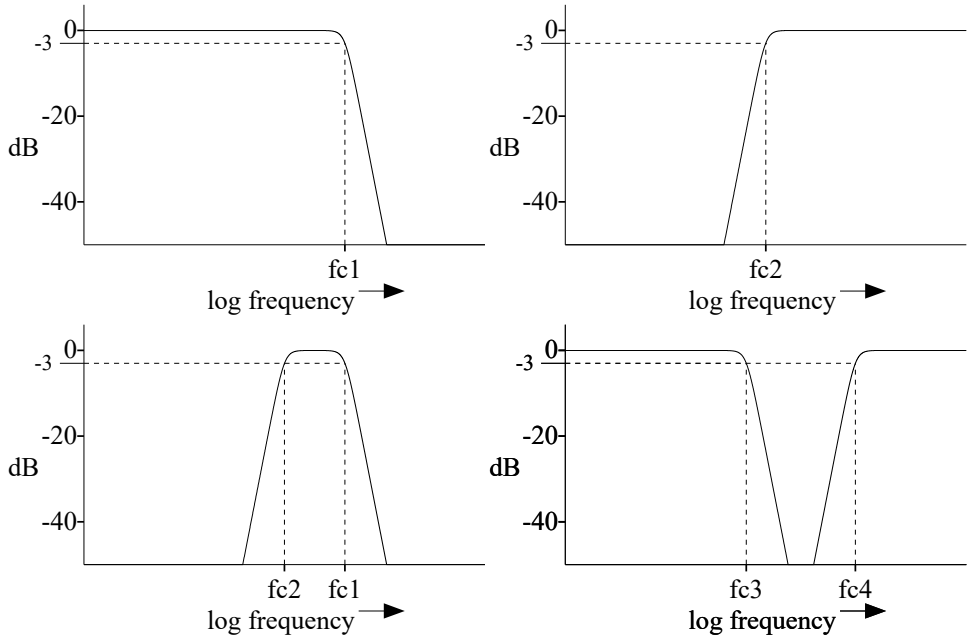


Fig. 8.10. Examples of low-pass, high-pass, band-pass and band suppress ('notch') filters.

What can be seen in fig. 8.10 is that the 0-dB reference level is positioned at the maximum of the graphs. Actually, the reference is determined by the output signal amplitude being equal to the input signal amplitude so that the ratio = 1, i.e. when there is no amplification or attenuation. Although resonators are able to 'amplify' the input amplitude at their peaks (think about the swing that can reach much greater deviations than the push movement itself), we will still scale their peaks to 0 dB for our purposes. That is only a matter of scale agreement; the forms of the filter functions obviously remain the same. Usually, graphs of filter functions are presented without the phase properties: only the ratios of output amplitudes and input amplitudes, their **gains**, are given.

Note the distinction of the reference of sound spectra and that of filters: sound spectra dBs refer to a *signal* level (the hearing threshold) whereas filters refer to their *output/input* ratios. Filters do not generate signals!

As you noticed, the frequency scales in fig. 8.10 are also logarithmic, just as the vertical (amplitude) axes. Although there are some exceptions, this is commonly applied for audio (= low frequency) filters, as the human frequency perception is logarithmic as well. In addition, because the functions of physical (analog) filters are usually exponential, displaying their functions on logarithmic scales has the effect that, at least some distance from the cross-over frequencies, the lines are quite straight, as you can see in the graphs. Therefore, a logarithmic frequency scale, together with a logarithmic amplitude scale, makes it possible to define the steepness (**slope**) of the low and high-

pass filters with a simple variable: the number of dBs per octave (**dB/oct**) as the steepness expressed in this manner of most types of linear filters is constant. In the figure, therefore, you see that the slopes are all rather straight lines. As you probably know, one octave means a factor 2 in frequency. The low and high-pass filters from fig. 8.10 have slopes of 72 dB/oct. For the purposes of this book, however, we generally use linear scales for the frequency, which is best suited for the explanation of the subjects the book covers.

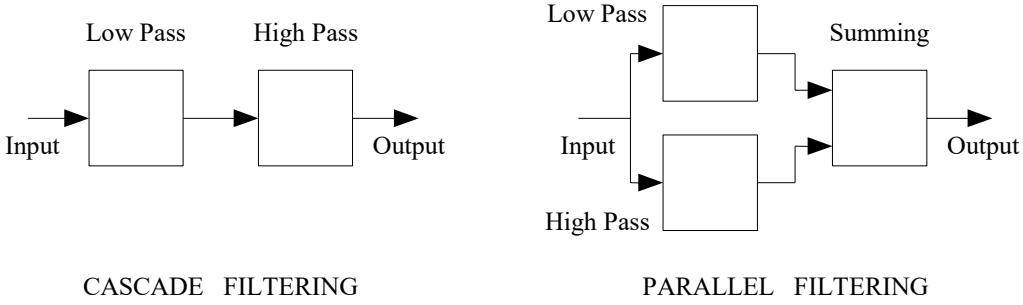


Fig. 8.11. Combination of low-pass and high-pass filters can form band-pass filters (left) or band suppression filters (right).

The slopes of the high-pass and low-pass filter examples are very steep (as said, 72 dB/oct) compared to the slopes of our resonator (the resonator slopes far from the resonance frequency are only 6 dB/oct). Increasing the steepness of a filter can be achieved by applying higher order terms in its mathematical transfer function. For many filtering purposes (i.e. selecting specific frequencies from a band of frequencies, suppressing interfering frequencies, etc.) one prefers a high steepness for obvious reasons, and to this respect the modern techniques have no limitations in practice, so, then why not make them extremely steep? The answer is that the phase behavior of a filter depends on the steepness: a very steep filter causes great phase angles between output and input, which imply *long response delay times*. As an example: the delay at the cross-over frequency of the low-pass filter of fig. 8.10 is almost 2.5 complete periods! It means that filters for low frequencies have longer delay times than high frequency filters. It's all a consequence of the relation $f = 1/t$ as we will see throughout the whole book.

The various delays of different frequency components of the spectrum caused by the phase behavior of a filter produces an unwanted *time smearing*, the **phase distortion**. Several types of filters have been developed to overcome this effect. Their phase angle increases or decreases linearly with increasing frequency so that the time delay is constant for all frequencies: the **linear phase** filters.

The mathematics available for designing practical filters is very comprehensive and the electronics to realize the theoretical developed filters in practice is very advanced. Later we will learn about the digital representation of signals (section 17) and digital filtering (section 18) which offer great possibilities, like zero phase *filtering*, an example of which we met already in fig. 8.9. The slope steepness and stability that can be reached with digital techniques go far beyond the possibilities in the analog area. However, every filter forms a compromise between slope (or 'selectivity') and response time. This compromise, as you will understand, is no matter of design limits but of the laws of physics.

The field of filtering is very broad and many good books about filters are available. So, at this point we will leave the subject here as the overall principle of filtering should be clear. In section the practical part of the book some special types of filters are described.

9. Continuous spectra

From DEMO8.1 we learned that the sound of the resonator can best be heard when the repetition rate is very low. In that case we hear the same sound of the once-activated resonator repeated. When we look at the output waveform of our resonator activated with a low repetition rate we can see a series of separate damped sine waves, see fig. 8.6. The excitation rate may be low enough for the damped sine waves to come to practically zero amplitude before a new excitation occurs. If we continue lowering the excitation rate, the waveform of the damped sine waves practically does not change any more, only the *time* between the damped sine bursts increases. The central frequency and the damping of the resonator remain constant which means that the peak position and peak width do not change: the filter function remains unaltered. It is approximated, however, with more and more accuracy. We can apply this increase of time to an already existing (recorded) sound of a resonator excited at a certain rate by 'artificial' insertion of time at the end of each period, just before a new excitation period. Then, the spectral lines of the output spectrum are shifted nearer to each other because the F_0 decreases:

$$F_0' = \frac{1}{T_0 + T_A} \tag{9.1}$$

Here F_0' is the new F_0, T_0 is the period time of the excitations and T_A is the added time to the period. (The overall power of the signal decreases by this manipulation so that the spectral amplitudes are all attenuated according to the factor $(T_0+T_A)/T_0$. On a log scale all components shift down the same number of dBs. Consequently, after scaling the graph, the shape of the spectrum remains the same.) Again, as in our thought experiment in the beginning of section 8, the limit is reached when $F_0 = 0$, i.e. equivalent to the resonator activated only once. We then have our impulse response back again. (Theoretically it is an approximation, not the real impulse response, because the exponential function is not exactly zero at $t = T_0$ and is forced to become zero there. The theoretical exponential never ends.)

When the repetition rate approaches zero the number of 'spectral lines' in the frequency domain tends to infinity. Fortunate enough, there is no need to calculate an infinite number of Fourier components: the mathematics provides us with the calculus of integrals. Instead of a Fourier *series* we use the **Fourier integral**. In appendix II.3 the Fourier integral is calculated, using *complex numbers*. For people who do not like mathematic calculus it is sufficient to know that the spectral functions of once-occurring *time events* can be estimated using integrals, offering spectra with infinite numbers of points, or in other words, **continuous spectra**. As a consequence, the displays of spectra of single time events will not contain spectral lines but are continuous graphs.

Unfortunately, Praat does not provide the calculation needed for the Fourier integral. The funny thing is: we do not need it! If we make the F_0 sufficiently low we can

approximate the continuous spectrum well enough. In fact, DEMO9.1 uses this trick (see also fig. 9.1). It generates damped sine waves during sufficient time for the

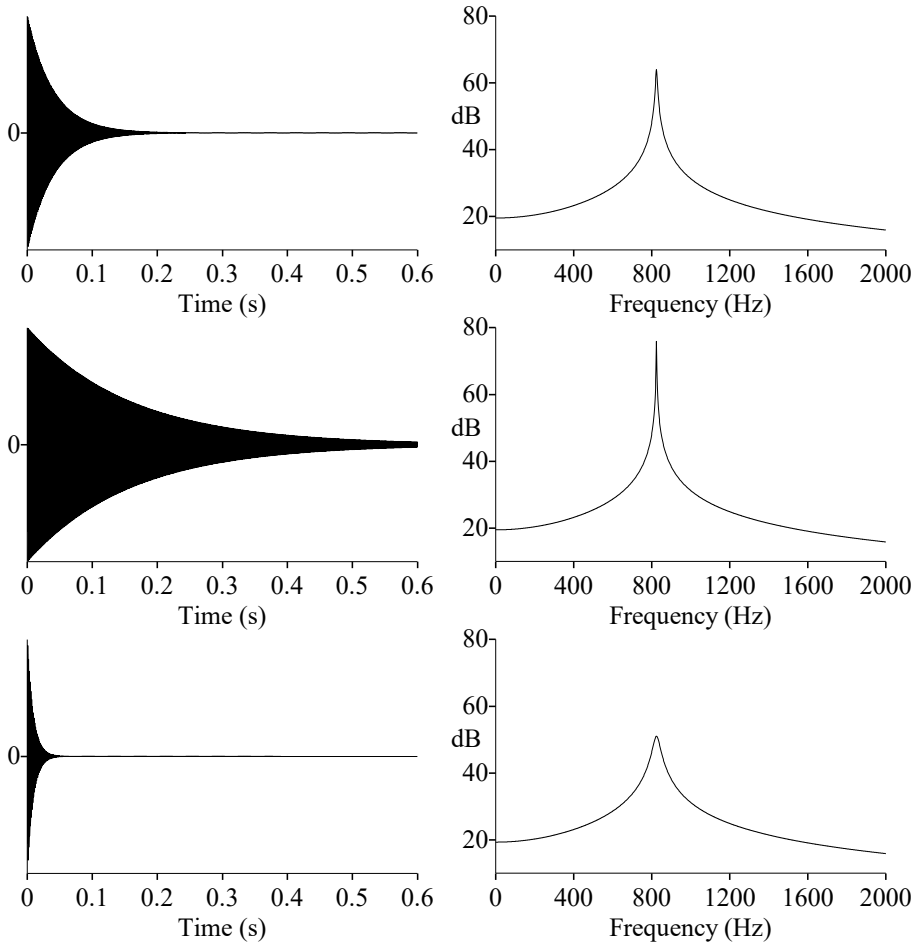


Fig. 9.1. Outputs of once-activated resonators with various damping factors and their spectra. Top: Bandwidth B = 8 Hz. Center: B = 2 Hz. Bottom: B = 35 Hz.

amplitude to become negligibly low, and extends the sounds to 1.25 s, even when the damped sine wave has 'long' before vanished already. The spectral points of the DFT, therefore, are positioned at 0.8 Hz apart. In addition, Praat's spectra *interpolate* linearly between the points, which fills the space between the spectral lines so that the graph looks very smooth and continuous.[1]

So, DEMO 9.1 generates some possible outputs of resonators when activated once. When we simulate a low *damping* (which can be seen as loss factor), the damped sine wave lasts much longer and its (continuous) spectrum shows a narrower peak, i.e. a smaller bandwidth, than the first spectrum (top of fig. 9.1). When we do the opposite: simulate a high loss factor then the tone comes much faster to an end and its spectrum

[1] As the DFT computes components that are 1/T apart in frequency, where T is the duration of the time signal, the widths of the frequency steps are called **bins**, to indicate a range instead of a point.

becomes much broader (bottom of fig. 9.1. In all cases the "resonance" frequency, however, remains the same (we hear the same tone).

We see that the spectrum of the output of a once-excited resonator (i.e. the impulse response) shows an almost symmetrical frequency area around its resonance frequency. (Later on, we will learn that the spectrum of a damped sine wave is not exactly symmetrical. For now, it is sufficient to regard it as symmetrical if we do not look too far from its peak.) The longer the decay time of the resonator, the narrower the spectral area and vice versa. When you think of increasing the decay time to infinity, the spectral width decreases to zero: the spectral line of one continuous sine wave is back again! The limit in the other direction can also be explored by reasoning: when you let the decay time tend to zero, the spectral width tends to infinity and we cannot speak of a sine wave any more. Indeed, the "spectrum" of an infinitely short pulse that still has energy (the Dirac pulse), has an infinitely high bandwidth and thus is a horizontal line from zero to infinitely high frequency.

As we have seen in section 7, the function that defines the decaying sine wave along the time axis is:

$$x(t) = A_0 e^{-\alpha t} \sin(\omega t), \;\; t \geq 0, \alpha > 0 \tag{9.2}$$

which is a multiplication of a continuous sine wave with a decaying exponential. In practice the damping α of a (mechanical) resonator is not independent of its center frequency ω which has the effect that, roughly speaking, low frequency resonators produce longer tones than high frequency ones, when excited once. In our computer, naturally, we are free to generate these kinds of damped sine waves with independent variation of the parameters α and ω. Consequently, we see that if we increase ω, the area underneath the enveloping exponential curve is 'filled-in' as it were with more and more periods of the central frequency sine wave, whereas the enveloping curve itself remains unaltered. Alternatively, when we increase α then obviously the enveloping curve becomes shorter while the period lengths within the curve remain constant.

For periodical activated resonators at a high repetition rate the spectral harmonics may be too far apart for a proper approximation of the underlying continuous spectrum. In that case we can very well use the time insertion trick described above and may reach an approximation of the filter function that is sufficient for measuring its resonance frequency to a reasonable accuracy.

To apply this time insertion later on to an already recorded repeatedly excited resonator signal, naturally we do not need to insert time into all repetition periods. As fig. 9.2 explains, we simply have to isolate one period from the train of periods, add a fair length of zero sound to it and look at its spectrum. You will know the reason why this is so from section 6: the DFT of one period of a periodical signal is equal to the DFT of an integer number of periods. (This is not so when the FFT is used: it lengthens the selected signal with some time.)

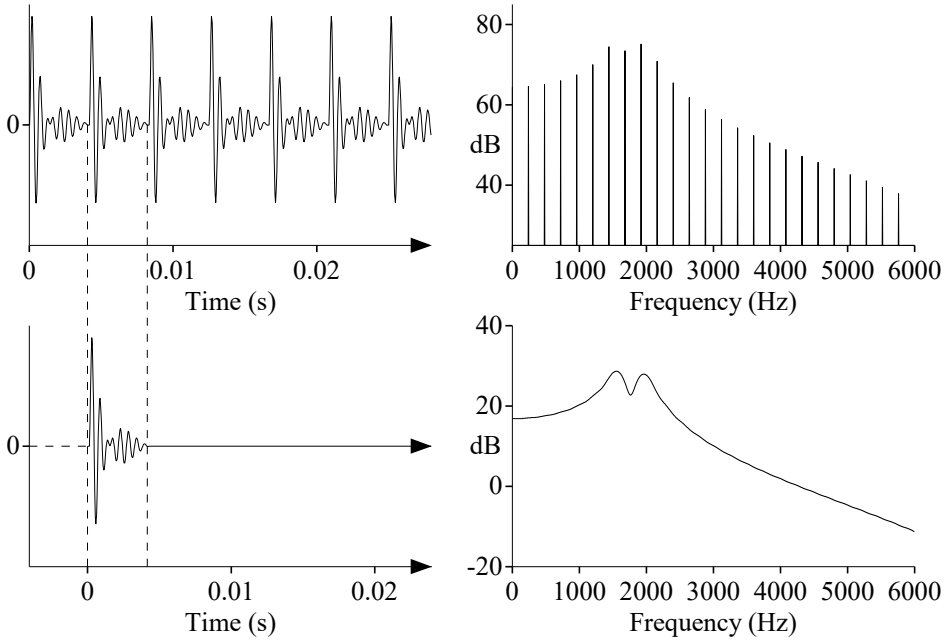


Fig. 9.2. Top: summed outputs and spectrum of two different resonators activated by 240 Hz pulse series. Isolating one period and insertion of time at the end (bottom) approximates the filter function very well.

The example of fig. 9.2 deals with two different resonators activated by the same source (i.e. the repetitive short pulses). The signal is the sum of the two resonator outputs. (Practical sound signals often contain a set of different resonances instead of only one.) From the line spectrum the two different resonance frequencies cannot be read very accurately. The continuous spectrum displays the two resonance frequency positions very clearly.

With this trick we are able to reveal the envelope (filter) function quite well, not really limited by the spectral line distance. Of course, there is some pitfall here... The period of the excitation rate must be sufficiently long for the exponential decaying amplitude to reach practically zero but the amplitude at the end of the original period could still be substantial, especially when the original repetition rate was high. The end of the decaying wave and the start of the new one then interact with each other. In fact, the continuous sine wave is not multiplied by a real exponential function but by a *truncated* one!

Obviously, the amplitude at the end of the period depends on the repetition rate as well as on the damping of the resonator. Increasing the damping (α) and decreasing the repetition rate (which increases the period T) both lower the end amplitude. In the spectra of fig. 9.1 you can see that a higher α causes a broader peak, which will tolerate greater distances between the spectral lines, and therefore higher repetition rates, to approximate the filter graph.

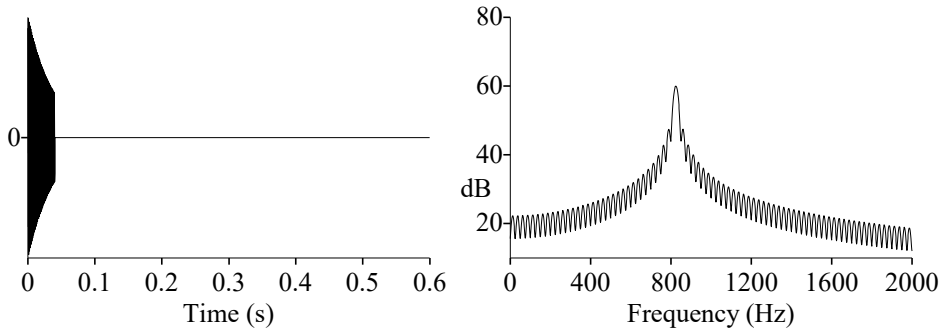


Fig. 9.3. Truncated damped sine wave and its spectrum.

With higher end amplitudes things are far more complicated, as you can see in fig. 9.3. In this case the truncated time function can be regarded as a damped sine wave multiplied with a rectangular function. What is the relation between these multiplied time functions and their spectra? Answering that question will require a separate and important section 13 about w*indowing*. First, however, we must take an important side step.

10. Multiplying signals

In section 9 we saw that the damped sine wave can be seen as a multiplication of a continuous sine wave with the decaying exponential. To get some insight in what happens spectrally when we multiply two different time functions we will start with a simplified case first: let's multiply two sinusoidal waves with different frequencies. Next, we can apply this knowledge to signals consisting of more sinusoidal components. The box called **MULTIPLICATION OF TWO SINUSOIDAL WAVES** shows that multiplication of two cosines of different angles ($\omega_1 t$ and $\omega_2 t$) results in the sum of two cosines, one of an angle $\omega_1 t + \omega_2 t$ and one of an angle $\omega_1 t - \omega_2 t$, resulting in the following waves:

$$x_A(t) = 0.5 \cdot A_1 \cdot A_2 \cos\{(\omega_1 + \omega_2)t\} \quad (10.1)$$

$$x_B(t) = 0.5 \cdot A_1 \cdot A_2 \cos\{(\omega_1 - \omega_2)t\} \quad (10.2)$$

MULTIPLICATION OF TWO SINUSOIDAL WAVES

From our school trigonometry we may possibly remember the following formulas:

$$\cos(\alpha + \beta) = \cos\alpha \cdot \cos\beta - \sin\alpha \cdot \sin\beta$$

$$\cos(\alpha - \beta) = \cos\alpha \cdot \cos\beta + \sin\alpha \cdot \sin\beta$$

Addition of the left sides of the equations must be equal to the addition of the right sides:

$$\cos(\alpha + \beta) - \cos(\alpha - \beta) = 2\cos\alpha \cdot \cos\beta$$

which can be rewritten as:

$$\cos\alpha \cdot \cos\beta = \frac{1}{2}[\cos(\alpha + \beta) + \cos(\alpha - \beta)]$$

To apply this to sinusoidal **waves** of amplitudes A_1 and A_2 we substitute $\omega_1 t$ and $\omega_2 t$ for α and β:

$$A_1 \cos(\omega_1 t) \cdot A_2 \cos(\omega_2 t) = \frac{1}{2} A_1 \cdot A_2 [\cos\{(\omega_1 + \omega_2)t\} + \cos\{(\omega_1 - \omega_2)t\}]$$

As the choice of the phase is irrelevant for the power spectrum when dealing with continuous sinusoidal waves we use cosines instead of sines to simplify the calculation. Had we used sines, then the formula would only differ as regards phase shift constants of $\pi/2$ radians:

$$A_1 \sin(\omega_1 t) \cdot A_2 \sin(\omega_2 t) = \frac{1}{2} A_1 \cdot A_2 [\cos\{(\omega_1 - \omega_2)t\} - \cos\{(\omega_1 + \omega_2)t\}]$$

We can see that each result consists of one sinusoidal wave having a frequency which is the sum of the frequencies and one sinusoidal wave with a frequency which is their difference. The original frequencies have disappeared.

We can see that the original frequencies have disappeared and that only waves with the sum frequency and difference frequency have emerged. Their amplitudes are equal (both $A_1 \cdot A_2$), regardless of any amplitude difference between the original waves. (The use of cosines instead of sines only means that a different position of the time origin is chosen, which simplifies the formula somewhat. Conceptually the sinusoidal waves

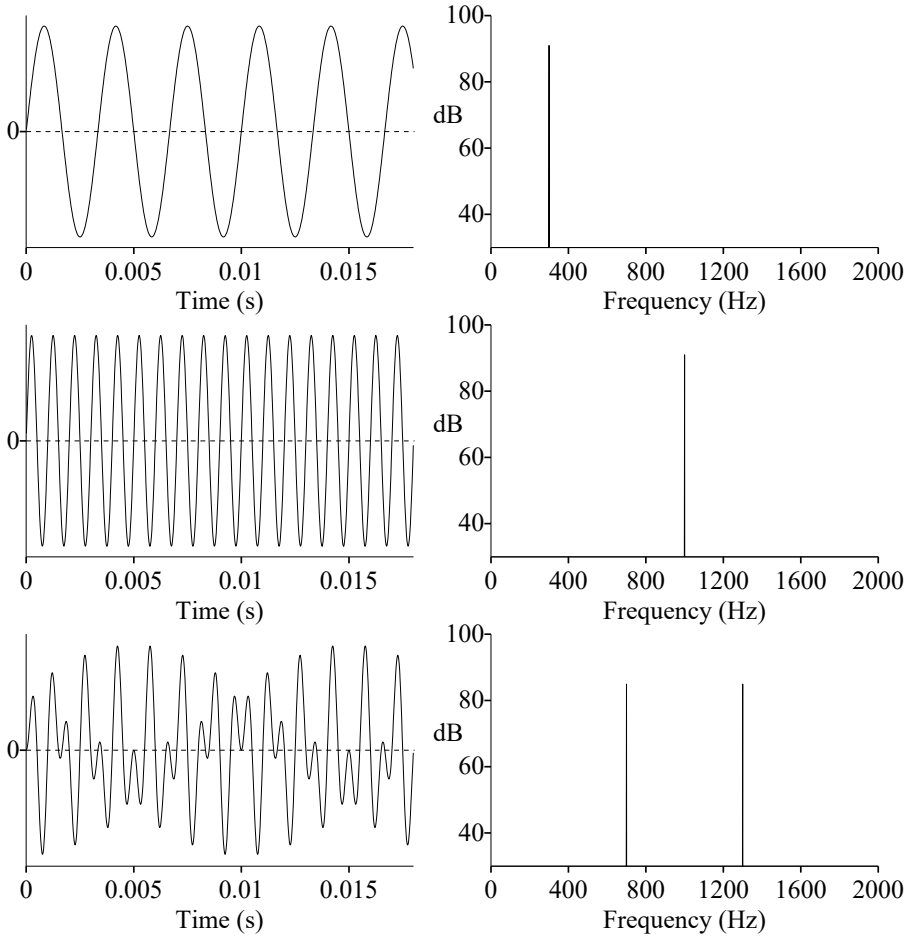


Fig. 10.1. Multiplication of 300 Hz and 1000 Hz sine tones produces only a sum and a difference frequency.

have infinite duration: they have always existed and will exist forever, and their position of origin is therefore not relevant.)

DEMO10.1 multiplies a sine wave with a frequency of 300 Hz with a sine wave of 1000 Hz (see also fig. 10.1). It plays the two original waves as well as the result one after each other, and displays the spectra. You will notice that in the resulting spectrum there are only a sum and a difference component (1300 Hz and 700 Hz respectively) while the original frequencies have disappeared. (This multiplication is an example of a *non-linear* process; that's why frequency components emerge which were not present originally. See the remarks about linearity mentioned in the box called **LINEAR SYSTEMS** in section 8.)

Our next step is to multiply a sine wave with the triangular wave from section 4. In fig. 10.2 we can see that all 10 harmonics of the triangle spectrum are shifted to the right of the single frequency of 5000 Hz of the sine, whereas a mirrored copy of the triangle spectrum is placed to the left of this frequency. The sine frequency itself has

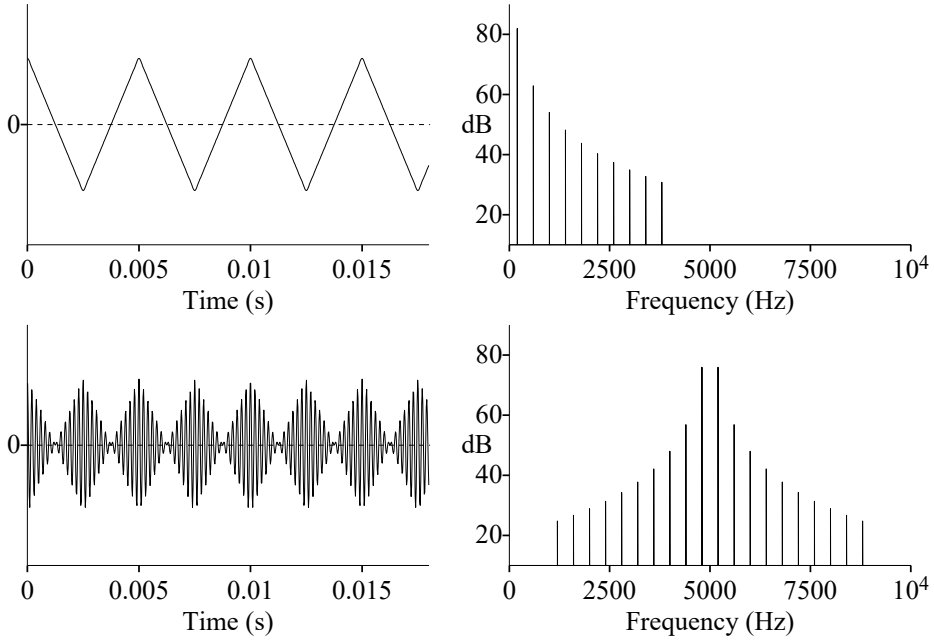


Fig. 10.2. Triangle wave from section 4 (top) multiplied with a sine wave of 5000 Hz (bottom), and spectra.

gone, just like all original frequency components of the triangular wave. These frequency shifts should not surprise us because the sums and differences must occur for each of all frequency components of the waves.

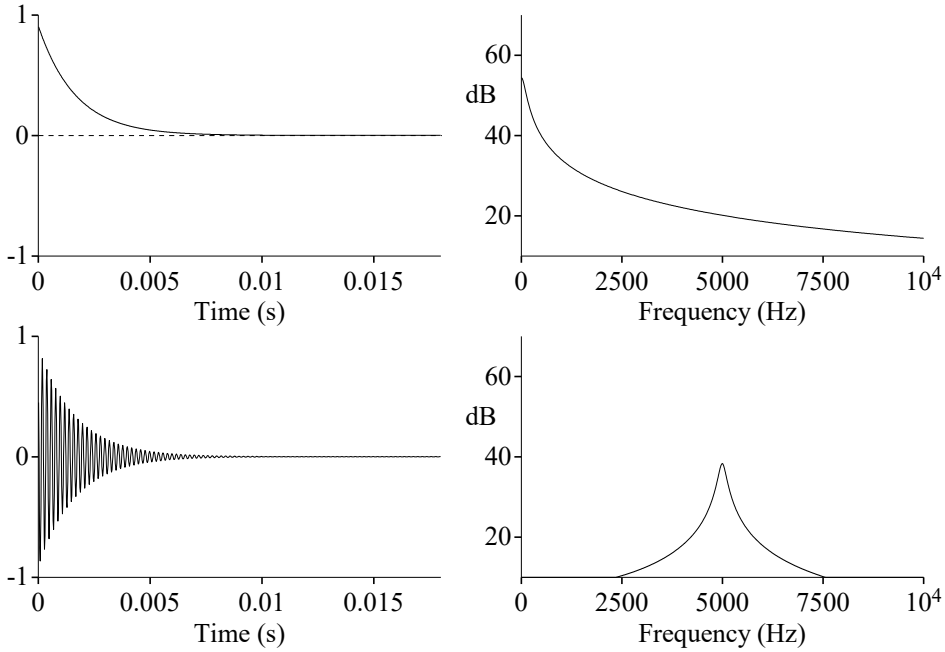


Fig. 10.3. Damped sine wave by multiplication of exponential waveform with sine wave of 5000 Hz, and spectra.

The same line of thought can be applied to the once-activated resonator: the damped sine wave output can be seen as the multiplication of a sine wave with a single exponential function (see formula 7.3 of section 7 about the resonator). As you may know from fig. 8.6 and 8.7 in section 8, the spectrum of this exponential function has the same shape as the envelope of the spectrum of the repetitive exponential but has an 'infinite number of spectral lines'. In other words: it is *continuous*. Fig. 10.3 shows the waveforms and spectra of this case. Again, we can see that around the 5000 Hz point on the frequency axis the (infinite number of) sum and difference frequencies are placed. Thus, when we multiply a signal consisting of an infinite number of sine components, and which therefore has a continuous spectrum, with a single sine wave, this continuous spectrum is shifted and mirrored around the single sine frequency in the same way as in the case of the separate harmonics that are shifted and mirrored around the 5000 Hz of fig. 10.2. Naturally, the result is also a continuous spectrum so the pertaining time function (i.e. the damped sine) is also once-occurring.

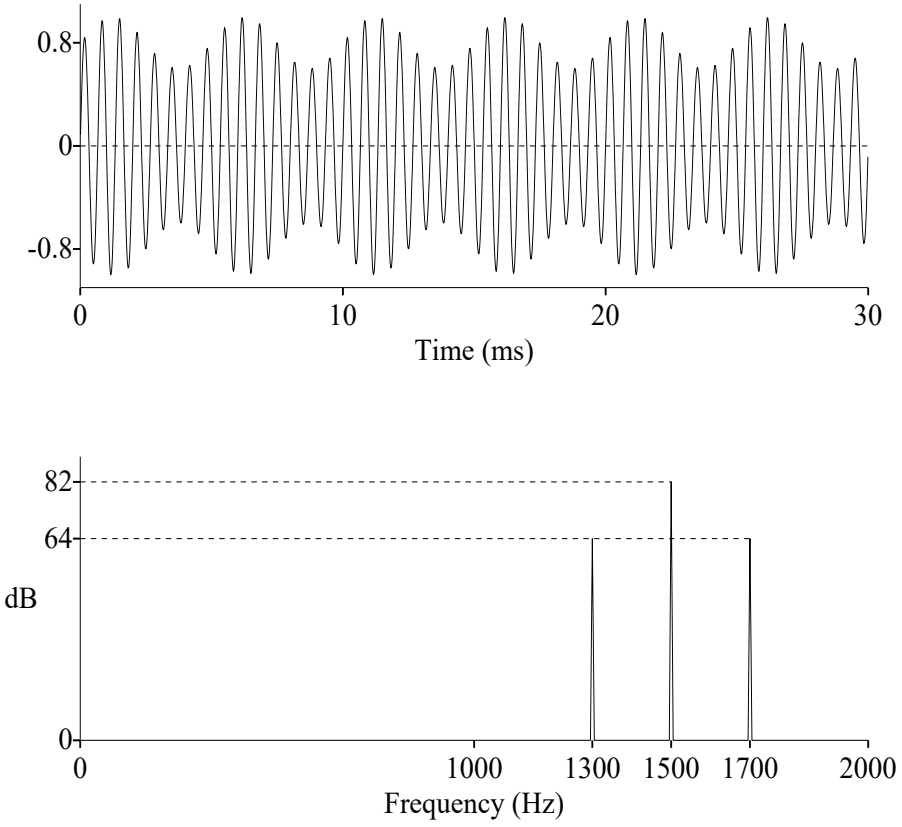


Fig. 10.4. Top: waveform of sine wave of 1500 Hz amplitude modulated with sine wave of 200 Hz. Bottom: its spectrum.

When we multiply sine waves all original frequencies will disappear. Suppose that we want to multiply two sine waves and preserve one of the frequencies. We could do that by adding the wave to be preserved to the result:

$$f(t) = A\sin(\omega_C t) \cdot B\sin(\omega_M t) + A\sin(\omega_C t) \tag{10.3}$$

which can be written as:

$$f(t) = \{1 + B\sin(\omega_M t)\} \cdot A\sin(\omega_C t) \tag{10.4}$$

This can be seen as a substitution of the constant amplitude A of the function $A\sin(\omega_C t)$ by the function $A\{1 + B\sin(\omega_M t)\}$. When ω_C is much higher than ω_M, like in fig. 10.2, it is easy to see that the value of amplitude A is varied in a sinusoidal way, so that formula 10.4 means **amplitude modulation** of one sine wave (the **carrier** ω_C) with a second sine wave (the **modulation** ω_M). Fig. 10.4 shows an example of this case. The frequency of the carrier is always much greater than the modulating frequency.

If the modulation has zero amplitude (B=0) then what we have left is the unaltered carrier sine wave. In fig. 10.4 the amplitude of the carrier is 0.8; the amplitude of the modulation is 0.2. The **modulation depth** is therefore 0.2/0.8 = 0.25. In practice, this modulation depth is always less than 1. On the log scale you can see that the modulation components in this example are 18 dB lower than the carrier component. The amplitude of the modulation is 1/4 of the carrier amplitude which amounts to -12 dB. In the spectrum, however, the modulation is presented by two mirrored components (just like

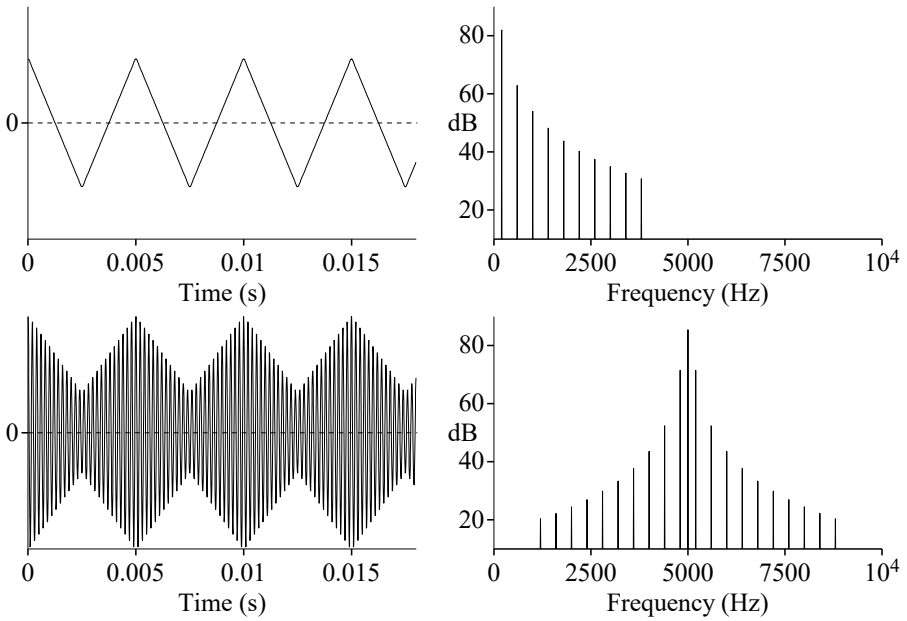


Fig. 10.5. Modulation instead of multiplication: triangle wave of 200 Hz from fig. 10.2 applied as modulation of a sine wave of 5000 Hz, and spectra.

we saw when we *multiplied* sine waves) each with half the amplitude so that the modulation components in the spectrum are lowered by another 6 dB. You will see that the only spectral difference of amplitude modulation with multiplication is the presence of the carrier frequency in case of AM whereas this carrier is completely absent in case of multiplication.

Naturally, instead of one modulating sine wave it is possible to apply a whole range of modulating sine waves with different frequencies. Again, they are all mirrored around the carrier frequency. Together they are called **side bands**. As an example, see fig. 10.5 where we can see that all 10 harmonics of the triangle spectrum are shifted and mirrored w.r.t. to the single frequency of 5000 Hz of the sine. All original frequency components of the triangular wave have gone, while the sine frequency itself remains present now because of the amplitude modulation applied instead of pure multiplication.

What purpose can this Amplitude Modulation serve? Instead of a fixed set of frequency components as the modulation, the varying components of speech or music can be applied. This makes it possible to modulate a carrier wave with speech or music signals, which is what happens in AM radio transmitters. The frequency components of the modulation are then changing all the time according to the varying spectrum of the speech or music sound (see fig. 10.6). The carrier frequency applied (f_C) is very much higher than the modulation frequencies (for example 10^6 Hz) to enable the transmission of the signal over great distances by using a tuned antenna for the transmitter. The great advantage of modulation is the possibility to apply many carriers with different frequencies so that every transmitter uses its own frequency part without interfering with other transmitters. With your radio receiver you can tune its band filter to the transmitter of your choice. The receiver has a device to extract the original speech or music from the modulated signal (the *detector* or *demodulator*). A full discussion of demodulation systems falls outside the scope of the book. However, one demodulation method can be deduced from what we have learnt so far: multiply the modulated signal with a sinusoidal wave with exactly the same frequency and phase as the carrier. Filter out the sum frequencies with a simple low-pass filter and, hey presto! the original modulation signal emerges.

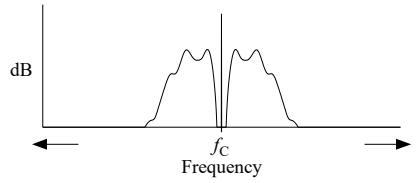


Fig. 10.6. AM radio transmitter signal. The varying spectrum of speech or music emerges as mirrored side bands.

Because the two side bands carry exactly the same information, it is possible to suppress one side band so that the transmitter occupies only half the bandwidth. This creates more space for other transmitters in the frequency range. This technique is called single side band (SSB). (You may ask: why not suppress the carrier as well? In fact, the carrier component cannot be missed because it is needed for the demodulation.)

There exist more modulation methods, like frequency modulation (FM) of the carrier, but we will leave this subject to be explained by the numerous books on radio and high frequency techniques.

11. ‘Negative frequencies’?

Another thought experiment: let's multiply our periodical triangle wave again with a sinusoidal wave, but this time we will let the frequency of the sinusoidal decrease gradually. The effect on the spectrum is that the position of the triangle wave spectrum shifts to the left.

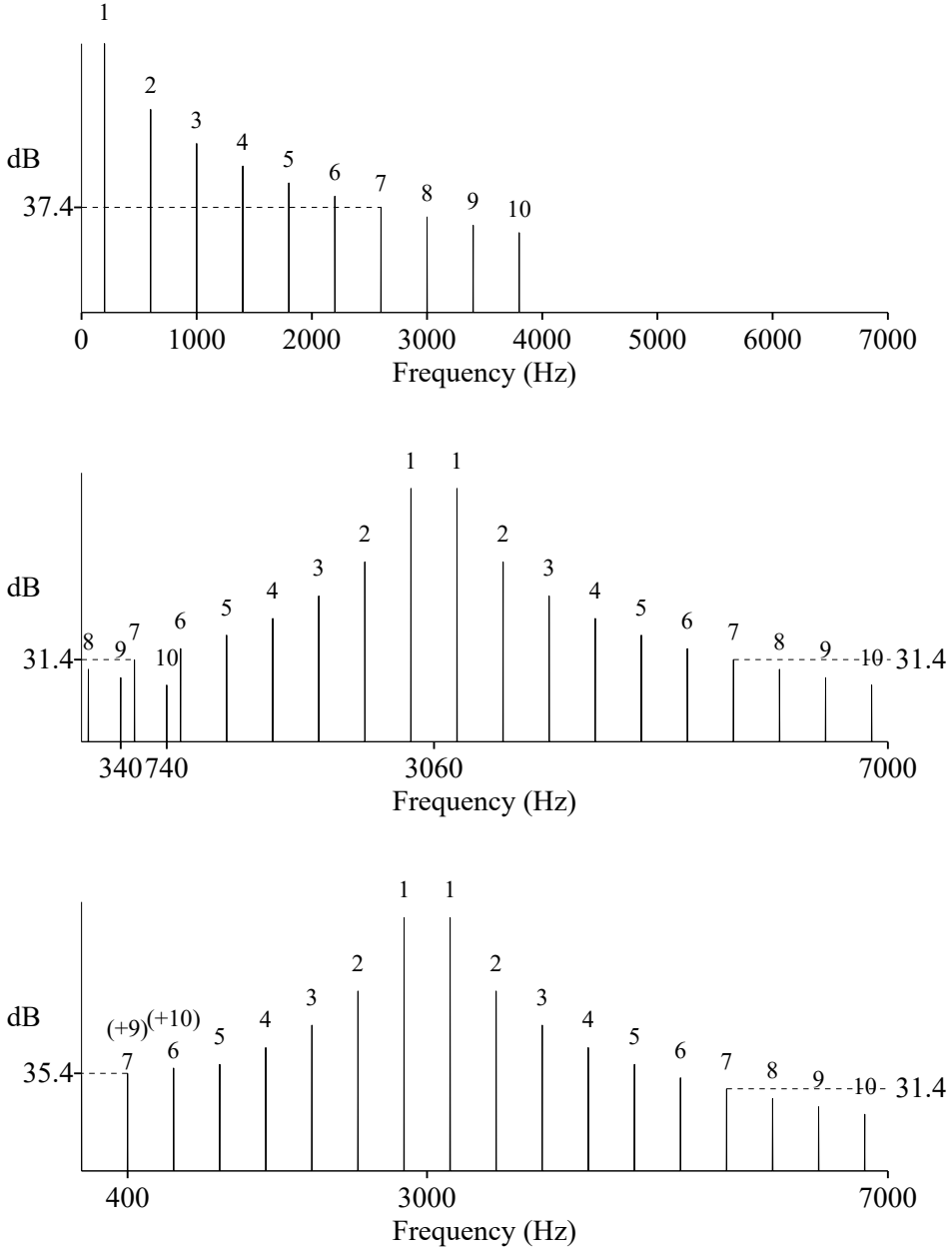


Fig. 11.1. Spectra of triangle wave (top), wave multiplied with 3060 Hz (center) and wave multiplied with 3000 Hz (bottom). In the bottom spectrum the components 9 and 10 coincide with components 7 and 6 respectively.

What happens when the position of a mirror harmonic passes zero frequency? We can see that in fig. 11.1 where the triangle sound has been multiplied by a 3060 Hz sinusoidal wave. (In fact, we here use *cosines* instead of sines which, as said before, is only a matter of choosing the zero point in time. Using sines instead of cosines in this procedure will be dealt with separately.) When we subtract the 9th and 10th component's frequencies, being 3400 Hz and 3800 Hz, from 3060Hz we get -340 Hz and -740 Hz respectively. They 'fold back' into the positive spectrum, of course, as there is no such a thing as a negative frequency... But now we multiply our triangle wave with a 3000Hz wave, so that the components which have been "folded back" coincide exactly with other components. Look at the 7th component to the left of the sine frequency (bottom part in fig. 11.1). This component is now positioned at 400 Hz. It should have a value of 31.4 dB, like all spectra showed earlier, but now it is increased to 35.4 dB, whereas its counterpart at the right of the sine frequency boundary still is 31.4 dB. The only possible cause of this phenomenon is the 'folding back' of the 9th component which otherwise would be - 400Hz!

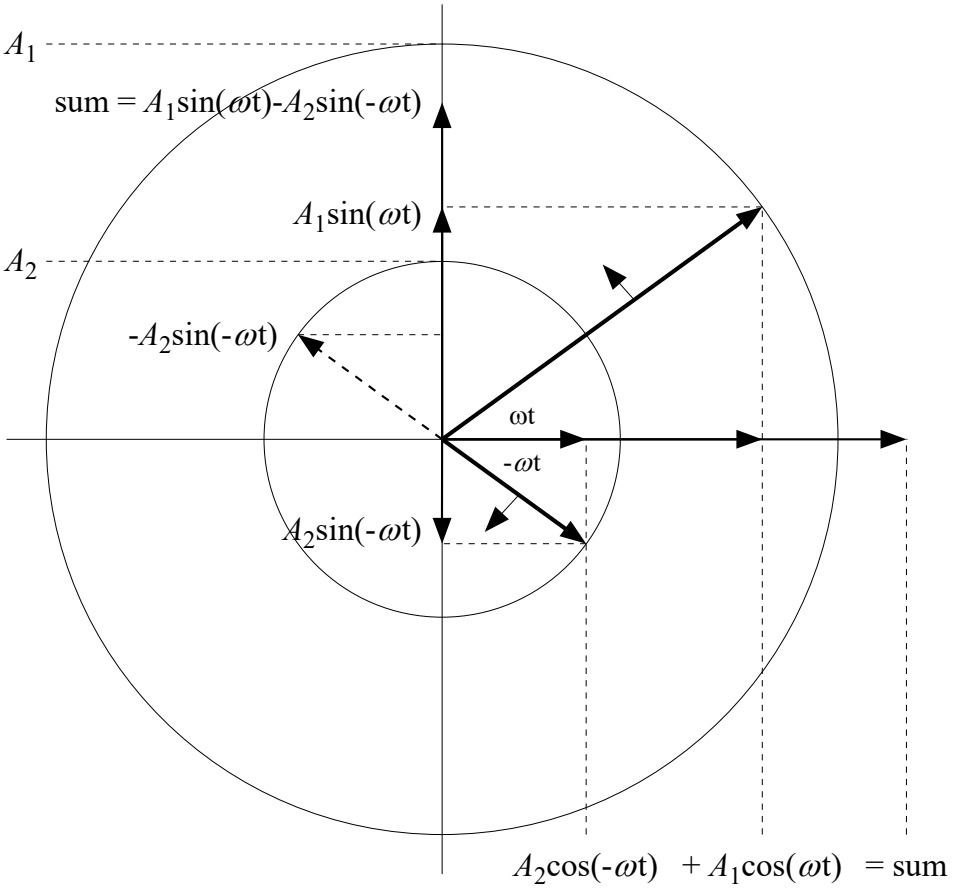


Fig. 11.2. Two sine waves as vectors rotating in opposite directions. Cosines are added on the x-axis, sines are *subtracted* on the y-axis.

If we look at the sine wave construction with the rotating vector again from section 5 (fig. 5.1) we realize that a negative ω means that the angle *decreases* with time and causes the vector to rotate in the *opposite direction*, as if the sine wave is played back to front. But what if we play one sinusoidal wave in 'normal' direction *and a second one* in the opposite direction? Let us look in some more detail at the construction of, in this case, the cosine wave. In fig. 11.2 we can see that cosines are projected on the *horizontal* axis. If we let a second vector rotate in the opposite direction then this is also projected on the horizontal axis, and in the same direction. Therefore, if we add these vectors, which we should do when the component which is folded back coincides with the other component, we get one sinusoidal wave with an amplitude which is equal to the *sum* of the individual amplitudes.

If we convert the values of the 7th and 9th component values into linear ones (by using the formula 2.5 from section 2) we can see that the 7th component of 31.4 dB equals an SPL value of 0.74 mPa (millipascal) and the 9th component of 26.7 dB, which folds back, has an SPL of 0.43 mPa. The sum then is 1.17 mPa. Converting it back to dBs again (with formula 2.5) produces 35.4 dB, which is exactly the value we can see at the 400 Hz position!

If the harmonics had been sines instead of cosines, the amplitudes should have been *subtracted* instead of added, which follows logically: two vectors rotating in opposite directions always produce projections on the *vertical* axis in *opposite* directions, as can be seen also in fig. 11.2. Therefore, in order to add them up, the projection of the backward rotating vector should be *subtracted* from that of the forward rotating one.

It seems *as if negative frequencies do exist*!

The lower the frequency of the sinusoidal wave the more components of the triangle spectrum will fold back. What if we multiply the triangle wave with a cosine wave when its frequency has become zero (which means a horizontal line at amplitude 1 because cos (0) = 1)? We will then see only the right half of the former spectra because the peak is positioned at $f = 0$. We have arrived at the original spectrum of the triangular wave. But what happened to the left half? Apparently, it has folded back completely. But there is no difference between our triangle wave signals whether or not multiplied by a value of 1. We must conclude that *each* spectral component in the spectrum consists of *a positive and a negative* frequency!

If we compare the levels of the triangle's spectral components after multiplication with the cosine wave with the levels of the original components, we can see that the components have become 6 dB higher than those of the original: the *sum* of the two equal components is two times as high as one component, and a factor 2 amounts to 6 dB.

For another demonstration of the implications of negative frequencies we can investigate the spectrum of the once-occurring damped sine wave, as shown in fig. 10.3,

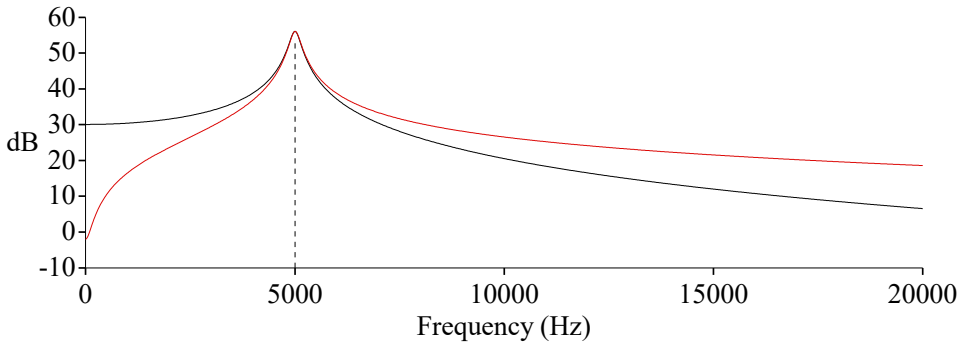


Fig. 11.3. Spectra of damped sinusoids. Black: spectrum of damped sine. Red: spectrum of damped cosine. Both spectra contain sine and cosine components.

in some more detail. When we look at this spectrum on a dB scale (see the black line in fig. 11.3) we can see that it is not symmetrical: the components of the lower side band have greater levels than those of the higher side band. In the figure the spectrum of a damped *cosine* is shown as well (red line). Here the opposite occurs: the lower side band levels are lower than the higher side band levels. From what is explained above we can conclude that these level differences are caused by the 'folding back' of the negative frequency components. In both cases the exponential envelope spectrum itself remains unchanged. Then the difference must be caused by the opposite sign of the negative counterpart of the sine spectrum w.r.t. the cosine spectrum, as shown in fig. 11.2. (In the spectrum the final result of the addition of frequency components is always displayed as its absolute value, or the log of the absolute value, as you will know.)

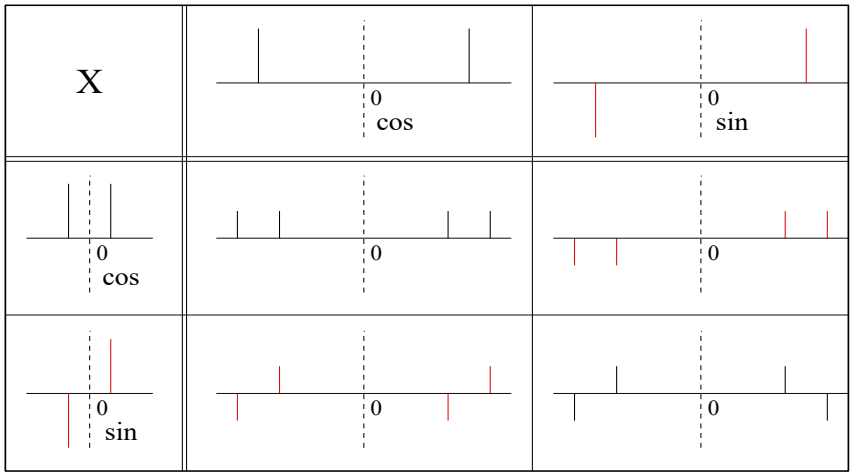


Fig. 11.4. All multiplication combinations of low frequency waves (left column) and higher frequency waves (upper row). Cosine components are black; sine components are red.

Now we must realize that the exponential envelope function is odd, so its spectrum contains cosines *and sines*, and the positive and negative sine components have opposite signs. This complicates the final levels of all coinciding components. Using similar procedures as in the box **MULTIPLICATION OF TWO SINUSOIDAL WAVES** from section 10, the resulting phases of all combinations of sine wave and cosine wave multiplications can be derived. See fig. 11.4 for an overview. Here the cosine components are presented in black and the sine components in red. (As can be seen, the multiplication of two sines results in *cosines*.)

When we look at the *separate* sine and cosine components of the spectra of the damped sine and the damped cosine, we get something like fig. 11.5. (The spectral levels are

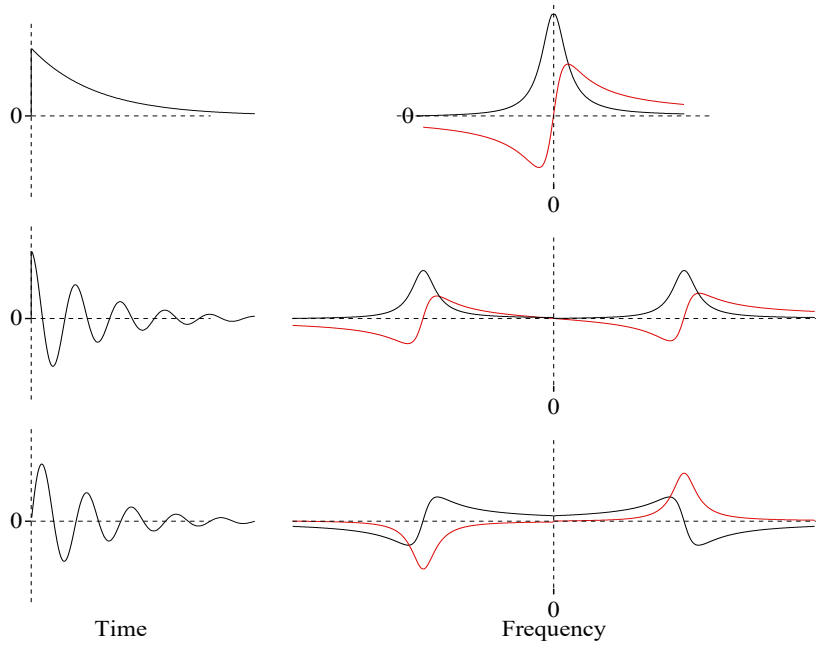


Fig. 11.5. Top: exponential envelope of damped sinusoids and its spectrum. Center: damped cosine and spectrum. Bottom: damped sine and spectrum. Cosine components in black; sine components in red.

drawn at a linear scale to distinguish negative from positive amplitudes.) The higher level of the components in the low frequency area of the damped sine compared to these of the damped cosine is caused by the *cosine* components of the damped sine spectrum.

This example teaches us that the shape of the spectrum of a damped sinusoidal wave depends on the position in time of the multiplying envelope w.r.t. the phase of the continuous sinusoidal wave. In other words: it depends on the phase of the start of the damped sine wave. If the start occurs at a zero crossing of the sine wave, we get a spectrum like the black line in the figure. If the start occurs at a peak (positive or negative) of the sine wave, we get a spectrum like the red line. Other starting positions result in graphs somewhere between the green and red lines. In practical systems the produced damped sinusoids usually start from zero, not from the maximum amplitude,

so the damped sine represents the practical situations better than the damped cosine. The main issue here is to explain that the spectral differences between a damped sine and a damped cosine are caused by the opposite sign of the 'folded back' negative frequency components.

We know that in reality, a negative frequency does not make sense. Ironically, nobody can hear if a sine wave is played backward or not. Is this a paradox then? Not really. Consider two cars moving at different speeds: v_1 and v_2. Speed v_2 is 10 km/h greater than v_1. This difference remains constant. (As a consequence, the distance between the cars increases continuously.) Now both cars limit their speeds at the same rate: they slow down at, say, 10 km/h per second. After some time, the speed of the first car will have become zero. The second car is still slowing down at the same rate. To keep the *difference* of their speeds constant, the first car will have to move *backward*. The same occurred in the beginning of this section when we multiplied sounds with a sine wave. When a frequency component of the sound 'folds back' it is actually running backward in time! The 'folding back' occurs because of the fact that only the positive frequencies are displayed in our spectra.

We can prove this reversing of time with a practical example. We saw in section 6 that the Spectrum object in Praat contains sine and cosine values to include the phase information. We also know that: cos (ωt) = cos ($-\omega t$) and sin (ωt) = -sin ($-\omega t$). Consequently, if we change the sign of all *sine* values of a spectrum, its inverse Fourier transform back into sound will produce the original sound, only *reversed* in time. This is valid for all sounds: even if you have recorded a long story, made a spectrum of the whole sound, changed the sign of all sine values (the second row within the Spectrum object) and inverse transformed it to sound, the whole story would be played *backwards*.

Thus, negative frequencies exist in reality if you see them as a way (maybe the best way) to spectrally express movements backward in time. It seems that there is much confusion about negative frequencies, probably caused by the fact that the word 'frequency' does not allow for negative values in normal language. Many people think then that negative frequencies are caused by mathematical peculiarities and stem from the *complex* representation of spectra (explained in Appendix II) but there is no fundamental connection as may be clear from the description here. It is only a matter of shifting a frequency beyond the zero boundary to cover the opposite direction of movement.

Anyway, the concept of negative frequencies can be seen as a very convenient way to describe many phenomena in the signal analysis field, as we will see.

12. Convolution in the frequency domain

Now we know about negative frequencies, we can look again at the multiplying matter of section 10 from this new angle. The 'mirror' components of the spectra of all figures of this section stem from the negative parts of their original spectra! So, the multiplication of a signal in time with a sinusoidal wave only results in a frequency shift of the *complete spectrum* of the time signal, thus including the negative part. The amount of frequency shift is equal to the frequency of the sine wave. In fig. 12.1 the complete spectra of the triangle wave and its multiplication with the sine, as described in section 10, are displayed. Of course, the sine wave too has its negative counterpart in its spectrum which means that the complete set of harmonics of the triangle wave is *also* shifted in the negative direction. In practical spectra, the negative areas usually are not shown. However, when shifts are involved, i.e. when functions of time are multiplied, it will be obvious that the negative areas must be taken into account.

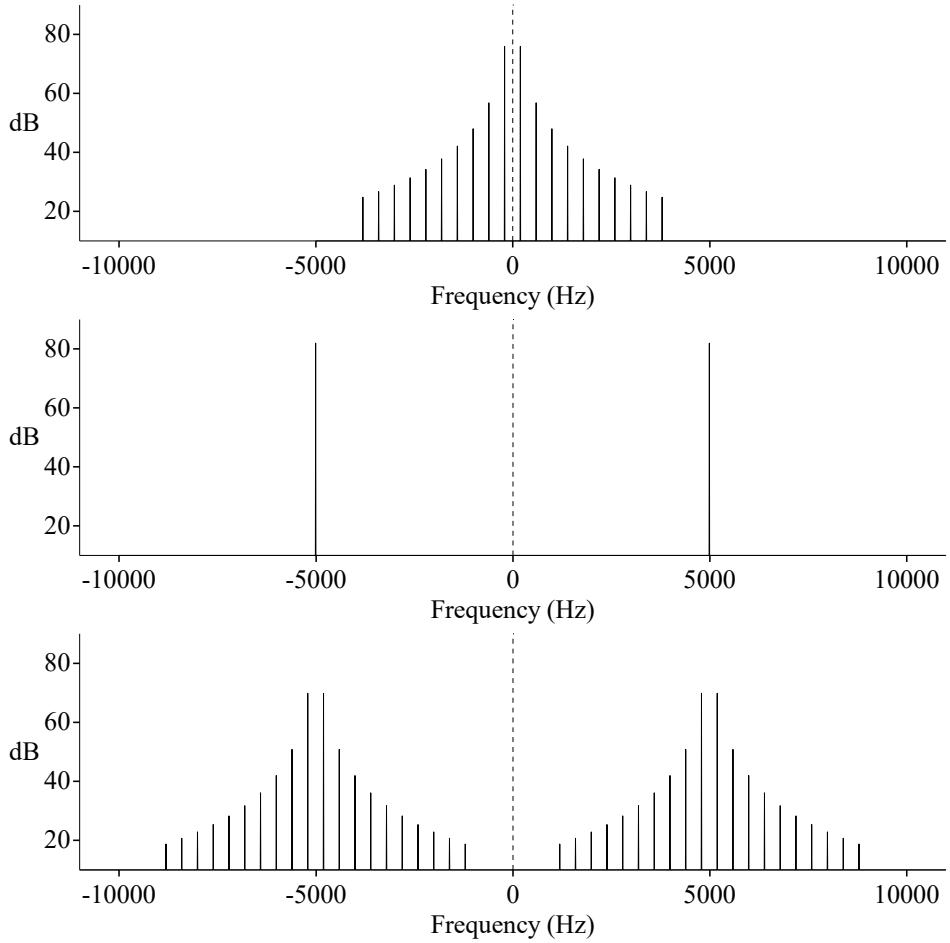


Fig. 12.1. Spectra including their negative parts of triangle wave of section 10 (top), sine wave of 5000 Hz (center) and the multiplied waves (bottom).

In our example of the triangle wave we can see that its spectrum has zero value at zero frequency. So, its mean is zero, as can be easily seen from its waveform. (In section 6 this mean value or *DC value* is explained.) When the triangle wave would have had some mean value, the 0 Hz component had also shifted and the resulting spectrum would have had a component at exactly 5000 Hz as well.

Naturally, apart from a limited number of spectral components like those of our generated triangle wave, this shifting mechanism as described applies also to the infinitely number of components of a continuous spectrum, as we have already seen in section 10 where the output of the resonator has been regarded as the multiplication of a sine wave with a once-occurring exponential function (fig. 10.3).

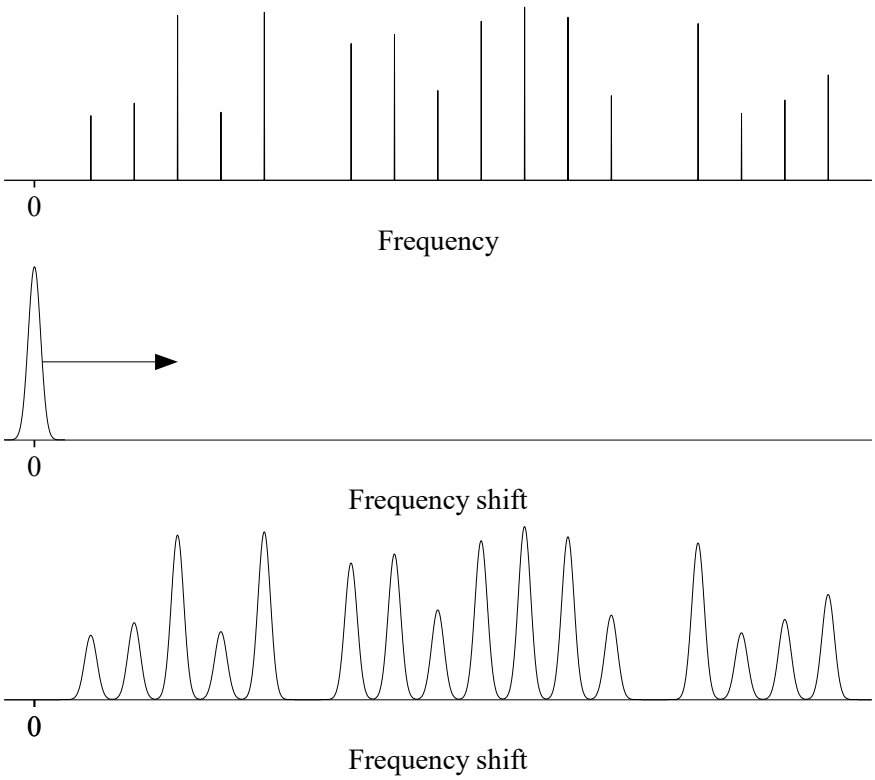


Fig. 12.2. Sum of 16 sine waves of different frequencies (spectrum at top) multiplied by a time function of which its spectrum is displayed in the center. Bottom: the (positive part of the) spectral result.

The next step we take is using a *set* of sine waves instead of a single one. It will be obvious that the spectrum of the multiplying function will be shifted around each separate sine wave frequency. Fig. 12.2 gives an example, using a multiplying function having a continuous spectrum. It may be clear that the complete continuous spectrum is copied at each spectral line while their heights are proportional to, or *weighted by*, the amplitude levels of the different spectral lines.

The arrow in the figure suggests a different manner to construct the result. The spectrum of the multiplying function is shifted from the negative end to the positive end of the whole frequency range of the line spectrum, using infinitesimal small steps. In each step both spectra are multiplied and the value is put into the resulting graph. (Actually, the multiplying function spectrum has to be *reversed* before the shifts: moving this spectrum from low to high frequencies cause the line spectrum components to shift into the multiplying function from its right side! Here, however, it makes no difference as the multiplying function is symmetrical in the frequency domain.)

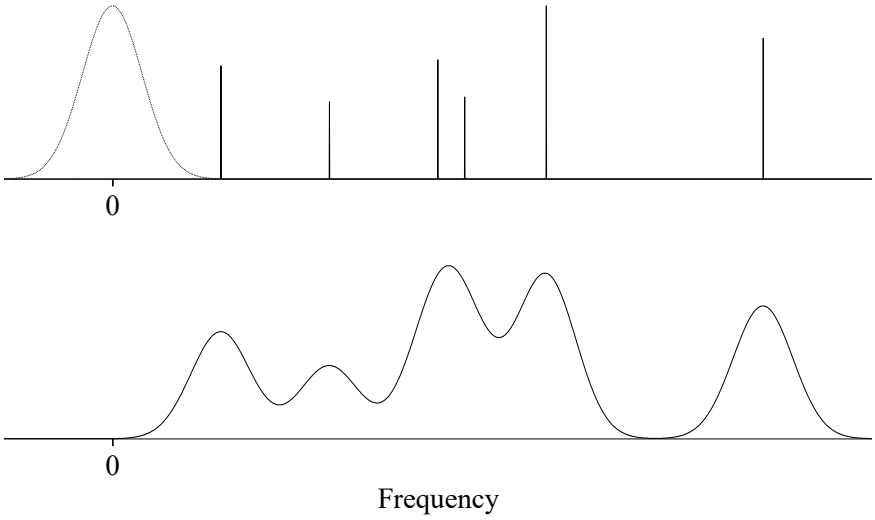


Fig. 12.3. When the spectral line distance is lower than the spectral range of the multiplying function there are overlapping spectral areas.

When the frequency range of the spectrum of the multiplying function exceeds the spectral line distances, as we can see in fig. 12.3, there are some overlapping areas where components 'belong' to different sine wave frequencies. These coinciding effects can be seen in the lower part of the figure, as the non-zero areas between the sine frequencies and as the position of the peak at the center of the frequency range: it is *higher* than the highest of the two sine components close together, while its maximum occurs somewhere between the two sine frequencies. If components coincide, their sum depends on the phase relation, as we have seen in the preceding section.

The effect in the frequency domain, when we multiply functions of time, is called **convolution**. The concept of convolution is *one of the most important mechanisms in the signal analysis area.* Understanding this mechanism opens the door to the explanation of most of the phenomena you will encounter when dealing with spectral analysis.

To get some feeling about the convolution mechanism in the frequency domain we may consider the swept and mirrored spectrum like that of the dotted line in fig. 12.3 as a tunable band filter function in the frequency domain which, in each tuning (shift) position, selects a small part of a whole spectrum, and suppresses all the rest (identical to shifting the tuning knob of an old fashioned AM radio receiver which selects only the small frequency part of one radio station from the total range). Although a filter is not the same as a signal (a filter does not produce signals) its function of frequency could be regarded as if it stems from a signal. The only difference, then, is the different scaling of signal amplitude and filter gain.

THE CONVOLUTION INTEGRAL

Each point of the convolution function is found by multiplying the one function with a frequency-shifted and *reversed* version of the other, and calculating (integrating) the encompassed area. The frequency shift is performed in infinitesimal steps over the whole frequency range.

The position of the filter must be independent of other variables, like ω or t, so we need a separate variable for the frequency *shift* (a kind of *shift domain*). This is accomplished by the formula:

$$F(\omega) * H(\omega) = \int_{-\infty}^{\infty} F(\gamma) \cdot H(\omega - \gamma) d\gamma$$

The asterisk denotes the convolution function and γ represents the frequency shift. The function F can be seen as the 'radio band spectrum' and the function H as the band filter as mentioned in the text. It is reversed in the shift domain by the minus sign.

Naturally, multiplication of spectra implies *vector* manipulation: in each multiplication the phase has to be taken into account.

Each point of this resulting graph then represents the *multiplication* of the whole line spectrum of the function and the filter function *at that particular position* of its center frequency. More accurately: *each point* of the resulting graph represents the sum of all frequency components that are encompassed (multiplied) by the filtered part. Of course, as we have seen in section 8 on filtering, we must apply *vector* multiplications of the spectra because of the phase properties. (In section 5 the sine wave is explained in terms of a rotating vector.) So, all multiplications of sinusoidal components must be done by applying the method as mentioned in the box **MULTIPLICATION OF TWO SINUSOIDAL WAVES**. (There is a mathematically more suitable method, however, which is described in appendix II.) Nevertheless, the graphs in this section deal only with amplitudes.

For now, it is important to conclude that *multiplication in the time domain is equivalent to convolution in the frequency domain*:

$$x(t) \cdot y(t) \Leftrightarrow X(\omega) * Y(\omega) \qquad (10.3)$$

The double arrow denotes the transform from time domain to frequency domain and vice versa. The asterisk represents the convolution mechanism. It is customary to use lower case symbols for functions of time and capitals for their frequency domain versions. The box called **THE CONVOLUTION INTEGRAL** describes the mathematical

procedure of convolution in the frequency domain. The procedure is exactly identical to the shifting method of the (reversed) filter graph as mentioned above.

13. Windows

In section 7 we concluded that the output of a once-excited resonator can be seen as a multiplication of a continuous sine wave with an exponential function:

$$x(t) = A_0 \cdot e^{-\alpha t} \sin(\omega t) \tag{13.1}$$

From the preceding sections 10 and 12 we learned what the effect is in the frequency domain: the spectrum of the single exponential and the spectrum of the sine wave are convolved. The (continuous) spectrum of the exponential and its mirror version are shifted to a position around the sine frequency, just as displayed in fig. 10.3.

We have to know more about the relation of the shape of a finite time function and its spectrum because in practice we do not have signals of infinitely length. Moreover, in cases of relatively long signals we want to be able to analyze smaller parts of it in succession, because often the waveform will not be steady all the time for long durations. This will be obvious for speech signals. It is very well possible to make a Fourier transform of a complete talk of half an hour or so but its spectrum will only present frequency amplitudes that are averages over the whole duration and we will have no spectral clue about the local time parts of the sound. Even if we take the phase information into account as well, it will give us no clue about the *local* spectral components because, as you will remember, the phase information only contains the phase of the *origin* of each component, whereas the component amplitude and frequency remain constant everywhere in time.

For measuring "overall" properties of voices or musical instruments a display of the spectrum of a long-time recording, which is called a "long time average spectrum" (LTAS, see part B) can be of value, but often we need spectral data that are more detailed in time.

So, in general, we need some selection of a part of the signal. As mentioned in section 6, when selecting parts of longer sounds, we have to make sure that we take exactly one F_0 period or a multiple (if the sound is periodic) and choose the DFT then. Often, however, the sound is not strictly periodic (varying F_0, noisy parts, sudden changes, etc.). Particularly in cases of speech sounds the 'real periods' may be hidden, and difficult to isolate. Generally, the only intention is to get spectral information over a specifically selected small part. As said before, if we simply take a DFT from an arbitrary part T of the sound we get frequency components $1/T$ hertz apart: the spectrum is *sampled* at these frequency intervals because the DFT 'assumes' that the selected part is *repeated*. But which spectrum is sampled? There must be an 'underlying' continuous spectrum if we regard our signal selection as once-occurring.

If we use our 'time-insertion trick' from section 9 we can approximate this underlying spectrum very well. After we cut the part of the sound from which we want to get the spectrum we add a fair amount of zero sound so that its DFT produces a spectrum with

closely packed spectral components. In fig. 13.1 we see the continuous spectrum approximated in this way of 50 ms of a 490 Hz sine wave. Quite disappointing, as you will agree. We would like to see one peak at 490 Hz instead of all these **side lobes**.

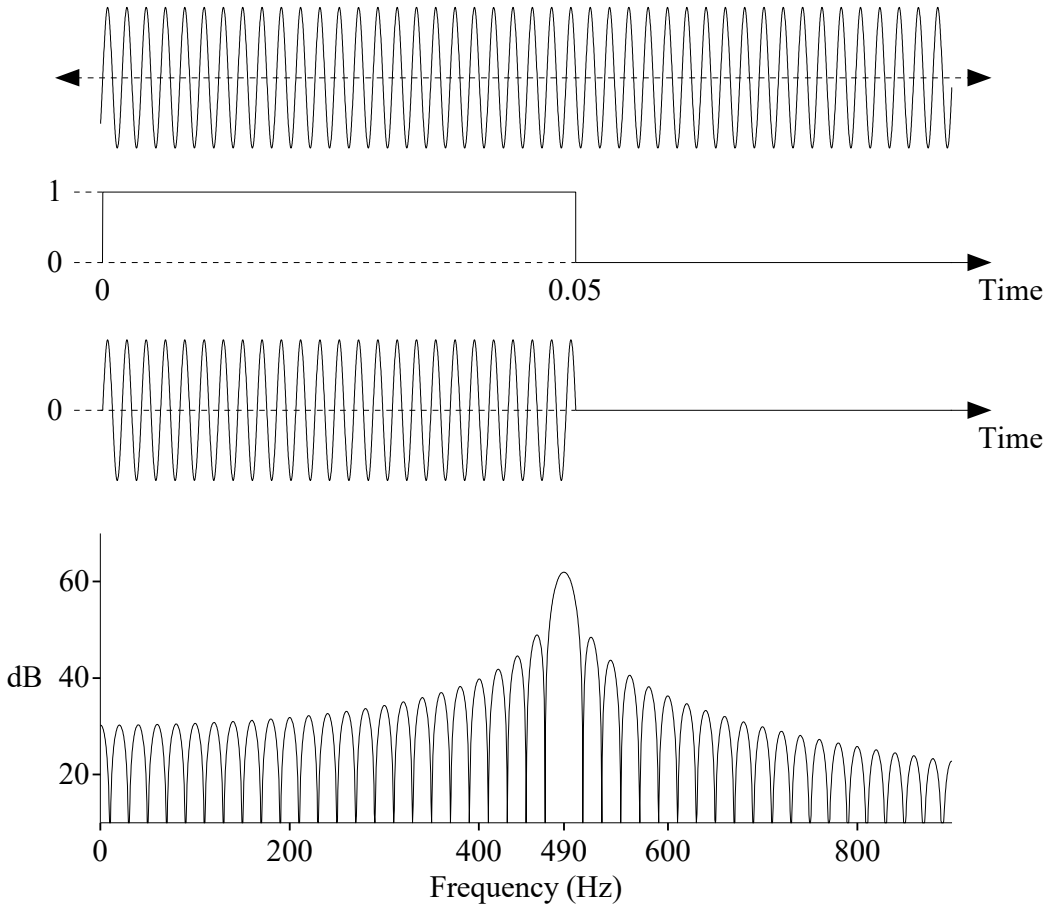


Fig. 13.1. Approximation of continuous spectrum of 50 ms of a 490 Hz sine wave.

In fact, we multiplied a periodic sound with a **rectangular** window with a value 1 within the desired segment and a value 0 outside it. So, what we want to find out is the spectral effect of this manipulation. But we know that multiplication of time functions means *convolution of their spectra* so that this spectrum must be the convolution of the underlying spectrum of the 490 Hz sine wave and the (continuous) spectrum of this rectangular window. Because the 'spectrum' of the sine wave consists of only one spectral line, the spectrum in fig. 13.1 must be a good approximation of the spectrum of the rectangular window, shifted to the right by 490 Hz.

As mentioned in section 9 the Fourier integral can be used to calculate the exact continuous spectra of time functions and appendix II.3 contains the way to do this for the rectangular window, using complex numbers.

In fig. 13.2 this rectangular window spectrum is shown, together with its absolute value (generally, spectrum displays are independent of the sign) and the commonly used logarithmic scaled version. The 'negative frequencies' are shown as well. Theoretically, the positive and 'negative' frequency ranges both extend to infinity and are of course symmetrical. Now its shape closely resembles our continuous spectrum of fig. 13.1. The phenomenon of the left and right sides of fig. 13.1 not being symmetrical is caused by the 'folding back' of its negative frequency components, as explained in section 11. In fig. 13.2 there is no folding back because of the inclusion of the negative range in the picture.

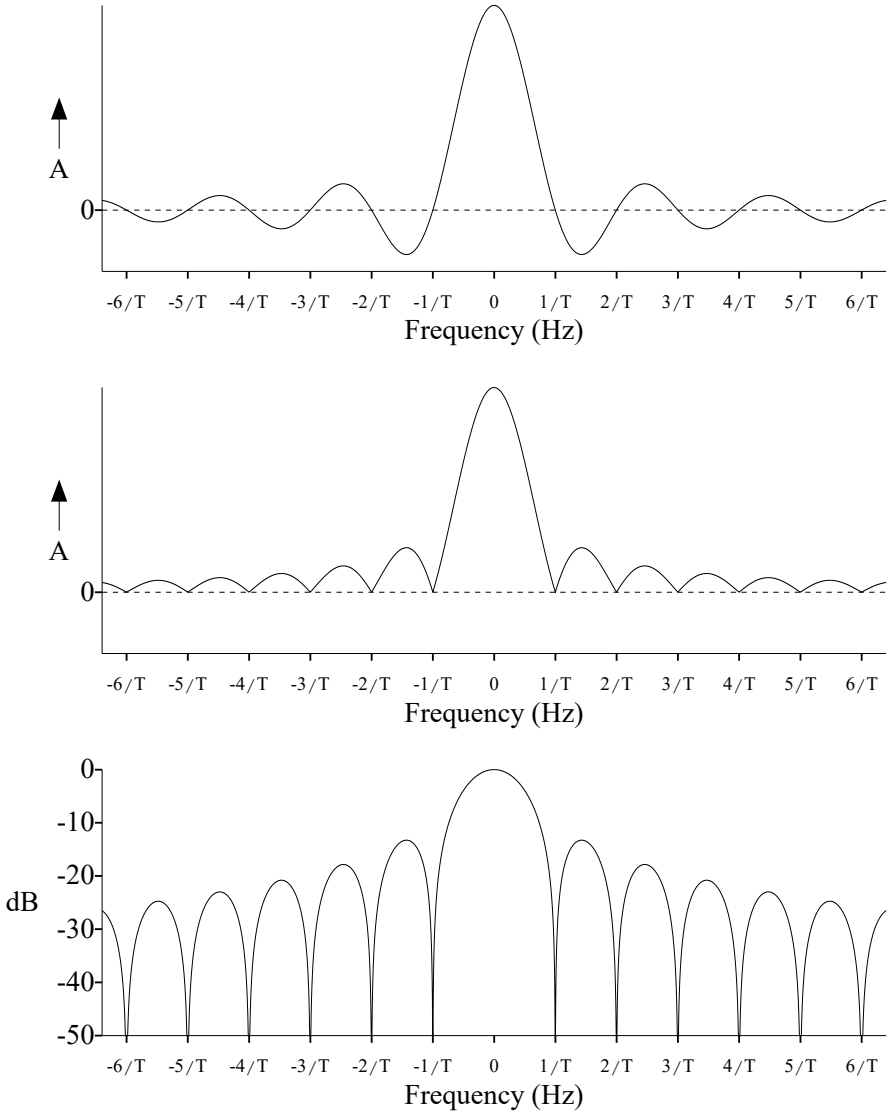


Fig. 13.2. Spectrum of rectangular time window with length T s. Top: 'linear' spectrum. Center: amplitude. Bottom: log scaled, maximum normalized to 0 dB.

The formula of this spectral function:

$$G(f) = T\frac{\sin(\pi fT)}{\pi fT} \tag{13.2}$$

shows that it is zero when f is a multiple of $1/T$ (because the sine of an angle of π radians or a multiple is zero) so that the 'lobe width' is $1/T$ Hz, where T is the window length.[1] At the center of the *main lobe* it has the value T which is only an amplitude scaling factor. This is the well-known **sinc function** which is short for **sine cardinal**. One should be aware on its influence on spectra: especially when there are more than one sine components in the original spectrum (and that's generally the case, of course) each sine component will have its 'own' side lobes and folding back components and they

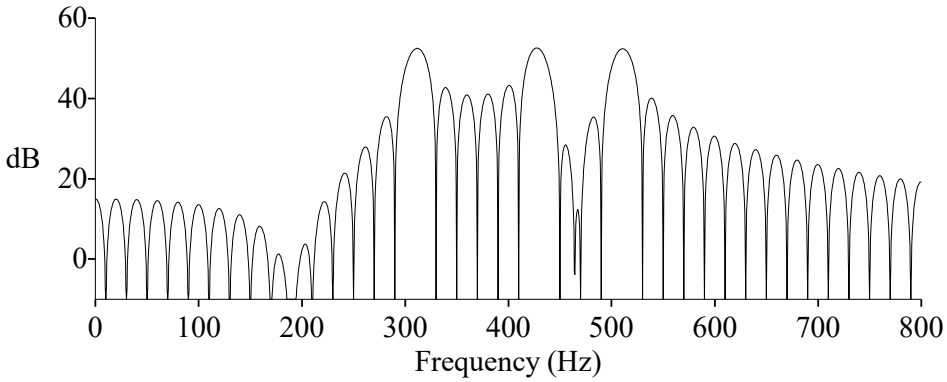


Fig. 13.3. Continuous spectrum example of three added sine waves, rectangular windowed.

can interact heavily in unexpected ways. See fig. 13.3 where the continuous spectrum of 50 ms from a combination of three sine waves is shown. Here also the 'folding back' of the negative frequency components cause interaction of components.

But what if an integer number n of periods of a sine wave fits exactly within the window length T so that the period of the sine wave is T/n? You will understand that it makes no difference for the shape of the *continuous* spectrum: the peak of the *main lobe* is positioned exactly at the sine wave frequency n/T and the side lobe width remains exactly $1/T$ Hz. The cause of the *DFT* of an integer number of sine periods producing only one clean spectral line at the sine frequency is that the DFT 'samples' the continuous spectrum at $1/T$ Hz distances, and *all multiples of 1/T or n/T Hz fall exactly on the zero points of the sinc function* except at the main lobe center where the value is positive!

[1] Obviously, the spectrum of the 'very short pulse' which was used in section 7 to activate the resonator is defined by this function. To approximate a flat spectrum in the range of interest, the first zero position at $1/T$ hertz should be sufficiently high. If we allow for a certain roll-off (say, 3 dB at 10000 Hz), the maximum length of the pulse can be calculated by: $fT = 0.443$. So, then T should be no longer than 44.3 μs (microsecond).

You will remember that the exponential decaying sine wave can be expressed as a multiplication of a continuous sine wave with the exponential function (formula 13.1). From its continuous spectrum (as in fig. 10.3) we can conclude that it shows no side lobes and has only one peak. Therefore, using an 'exponential window' instead of a rectangular window seems to be advantageous.

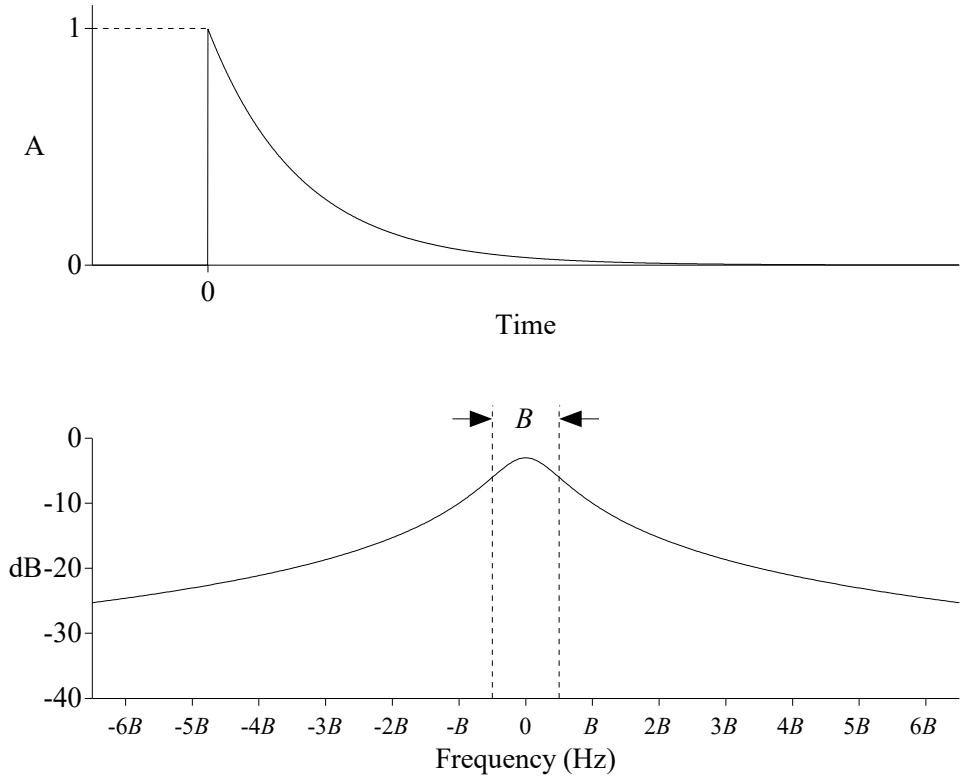


Fig. 13.4. Continuous spectrum of exponential window; peak normalized to 0 dB.

Fig. 13.4 shows the continuous spectrum of this exponential window (as already indicated in section 11). We can see that the spectral width is quite substantial: the attenuation of frequencies beyond the 3-dB bandwidth is fairly gradual so that the *selectivity* is poor. In practice, therefore, this window can only be used to investigate the start of a signal spectrally (the *transient*). In addition, theoretically, the time function goes on to infinity and in practice it has to be truncated. The consequence of truncation, however, is an extra multiplication with a rectangular window which again introduces side lobes (although with lower amplitudes). We have seen this in fig. 9.2 of section 9: the 'ripple' in the spectrum is manifested by these side lobes! The lobes are 25 Hz apart, caused by the truncation at 40 ms.

A much better window is the **Gauss** window, named after Karl Friedrich Gauss, a German mathematician (1777-1855). Although this window presents us with the same problem of 'never becoming zero', it has a better spectral selectivity and no side lobes, see fig. 13.5. In statistics this bell-shaped curve is known as the **normal distribution**.

If we scale its peak to 1, the Gaussian window formula is:

$$f(t) = e^{-\alpha^2 t^2} \tag{13.3}$$

where the peak is centered at $t = 0$. Its spectrum from appendix II.3 is defined by the formula:

$$F(\omega) = \frac{\sqrt{\pi}}{\alpha} e^{-\omega^2/(2\alpha)^2} \tag{13.4}$$

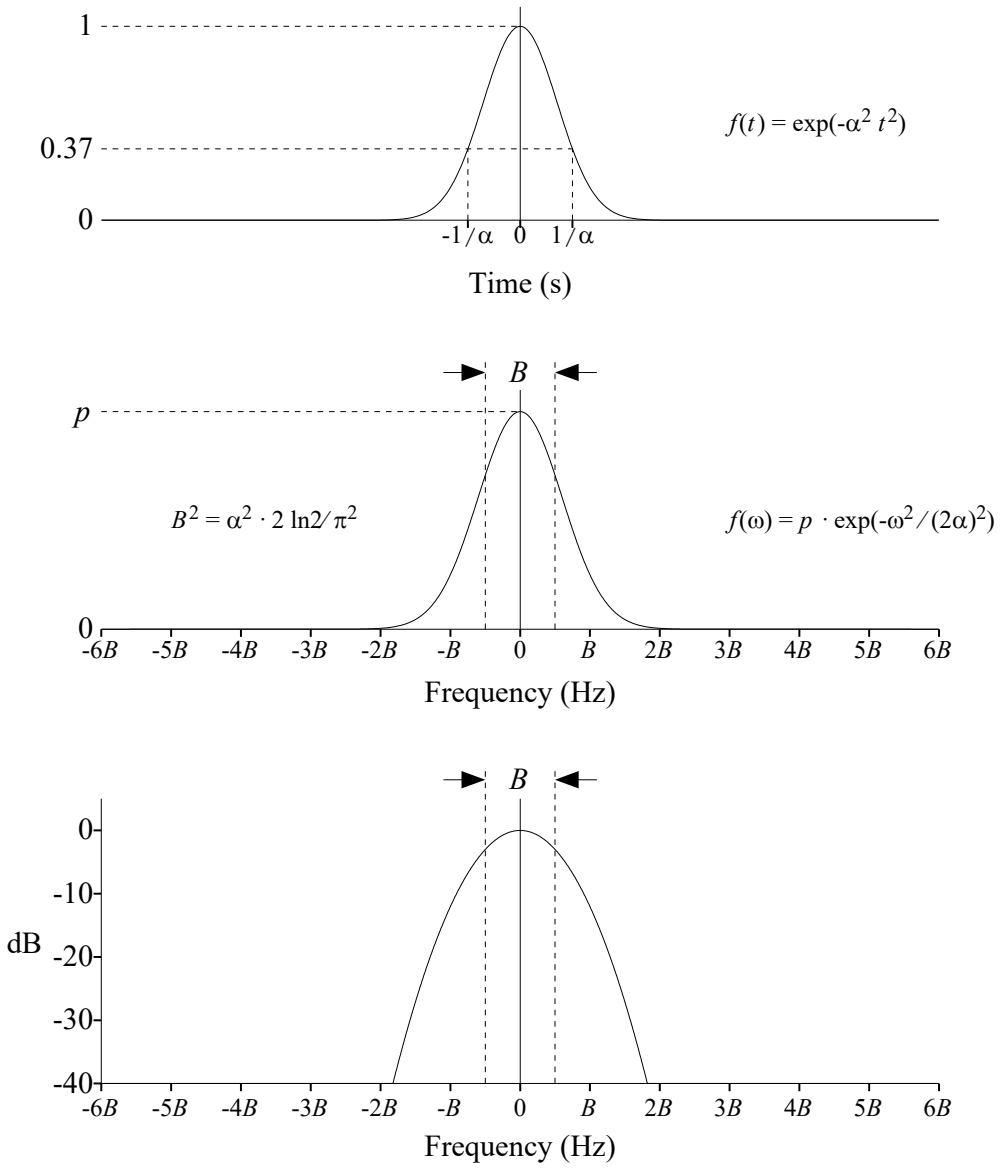


Fig. 13.5. Top: Gaussian window. The linear scaled spectrum (mid) is also a gauss function. Bottom: spectrum on log scale (peak normalized to 0 dB). Scale factor $p = \sqrt{\pi}/\alpha$.

As you can see, this spectrum function is also a Gauss function and therefore completely 'side lobe free'. From the formula you can also conclude that 'shortening' the time function by increasing α means 'broadening' the spectrum and vice versa, just like we saw at the exponential decaying function. (The factor $\sqrt{\pi/\alpha}$ is only a scaling factor and has no influence on the function's shape.)

Symmetry

Recalling what is explained about symmetry in section 6 you will understand that all windows described so far, except the exponential window, are *even functions*. The result in the frequency domain is that only cosines exist. In appendix II.3 it is shown that in this case the time function and its frequency function are interchangeable. That means, if we regard the spectral function of, for example, the sinc function (top of fig. 13.2), as a *sound* instead of a spectrum, its Fourier transform will be perfectly *rectangular*. In other words, multiplication of a sound with this sinc function acts like an *ideal low-pass filter*. (Mathematically, ideal filters pass all frequencies within the *pass band* un-attenuated and completely block all frequencies outside that range.) In fact, the sinc function can be seen as the impulse response of the ideal low-pass filter. Of course, this sinc function is not a possible impulse response of physical filters in practice. This trick, however, is often used in *digital filtering* which will be described later.

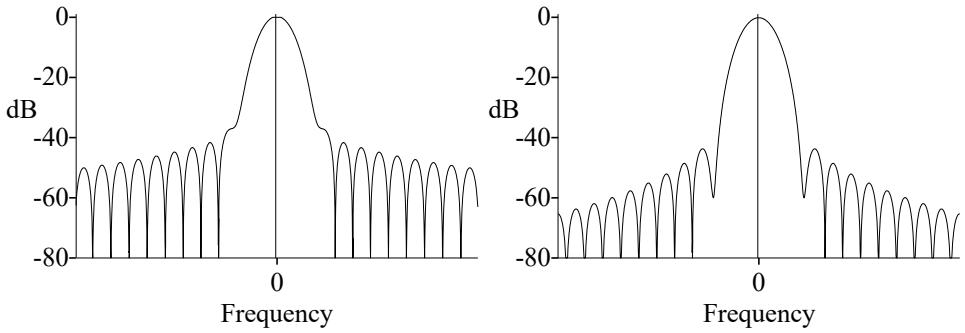


Fig. 13.6. Spectrum of Gauss window, truncated at 5% of max. value (left) and tampered after truncation (right).

The Gauss window 'never becomes zero' so, in practice, it has to be truncated, which will cause side lobes. To limit the heights of the side lobes, sometimes the Gauss window is *tampered* (modified) as follows: truncate the function at the positions where the value is, say, 5% of the peak value, subtract this level from the function so that the start and end are zero, which eliminates the steps of the function, and finally multiply the function with 100/95 so that its peak is 1 again. This eliminates the side lobes not completely (as the tampered function is no real Gauss function any more) but it suppresses them noticeably. See fig. 13.6 for the effect on the side lobes.

In attempting to overcome the problem of the side lobes many alternative window functions have been developed, all of them being ways to compromise for frequency

selectivity, time length and side lobe amplitudes. Fig. 13.7 shows Fourier transforms of some popular window functions. The Blackman and Kaiser windows have no fixed properties: they can be modified by using parameters to limit the main lobe width at the cost of suppression of side lobe amplitude, or vice versa. The relative selectivity of each window can be read from the position of the markers $2/T$ or $-2/T$ where T is the length of the window time function.

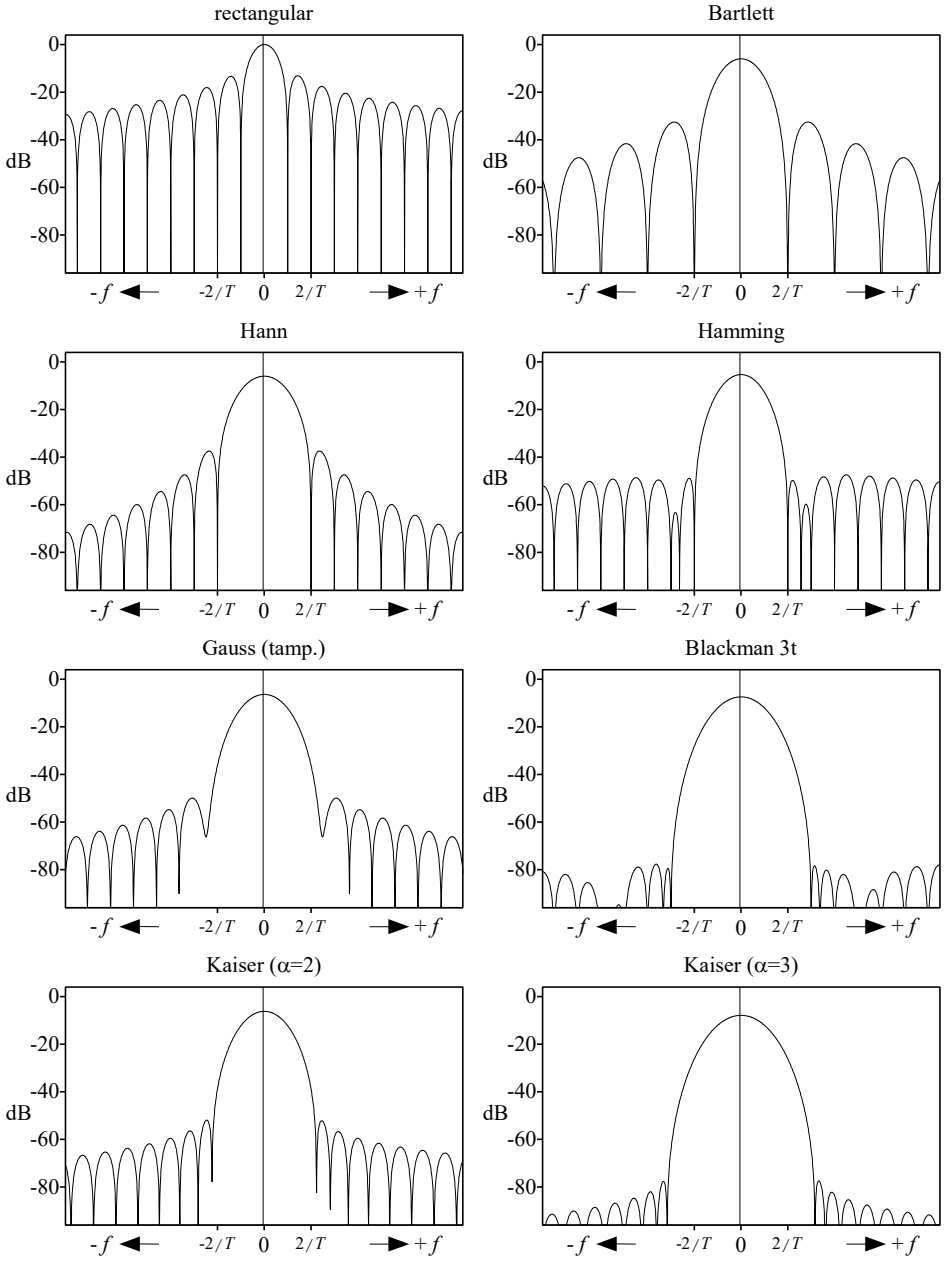


Fig. 13.7. Continuous spectra of some window types. The time lengths of all windows are equal to T. All dB values refer to the peak level of the rectangular window.

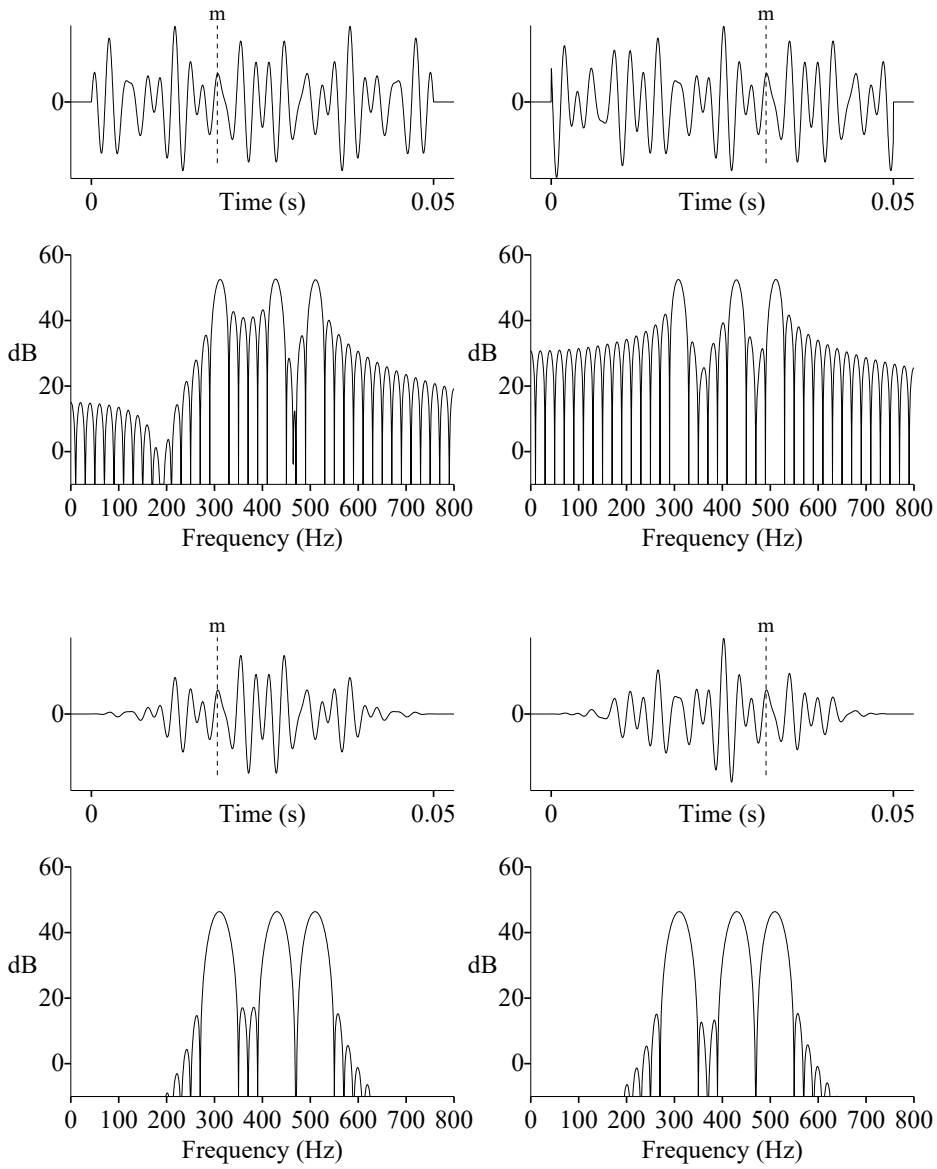


Fig. 13.8. Different selections of 50 ms of a signal containing 3 added sine waves, and their effects in the spectrum. Right half of the picture: selection window shifted 18 ms to the left. Top half of the picture: selections rectangular 'windowed' and their spectra. Bottom half: selections windowed with Hann window and spectra. A random position in the signal is marked as **m** to show the shift.

The importance of suppressing the side lobes can be demonstrated by an example. We will follow the same procedure in fig. 13.8 as in the example of the three sine waves signal of fig. 13.3, the only difference being the *position* of the rectangular window: on the right-hand side it is shifted to the left by 13 ms. The spectrum should not change because only the phases of the components have been altered by the time shift. Compared to fig. 13.3 we see that the components remain in position but the shape of the spectrum has changed a lot. The 'folded back' components interact differently as their phase is altered by the shifted window position! Also, in fig. 13.8 (in the lower

half) the signal is windowed with a Hann window instead. You will see that the spectra in both window positions are practically the same. The side lobe suppression by proper windowing greatly limits the interaction of the folded back components and at the same time it makes the spectral components much more visible, at the cost of a slightly wider main lobe.

The number of window types that has been developed is overwhelming: only a small number is shown in fig. 13.7. Which is the best window to use? The answer (as always) depends on the purpose of your analysis. First of all, if the frequency components in the signal remain steady for a relatively long duration, you can apply a long-time window. The choice of the window type is then only a matter of side lobe *roll-off* because the main lobe is narrow enough. Any window will do, except, of course, the rectangular window and the Bartlett (triangular) window. (These two you should better avoid in all cases!)

When the frequency components in the signal are fast varying (as in the case of the signal of a sentence of speech), the window length has to be a compromise. On the one hand you will prefer a short window moving through the signal, thus producing a spectral contour detailed in time. On the other hand, a short window will have a broad main lobe in its spectrum so that the frequency resolution is limited. Therefore, the window should offer both maximum selectivity and short length. In addition, the side lobes should be suppressed sufficiently. The Blackman, Gauss or Kaiser windows offer reasonable solutions. When the window is moved through a longer signal it is not necessary to truncate those windows that 'never become zero' (apart from the window positions at the start and end of the long signal). Then the Gauss window is superior to all others, and a good compromise can be made by choosing between frequency resolution (selectivity) and time resolution. In part B the sections about speech measurement explain some further complications in the spectral analysis of speech signals.

In Praat, several windows are built-in to facilitate *windowing* of an extracted part of a signal. The Gauss (tamp) and Kaiser ($\alpha = 2$) windows there are named 'Gauss1' and 'Kaiser1' respectively.

An extensive overview of many window functions and their properties is produced by Heinzel, Rüdiger and Schilling [5].

14. Convolution in the time domain

We began all this about convolution and windows by multiplying in the time domain. Let's now take the frequency domain as a starting point. We learned that the spectrum of the resonator output is formed by multiplication of the spectrum of the input (the train of short pulses, for example) and the filter function of the resonator. In the time domain representation of the output (as in fig. 8.6) you can see that the resonator is activated repeatedly at each position of a pulse. The damped sine wave of the resonator in a way has been copied to all periods of the pulse signal. All periods consist of the same damped sine of our resonator. Not surprising: the resonator is excited at every pulse. But this is exactly the convolution mechanism as described in section 12: shifting one function (in infinitesimal small steps) and multiplying it with the other in all shifting positions. DEMO14.1 shows what happens, applying small sequential steps of shifts in time. Here we can see also that, prior to the shifting, the damped sine necessarily has to be reversed in time (see also fig. 14.1). In each shift position the multiplication of the functions only produces values at the pulse positions.

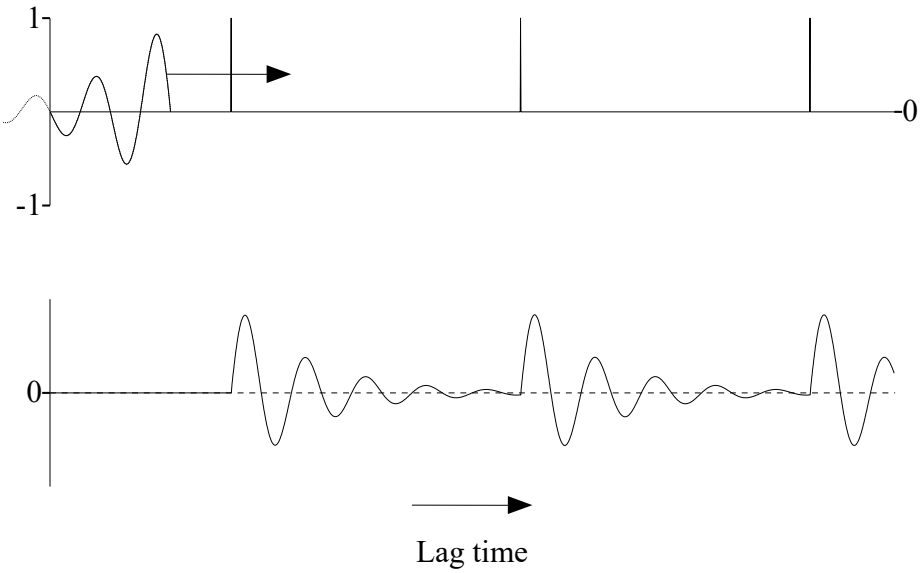


Fig. 14.1. Convolution of short pulses with a sampled sine wave.

When the pulses are infinitesimally short, indicated in the top of fig. 14.1, there is only one value at each pulse; between pulses the values are zero. (This is a relatively simple case. The shift causes the resonator function being 'scanned' at each pulse.)

If the fundamental frequency of the pulses becomes higher, there is not enough time for the resonator to come to rest before a new excitation occurs. In that case the damped sine of the resonator still has some amplitude and this will influence the waveform of the next period. When the **steady state** has been reached, all next periods become the same (because the state of the resonator at the start of the following periods has reached

an equilibrium). The influence of a period on the next one is constant from now on (this steady state is what you saw in fig. 8.7 in section 8 about filtering).

Obviously, if the fundamental frequency of the pulses is so low that the amplitude of the damped sine is negligibly low before a new excitation occurs, the result is equal to repetitions of impulse responses of the resonator, and the spectrum is a sampled version of the continuous spectrum of the impulse response. In other words, a sampled version of its filter function.

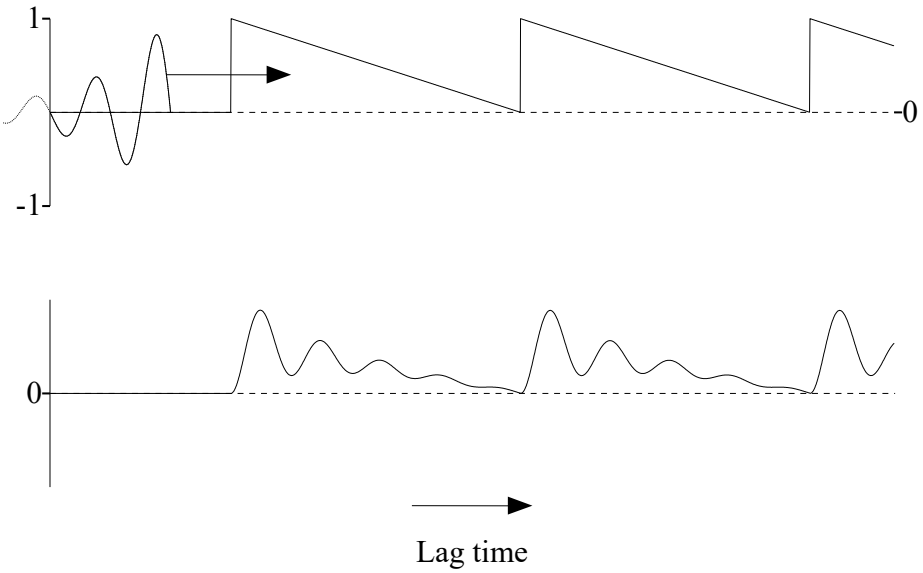


Fig. 14.2. Convolution of sawtooth wave with damped sine wave.

Instead of very short pulses, if we apply a *waveform* of some duration (like a sawtooth wave, for example) we have a more complicated situation as fig. 14.2 shows.

Obviously, the damped sine is not simply copied into each period because of the influence of the contents of the sawtooth wave on the result. To investigate in some detail what happens you may follow some sequential steps in the convolution procedure by running DEMO14.2 which works exactly like DEMO14.1, but now with a sawtooth wave instead of short pulses.[1]

1 In the demos and figures the *mean* of the multiplied functions is shown. The math, however, uses the *integral* which calculates the total *area* of the multiplied functions. Strictly spoken, these mean values are only valid when the signal duration equals 1s. So, to be correct, all resulting values should be multiplied by the signal duration, which is only a matter of scaling.

CONVOLUTION IN THE TIME DOMAIN

Each point of the convolution function is found by multiplying the one function with a time-shifted and *reversed* version of the other, and calculating (integrating) the encompassed area, just like convolution in the frequency domain explained before. The time shifts are performed in infinitesimal steps over the whole time range.

Because the position of the shifted function must be independent of other variables, like *f* and *t*, we need a separate variable for the time *shift* (here called the *lag domain*). This is accomplished by the formula:

$$f(t) * h(t) = \int_{-\infty}^{\infty} f(\tau) \cdot h(t-\tau) d\tau$$

The asterisk denotes the convolution function and τ represents the time shift. The function *f* can be seen as the signal input to a filter and the function *h* as the impulse response of the filter. It is reversed in the lag domain by the minus sign. Seeing the convolution strictly as a function in the *lag* domain we can interchange the abscissa variables:

$$y(\tau) = \int_{-\infty}^{\infty} f(t) \cdot h(\tau - t) dt$$

which is completely equivalent. The upper formula, however, is used most often.

Concluding, we can state that the effect of multiplying two functions in the frequency domain is that their respective time functions become convolved, simply according to the convolution mechanism, but this time in the time domain.

The box called **CONVOLUTION IN THE TIME DOMAIN** describes the mathematical procedure of the convolution mechanism, which corresponds exactly to the frequency domain convolution found in section 10. Thus, multiplication in the frequency domain means convolution in the time domain:

$$F(\omega) \cdot H(\omega) \Leftrightarrow f(t) * h(t) \qquad (14.1)$$

Compared to formula 10.3 you will see the symmetry of the convolution:

- multiplication in the time domain means convolution in the frequency domain;
- multiplication in the frequency domain means convolution in the time domain.

Fig. 14.3 depicts these symmetries in schematic form. The convolution process requires much computing power as in each shifting step the signals have to be multiplied and integrated. Usually therefore the signals are Fourier transformed (with the Fast Fourier Transform), then multiplied and finally inverse transformed back into sound. Obviously, the result is exactly the same. (Praat also uses this trick.)

A practical example of convolution is presented by DEMO 14.3.[1] You can hear the influence of the acoustics of a church space on the sound produced. (The use of

[1] The script uses a studio recording and an impulse response of the Open Air Library of the University of York, see refs. [8] and [9], respectively.

headphones or earphones will demonstrate the effect optimally.) In fact, we can consider the acoustic properties of the room where the sound is produced as a filter which alters the signal of the sound source. The resulting signal is a convolution of the sound source signal with the impulse response of the room. You will hear a striking difference between a recorded sound in an anechoic room (which is a 'dry' recording, almost without any reverberation) and the same sound convolved with the impulse response of the church space. It sounds exactly as if the speaker were present there. This convolution with the impulse response of a room makes it possible to modify any 'dry' recorded sound as if it was played in the room. In principle, the impulse response can be acquired by producing a very short pulse sound in the room and record the result at some listening distance from the sound source. In practice however, the energy per frequency is too low compared with that of the background noise. There are better ways to obtain the impulse response, like using electronically generated noise (the next section deals with the properties of noise) or swept frequency sinusoidal generation. How to extract the impulse response from these signals falls beyond the scope of this book. A practical method with a swept frequency sound source is described by Farina [3].

Reversely, the quality of a sound recording can be poor due to reverberation and frequency-dependent absorption by the interior of an inappropriate recording room. This unwanted filtering can be corrected to some extent by **deconvolution**. Now the spectrum of the signal is divided by the spectrum of the room (i.e. the spectrum of the impulse response). Finally, this resulting spectrum is reverse Fourier transformed into sound. In practice, the room spectrum can have values that are almost zero if sounds from different directions almost cancelled each other. So, the division by this spectrum may cause very high peaks and the reverse Fourier transform may produce dominating sine sounds. Many attempts to solve this problem are still not working very satisfying.

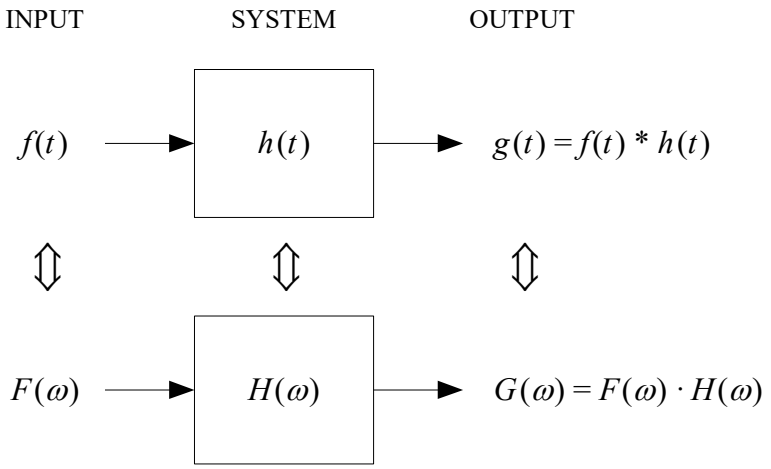


Fig. 14.3. Relation between input, output and system, and their Fourier transforms.

When the impulse response is not known (of speech sounds for example), the deconvolution process is even more difficult. When one tries to distinguish between the *source* properties and *filter* properties from the resulting sound waveform (which is only available), the problems sometimes cannot be solved whatsoever, as we will see.

15. Noise

As you may know, an electric current can flow through a conductive material, let's say a wire, because of the presence of free electrons in the material. A particular free electron, having the smallest possible negative electric charge, will not remain free, but when it has the chance will take the available space within a nearby atom (which will have had a positive electric charge (an ion)). When the electron is added to the atom, the charges of the atom and the electron cancel each other out, and the atom's charge becomes neutral. In the meantime, another electron escapes from its atom, anywhere within the wire and recombines later with an atom with has a space available. This process has been going on for trillions of electrons everywhere in the material so that the total charge will be neutral. Stated differently: there is no voltage between the ends of the wire. However, if we 'zoom in' on the electrical voltage (by using an electronic amplifier) we see small random fluctuations of the voltage around the zero value (see fig. 15.1) caused by fluctuations of the number of free electrons.

So, when a sound has been converted into its electrical representation, it will not be completely 'clean' but contain a (usually small) amount of noise as well. Therefore, we will look at the properties of noise.

Because of the fact that the free electron's behavior is caused by the temperature of the material, the term **thermal noise** is sometimes used. Obviously, the resulting

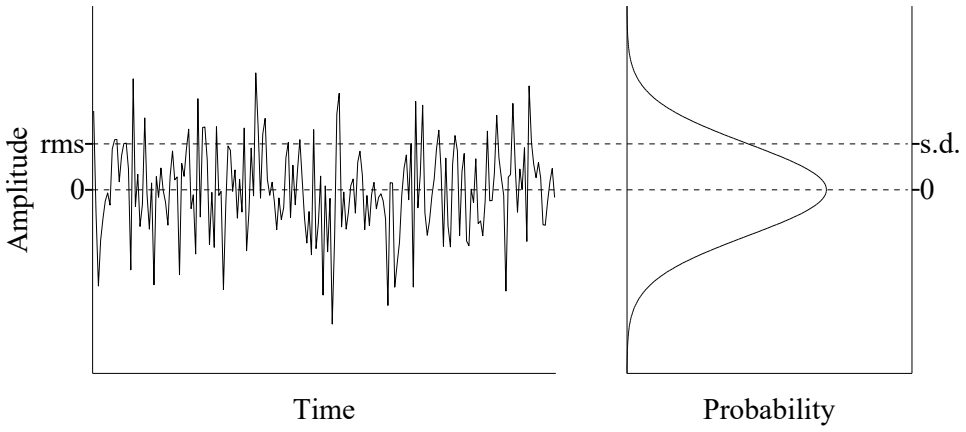


Fig. 15.1. Part of white noise and its Gaussian amplitude probability function.

fluctuations will be completely unpredictable. One thing we know is that the mean amplitude is zero. Another thing we know is that large deviations from zero (positive or negative) will occur less frequently than small deviations because for large deviations there must be a great number of free electrons and a great number of positive ions (charged atoms) at opposite positions in the length direction in the material. The chance that this may occur will decrease with increasing deviation values. When we

construct a graph which displays the *chance of occurring* of amplitude fluctuations as a function of amplitude, we see the well-known bell-shaped curve, the **Gaussian** or **normal distribution.** Its 90^0-rotated version is also drawn in fig. 15.1. (It will look familiar to you: we have met this function as a window function in section 13.) For an obvious reason this type of noise is also called *Gaussian noise*.

Naturally, the intensity of the deviations from zero will depend on a couple of things. First of all, as mentioned above, it will depend on the *temperature* of the material. Furthermore, the fluctuations can vary from extremely slow to extremely fast. Somehow this *frequency range* must be taken into account. Now the formula for the thermal noise intensity (power) occurring in an electrical circuit will be not completely surprising:

$$P_N = 4kTB \tag{15.1}$$

Here P_N is the noise power; T is the absolute temperature in Kelvin, and B the choice of the frequency range (bandwidth) of the fluctuations which should be taken into account. The symbol k represents the *constant of Boltzmann*. being $1.38 \cdot 10^{-23}$ which relates the energy of moving particles to temperature [1].

To express the noise in volts (the amplitude domain) we must realize that the voltage causes an electrical current through the conductor (the wire) and that the power is the product of voltage and current $P = V.I$ (see the box **OHM'S LAW** in section 1). The current is the voltage divided by the resistance: $I = V/R$ which means that $P = V^2/R$ or $V = \sqrt{P.R}$. Combining this with formula 15.1 produces the formula for the thermal noise voltage produced by any electrical circuit:

$$V_R = \sqrt{4kTBR} \tag{15.2}$$

Of course, this voltage is the theoretical rms (root mean square) value. In practice the mean must be taken over long times as the power of short time intervals varies a lot because of the unpredictability of the real fluctuations. In fig. 15.1 you can see that in general, the amplitudes remain within a limited area from zero. Although any amplitude is possible when the values are unpredictable, the chance that high deviations from zero occur is extremely low, caused by the exponential decay of the probability curve. In fact, the rms value is equal to the standard deviation of the Gaussian curve, like displayed in fig. 15.1.

What can we find out about the spectral properties of this thermal noise? In fact, we cannot think of sine wave components at all: it would be extremely unlikely that at some time interval the fluctuations change like a sine wave. But if we are *forced* to transform the noise voltage into the frequency domain we define the noise being assembled by sine waves, no matter how many of them are needed. Then we can say that there cannot be a frequency which *on average* has a higher or lower amplitude than

1 Numbers like 1.38×10^{-23} sometimes are notated as 1.38e-23. The e stands for *exponent*.

other frequencies: all frequencies have the *same* chance to occur even if their amplitudes are very low. In other words, the spectrum of this kind of noise *on average* must be a horizontal line at some (low) level. For the spectrum to have some relevance, the power of each point must not be zero. But then it would mean that the total power of the spectrum adds up to infinity because the number of points of the spectrum is infinitely high. This cannot be true in practice; therefore, we will break up the total spectral range in small spectral bands (or frequency **bins**) and estimate the average power in each band. To this purpose the term *spectral density* is used. The wider the bands, the higher the probability that frequencies within a band will occur.

Now, if we make a spectrum of such a noise, taking a sufficiently long section to reach

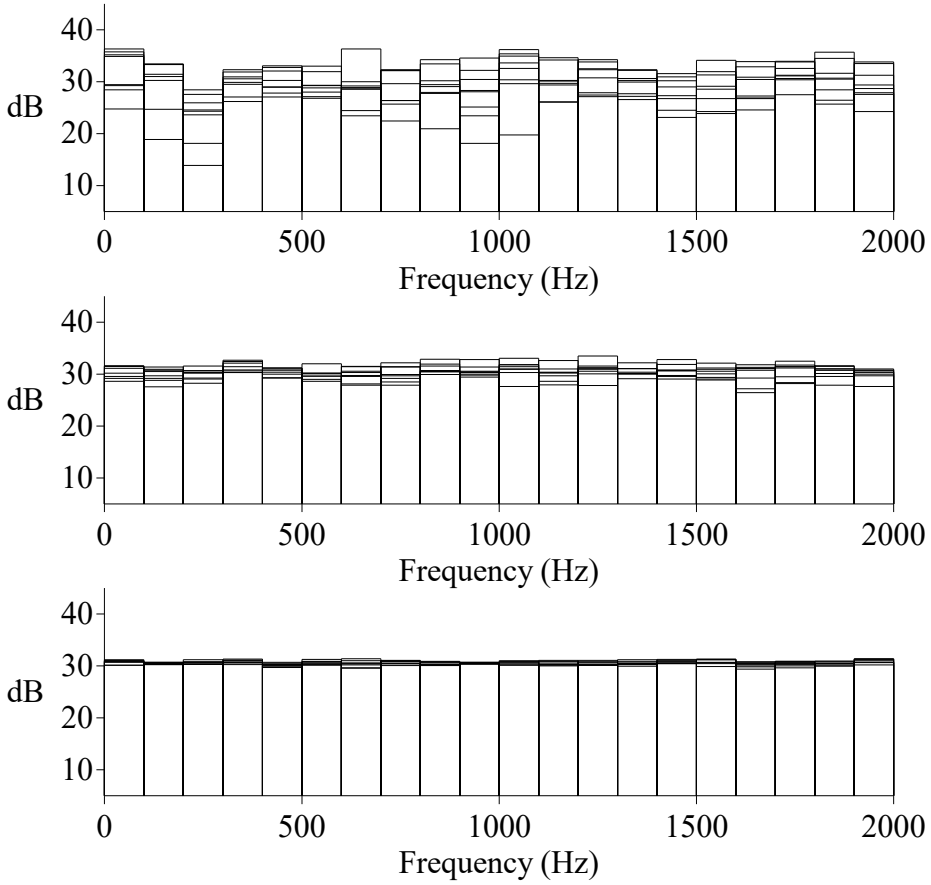


Fig. 15.2. Spectra of white noise consisting of 100 Hz frequency bins, repeated with newly generated noise and displayed on top of each other. Top: sound length is 10 ms; center: sound length is 100 ms; bottom: sound length is 1 s.

a reliable average, we get a straight horizontal line: the noise is **white**, named after white light which contains all wavelengths (however, not exactly in equal intensities). Fig. 15.2 shows spectra with 100 Hz bins of various lengths of white noise. The white noise sounds are repeatedly generated (eight times) and their spectra drawn on top of each other to show their variations. It is important to realize that the spectrum of a sound

already has *bins* which are defined by its length: the DFT of, for example, a 0.1 second sound has spectral points that are 10 Hz apart. Applying 100 Hz bins for this spectrum will average only 10 points per bin. Analyzing noisy signals (fricatives in speech, for example, see section 21) therefore needs attention as regards to proper averaging.

White noise is often artificially generated for measuring purposes. For example, the frequency behavior of signal processing equipment, like amplifiers and filters, can be estimated by feeding a white noise signal to the input and look at the spectra of the output.

Naturally, some noisy signals may have spectra which are not flat. One type of noise that is widely known is *pink noise*. While white noise has a constant energy per hertz (on average), pink noise has a constant energy per *octave*. As the frequency range of each next octave is twice the range of the current one, its energy is proportional to $1/f$ which is equivalent to a roll-off of 3 dB per octave. This type of spectrum seems to apply to many natural sounds (like long term speech, heart beats, air flow, etc.). The reason why this is so has probably something to do with the fact that fast acceleration of material with a particular mass needs more (kinetic) energy than slow acceleration so that the sound intensity must decrease when the frequency increases, provided that the energy is evenly distributed. Many investigations on this subject are still ongoing.

Another type of noise is *brown* noise which has a roll-off of 6 dB per octave. This is the result we get when white noise is integrated. The simplest electronic low-pass filters are integrating circuits, which probably explains why this type of noise exists. Their practical value is limited.

Noise types with opposite spectral slopes exist too: *blue* noise with a spectral increase of 3 dB per octave and *purple* noise increases 6 dB per octave. Other types of noise can have some spectral areas that are enhanced or attenuated, like grey noise which has a spectrum corrected for the human hearing sensitivity differences for different frequencies. For that purpose, its spectrum is multiplied with a reversed *A-weighting* curve (more of this curve in part B, section 27.4).

You can run DEMO 15.1 to listen to examples of various types of noise.

The unwanted noise generated in electronic amplifiers or *within* some types of microphones is very much like white noise, or like pink noise. That is not surprising because the necessary amplification of very small signals (from microphones or earlier recording media) has to be very high so that the thermal noise is highly amplified too. With the figure for *signal-to-noise ratio* (S/N) the sound quality as regards the noise is expressed in dB. Some practical aspects about this S/N ratio are mentioned in part B of the book.

16. Correlation

In section 6 we learned that the magnitude of the Fourier components can be found by estimating the cross-correlation factor of the signal and each sine or cosine F_0 multiple. With this wave comparison tool, we can do interesting things. Suppose we want to find a specific section of some duration within a long noisy signal (for example to know when a radar impulse echo was received back after sending it). The answer is: simply shift the pulse time function along the entire length of the noisy signal and calculate the cross-correlation factor at each position. A graph will emerge, displaying the cross correlation as a function of the time shift: this is the **cross-correlation function** (cc). At the position of the reflected radar pulse (the echo) a peak in this graph can be expected. See fig. 16.1 for an example of such a function. In the waveform of the receiver the reflection of the pulse is masked completely by the noise, as the center of the picture shows. From this cross-correlation the position of the reflected pulse can be estimated accurately. In the time lapse between the moment of sending the pulse and

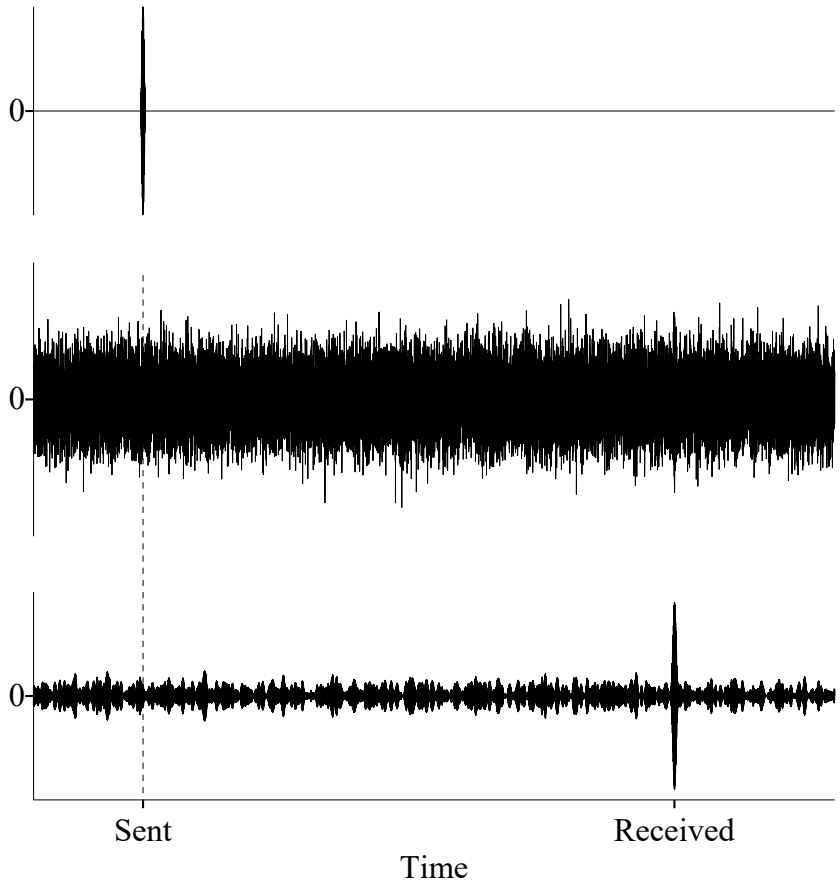


Fig. 16.1. Cross-correlation of radar pulse (top) with its reflection received in noise. The reflected pulse is not visible in the received waveform (center). After cross-correlation the position of the reflected pulse can be estimated accurately (bottom). The pulse in this example is a short-windowed sine wave.

receiving its echo the wave will have been sent forward and then backwards. It will be clear then, that the distance from the transmitter to the obstacle can be found by applying the formula:

$$d = \frac{\Delta f \cdot c}{2} \tag{16.1}$$

where d is the distance in m, Δt the time lapse and c the propagation speed of the wave. (For radar signals c is about 3.10^8 m/s, for sound waves in air c is 340 m/s, for sound waves in water c is about 1500 m/s.)

Another example of a cc application is the determination of the angle of a sound source with respect to the position of two microphones. In fig. 16.2 this is shown schematically. The placement of sound source and microphones is seen from above. The two microphone signals are recorded as a *stereo* sound which preserves the time relation between the two sounds. The distance from the source to the right mic is shorter than the distance from the source to the left mic so that the right signal will be ahead of the left signal. Cross-correlation of the left and right signals from this stereo sound can accurately reveal the time difference. A practical example is shown in fig. 16.3. The sound applied is an arbitrary "rasping" sound which was recorded out of doors to prevent reverberation effects. In the figure you see that the waveforms of the left and right signals (upper and center part, respectively) do not show any resemblance so that it is impossible to determine the time shift of the signals by looking at the waveforms. From the cc graph, however, we see that the correlation has a pronounced maximum at -2.15 ms which means that the right sound was 2.15 ms ahead of the left sound.

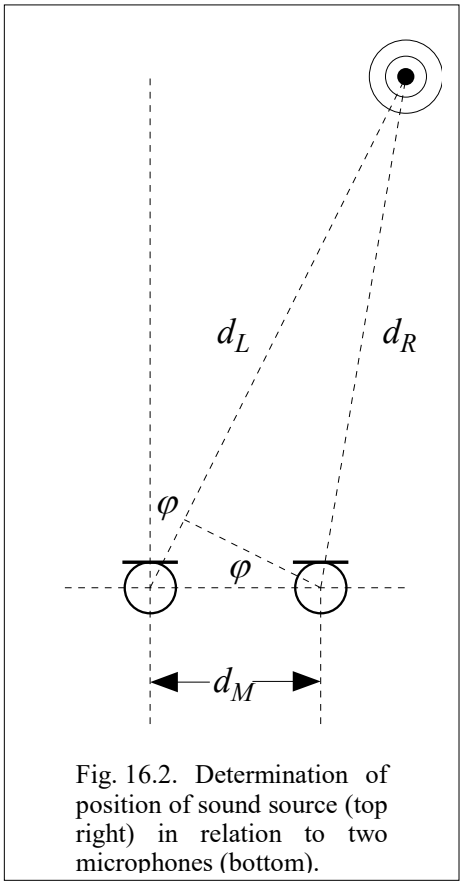


Fig. 16.2. Determination of position of sound source (top right) in relation to two microphones (bottom).

Fig. 16.2 shows that the sine of the angle φ is practically equal to $(d_L - d_R)/d_M$ when the distance from sound source to microphones is much greater than d_M, the distance between the two microphones. We know that the time for the sound to travel the distance $d_L - d_R$ *is* equal to the time shift we see in the cc graph. The angle φ can therefore be found by using the formula:

$$\varphi = \arcsin\left[\frac{\Delta t \cdot v_S}{d_M}\right] \tag{16.2}$$

where Δt is the measured time shift, v_S the propagation speed of sound in air (340 m/s) and d_M the distance between the microphones (which was 1 m in this case). Using the values from this practical example gives us the angle φ: 0.82 radians or 47 degrees.

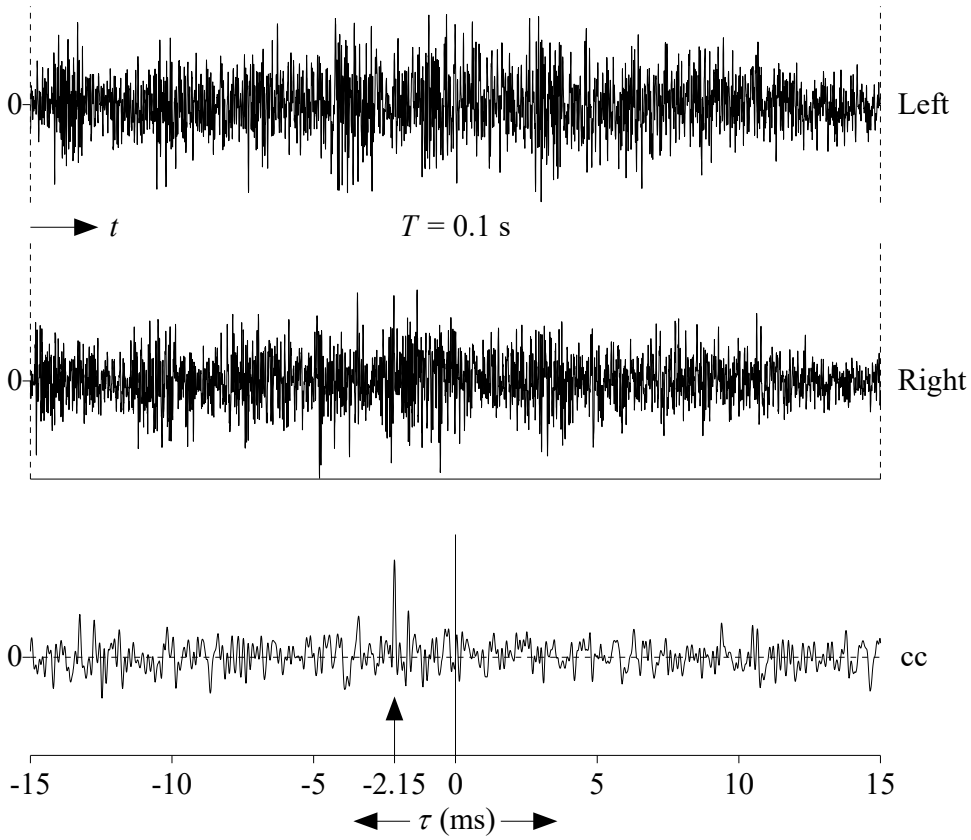


Fig. 16.3. Cross-correlation of the left and right channel of an outside stereo microphone recording of rasping noise. The direction of the sound source can be determined from the time-shift (τ) of the position of the peak in the cc function.

CROSS-CORRELATION

Each point of the correlation function is found by multiplying the one function with a time-shifted version of the other function, and calculating (integrating) the encompassed area, just like the process of convolution discussed earlier before, *leaving out the reversion* of one of the functions. The time shifts are again performed in infinitesimal steps over the whole time range.

We can simply modify the second formula from the CONVOLUTION box, using the same shift variable τ again:

$$r_{xy}(\tau) = \int_{-\infty}^{\infty} f(t) \cdot h(t+\tau)dt$$

How did we arrive at these cross-correlation graphs? From what was said before, we know that each point of the cross-correlation *function* represents the cross-correlation *factor* for that shift position. And each cc factor is found by multiplication of one signal with the other one, which is shifted, whereby the shift can be forward or backward in time. Now, that sounds familiar, doesn't it? In section 14 about convolution in the time domain we did the same *after we reversed in time one function*. Indeed, the only difference of correlation with convolution is *omitting* this reversion of one function. In the box called **CROSS-CORRELATION** you

can see that the formula for the cc function is equal to the convolution's formula, except for the opposite sign of the shift variable.

When the correlation function is carried out on a signal correlating it *with itself*, the resulting function is called the **autocorrelation function**. We know how to do that now, but, what can we do with it?

For an example of the application of autocorrelation (ac), let's start with the top of fig. 16.4 where we see a part of a periodic waveform of a (speech-like) sound. Suppose we would like to measure the fundamental frequency (F_0) of this part. We could extract it on sight from the waveform, if we measure the time interval of the period. But we must not be fooled: we must not simply take the distance between the highest peaks because we can see that that distance varies too much. We must look at the complete waveform to see what part of it is repeated *as a whole*. We can do that without actually knowing how our brains perform that task. So, using a method to measure the period automatically and reliably, demands an objective way to compare the different sections of the waveform. The obvious method, naturally, is using an autocorrelation of the sections. The bottom part of fig. 16.4 displays this autocorrelation function. It reaches its maximum at a shift of zero: of course, the parts are equal then. When the shift increases (or decreases in the other direction) a new peak will emerge exactly when the time shift is equal to one period. At each next shift of one period a new peak will emerge. These local maxima can be read from the waveform very easily as they stand out clearly from the other peaks, so that the F_0 can be determined accurately.

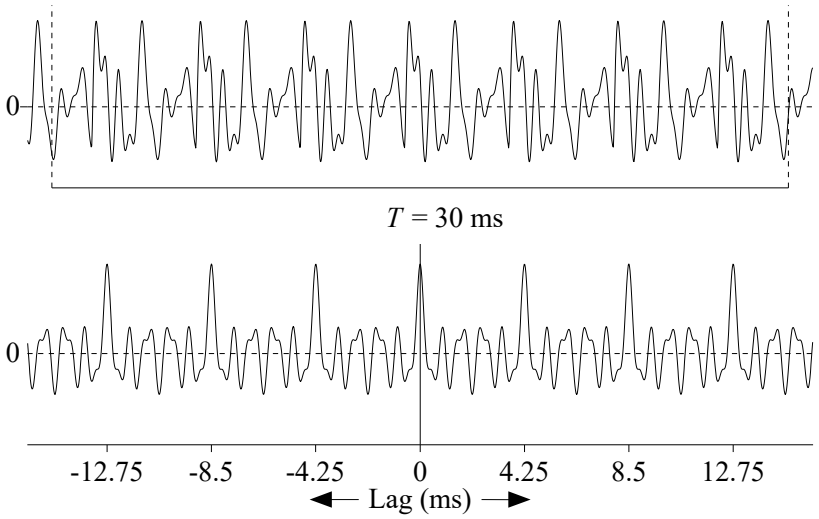


Fig. 16.4. Autocorrelation of a periodic sound, using a 30ms part of it.

You may ask, "Why not take the spectrum of the waveform and estimate the lowest component?" Of course, you can do it like that. However, if the F_0 is varying as a function of time (which, in the first place, is always the case in speech sounds) we want to select a small part of the sound in order to follow the F_0 variations. You may

remember from previous sections that the spectral *bins* have a width that is equal to $1/T_I$ hertz where T_I represents the length of the sound interval chosen. In the case of speech sounds, this interval should not comprise a much larger section than a few periods of the waveform, which means that the accuracy of the measurement from the spectrum is quite limited and will be much less than the accuracy of the ac. Another disadvantage of the spectral method is the possible presence of low frequency components (from 'airy' speech or background noise) of the speech signal that may be weak and should not be considered as an F_0 of the sound. The method of using the strongest spectral component cannot be applied here because the highest peak is not always the fundamental.

As an example of the F_0 measurement of a speech utterance using ac, see fig. 16.5. This method to measure the F_0 as a function of time for speech sounds (the *pitch contour*) has been built-in in the Praat program.[1] In part B of the book we will go into more detail about measuring the pitch contour.

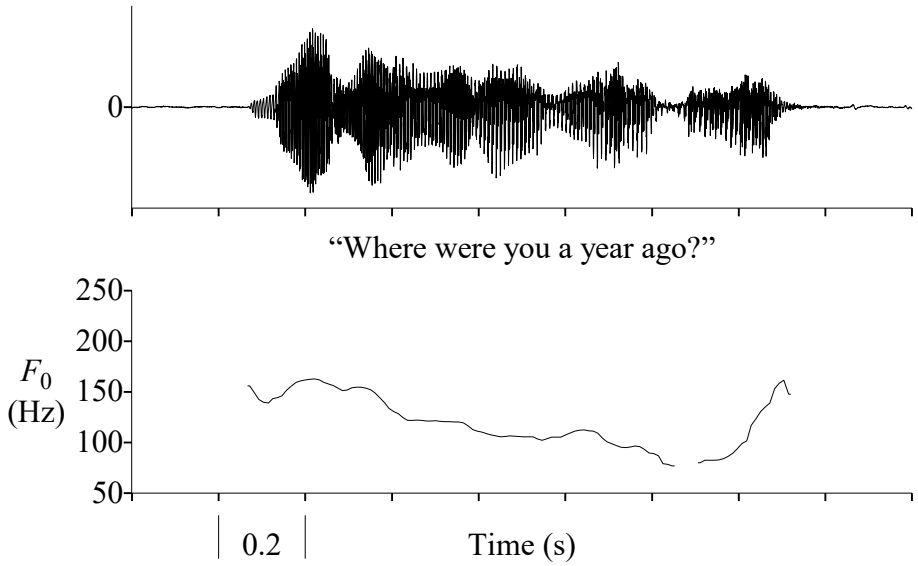


Fig. 16.5. Waveform of a spoken sentence with its *pitch contour*. Each point in the pitch graph is defined by the peak distances of the local autocorrelation functions.

For a second example of an ac application, please run DEMO16.1. A fair amount of (white) noise is added to a sound containing three sinusoidal waves. (The example used here is a musical chord.) The noise intensity is greater than the intensity of the sine waves (the signal-to-noise ratio is -2 dB). You can hear the 'clean' version and the noisy version in sequence. After autocorrelation of the noisy signal with an arbitrary section of 0.1 s you can hear that the sound quality has improved greatly: the noise level has

[1] The word pitch refers to the subjective impression of frequency. The program Praat ignores possible perceived deviations from 'real' fundamental frequency values.

been suppressed substantially, whereas the sound of the chord remains practically identical to the clean version. The waveform, however, has been changed a lot. The difference, as you may know already from section 8 about filtering, is caused by the change of the phases of the frequency components. (See also fig. 16.6.)

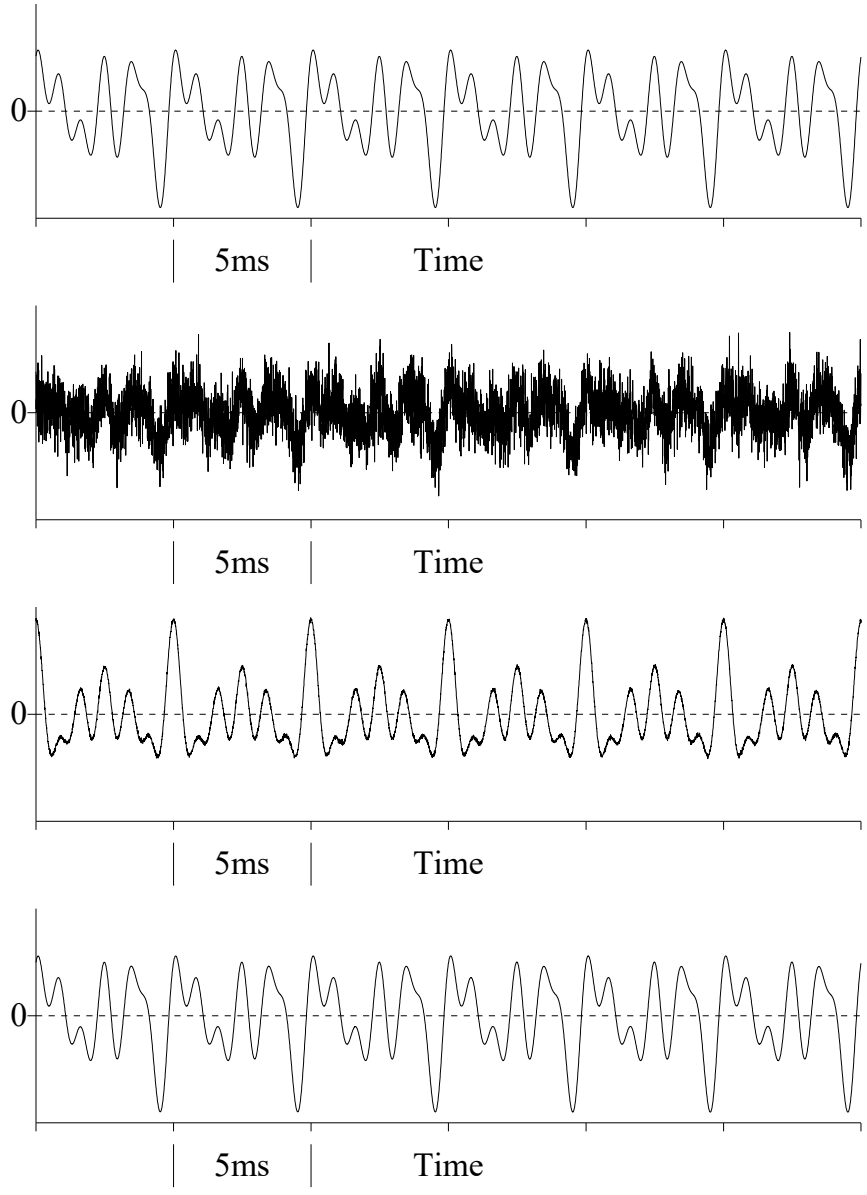


Fig. 16.6. From top to bottom: 200 Hz chord sound made of 4 added sine waves, white noise added, autocorrelation with a section of 0.1s of the noisy signal, ac wave convolved with noisy section.

As you can see, each period of the ac is symmetrical, as expressed by the following formula:

$$r_{xx}(-\tau) = r_{xx}(\tau) \tag{16.3}$$

which means that the autocorrelation is an *even* function, and in its spectrum, as we learned from section 6, all sine components are zero if we define the time origin at a point of symmetry, i.e. at one of the peaks of the ac function.

The last step of the demo takes the *convolution* of the ac result and the same noisy section we extracted before. If you now listen to the sound you will hear even less noise. What's more, the resulting waveform looks as if it has been *preserved in its entirety*. It looks the same as the clean chord signal. How is that possible?

To explain this surprising result, we must return to the spectral properties. In section 14 we learned that convolution in the time domain requires the reversing of one of the time functions. In the frequency domain the spectra are multiplied. When we multiply the spectra in the case of two equal signals, we know that each component of the spectrum, regarding as a rotating vector, is multiplied by itself. That means that the amplitude is squared and the phase angle is altered due to the changed ratio of *x* and *y* values (in fact the angle is *doubled*, which follows from the trigonometry). In section 11 we read that if we change the sign of all sine components of a spectrum, its time function runs backward. Consequently, *if we do not reverse* one of the signals, as occurs in autocorrelation, we actually multiply each rotating vector with an equal vector that runs in the *opposite direction*. (This is the *complex conjugate*. In appendix II this is explained

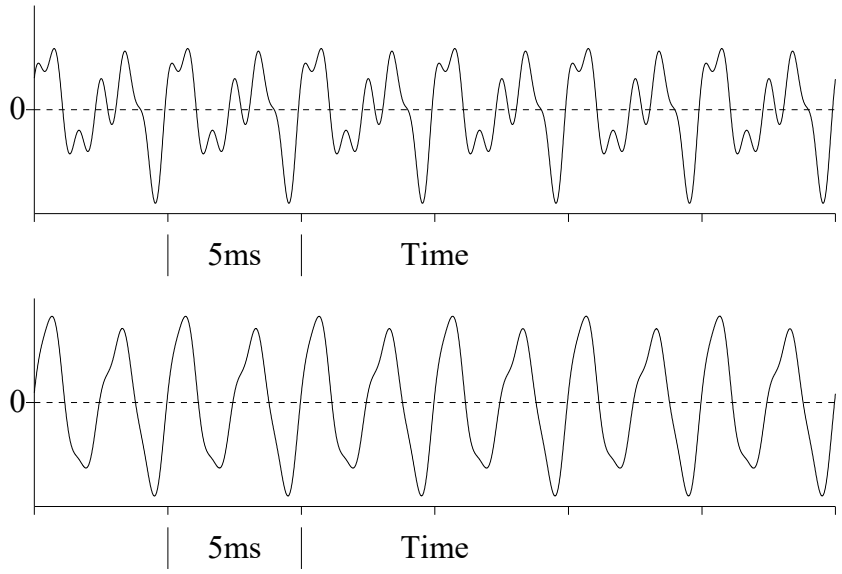


Fig. 16.7. Top: waveform of chord with different spectral component magnitudes. Bottom: waveform after ac and convolution as done in fig. 16.6.

in more detail.) The result is that after spectral multiplication all amplitudes are squared and *all phases are turned to zero*. That explains the *zero-phase* waveform of the ac function. Now, in the last stage, if we multiply *again* with a normal spectrum with rotating vectors in the positive direction, which is equivalent to *convolution*, the *original* phases are added to all zero phases, so the original phase relations are restored completely. The amplitudes are again multiplied so that the final amplitudes are equal to the *cube* of the original amplitudes.

Because all frequency components of the demo's chord have equal amplitudes, the ratio of the component magnitudes will remain the same. Of course, if the amplitudes differ substantially, we would not be able to restore the waveform. See fig. 16.7 where the same autocorrelation/convolution combination is used for the chord with relative component magnitudes 1, 0.6, 0.2 and 0.5.

The noise suppression method using autocorrelation is based on the fact that the individual spectral noise components are much lower in magnitude than the signal's components we are hoping to retain. When the magnitudes are squared (or cubed) the low-level components are weakened in favor of the high-level components.

With this insight, it is easy to think of an alternative way for noise suppression, preserving the signal's phase relations. We can operate in the frequency domain: first we make a spectrum of the complete signal, then calculate the spectral component magnitudes, then multiply the cosine and sine components of the complete spectrum with the magnitude values. This will preserve the amplitude ratios of the sine/cosine

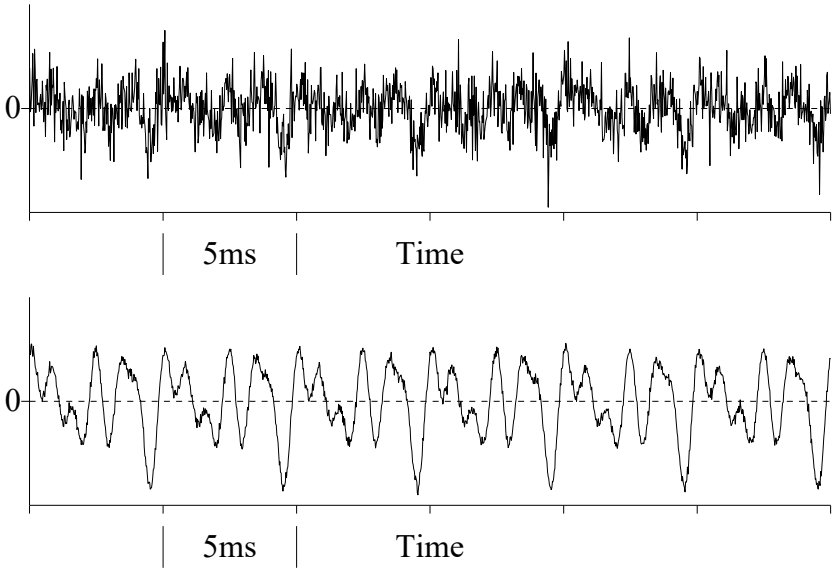


Fig. 16.8. Noisy chord sound of fig. 16.6 (top). Noise suppressed by multiplication of its spectrum with magnitude values (bottom).

pairs so that all phases remain unaltered. The noise suppression will then be equal to that of the autocorrelated signal. Fig. 16.8 shows the result. Of course, this method also only works when the signal's spectral component magnitudes do not differ much, as the ratio of their magnitudes is altered due to the non-linear processing. For example, the ratio of, say, 5 and 2 is 2.5; after squaring, the ratio has increased to 25/4 = 6.25. (Also, when the magnitudes of the signal's cosine and sine components differ much, the final phase will have been changed considerably.) The idea of modifying the spectral values dependent on the lengths (or magnitudes) of their vectors is known as Wiener **filtering.** The term *filtering* suggests some kind of linear function as you may remember from section 8. However, Wiener filtering is actually a non-linear process caused by the squaring of the magnitudes. The requirements for linear systems, mentioned in the box called **LINEAR SYSTEMS** in section 8, are therefore not met.

For signals which are not steady (like speech, for example), Wiener filtering can be made to depend on the local spectra of the sound, using a moving window: the *dynamic Wiener filter*. Instead of simply squaring the magnitudes it is possible to use other functions which offer better compromises between noise suppression and signal component magnitude ratio distortions. In part B we discuss noise suppression in practice in some more detail.

17. Sampling of sounds

Up to now we have dealt with signals that are *continuous*. This means that the signal value changes gradually and continuously as the time goes on. No matter how fast or how irregularly the value changes, given a specific time interval, the signal will have gone through all values between the maximum value and the minimum value within the interval. There are an infinite number of different values and an infinite number of different time points in each interval, no matter how short it is. Almost all phenomena in nature behave like this (temperature changes, light intensity changes, mechanical movements, etc.). So, do air pressure changes, which thus include sounds. The signals change *analogous* to the gradually changing phenomena.

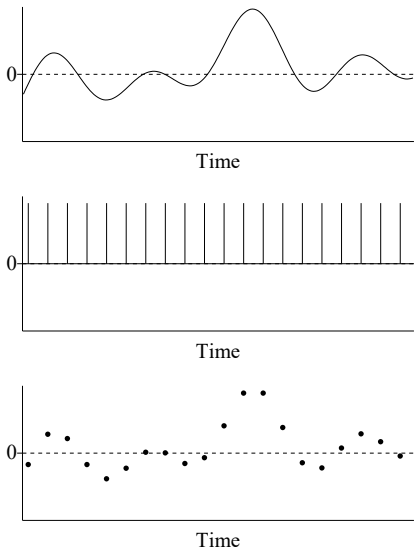


Fig. 17.1. Sampling of sound in time. The analog sound is multiplied with a delta pulse series to get discrete values.

Signal analysis and processing being generally complicated, we will need to use a computer suitable for that purpose. Because the computer deals with numbers only, we must somehow convert the analog signals to numbers. We cannot store an infinite quantity of numbers in a digital machine like a computer[1]. Therefore, the signal must be *sampled* which means that successive amplitude values are measured at discrete points in time that are so near to each other that the signal is 'followed' with sufficient accuracy. All sample values then are converted to numbers by the **ADC** (analog to digital converter) and all these numbers are fed into the computer, together with the information that represents the pace of sampling, the **sampling frequency**. Now this sampling in time solves only part of the problem: we need only a limited *number of samples* to represent a specific signal but we still need an infinite *number of possible values* for each sample, as the signal can have any value between two different arbitrary values. So, we must not only sample in time but also in amplitude. The signal has to be *quantized*. Only then we have our signal made completely *digital*. We will discuss this amplitude digitizing problem later on. So, to see what 'sufficient accuracy' on the time axis is, we will assume for now that we can store the exact *amplitude* values.

1 Theoretically there is no signal analysis or manipulation that cannot be done by analog electronic hardware or even an *analog computer*. The complexity of the electronics for performing some types of signal analyses or manipulations, however, would be extremely high and the problems with stability of values would be almost impossible to solve. With the digital computer there is virtually no limitation in accuracy and stability.

In fig. 17.1 we see a part of an arbitrary signal, and the result after sampling it in time. The figure shows that the process can be seen as a multiplication of the analog signal with a series of delta functions (Dirac pulses) with equal amplitude values (in section 8 we saw this delta function concept as an excitation function of the resonator). The result of the multiplication is one number at each position of a pulse. Each number represents the amplitude of the signal at that position in time. What does that mean if we look at it in the frequency domain? We know that the spectrum of one Dirac pulse is flat and horizontal: all frequencies are evenly distributed. Repeating the pulse means that the pulse sound becomes periodic and that only multiples of $1/T_S$ can exist in the spectrum, T_S being the time between two adjacent pulse peaks. The spectrum then is also 'sampled', but in the frequency domain: it consists of spectral lines, at a distance of $1/T_S$, all of the same amplitude, and with nothing in between. We already saw all this in section 8. Multiplication in the time domain means convolution in the frequency domain (as explained in section 10), which implies that at each spectral line, sum and difference frequencies of the spectral line frequency and all spectral components of the analog signal will emerge. Consequently, the spectrum of the original sound, together with its mirror image, is *repeated ad infinitum* along the frequency axis.

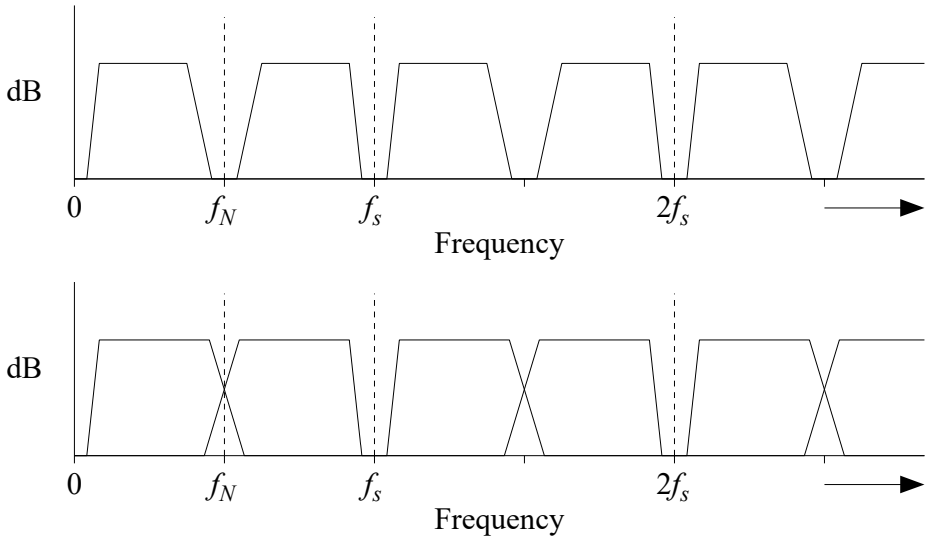


Fig. 17.2. Top: spectrum of sampled analog signal of properly limited frequency range where f_s represents the sampling frequency and f_N the Nyquist frequency (see text). Bottom: same, but with insufficiently limited frequency range: alias frequencies emerge.

In fig. 17.2 a few of these adjacent spectrum-and-mirror sets are displayed. (Here the spectrum of the analog signal is shown as a limited frequency band within which all frequencies of the signals can occur.) Now you will see the limiting factor of sampling in time: the high ends of the spectra may interfere with the low ends of the next spectral set, thus producing shifted frequencies that are *placed in the original spectral range*

and which then may become audible, the **alias** components. So, the distance between the spectral lines of the delta pulse series must be sufficient to overcome this problem.

In other words: the spectral range of the original signal should be limited to one half of the frequency range between two adjacent spectral lines of the pulse series spectrum. This range is equal to the fundamental frequency of the pulse series which is the *sampling frequency*. Therefore, the sampling frequency should be at least *twice* the highest frequency in the original sound. To put it differently: for a sine wave to be sampled without loss you need at least two samples per period.

We have now proven **Shannon's sampling theorem**, also known as the **Nyquist sampling theorem**, simply by reasoning. Nyquist mentioned this theorem as early as 1928 and Shannon proved it more mathematically in his 1949 publication. The term *Nyquist frequency* is commonly used for the highest frequency of a signal that can be sampled without aliasing, which therefore equals half the sampling frequency. Fig. 17.3 shows what happens when a sine wave is sampled with a sampling frequency that is too low. The appearance of the alias frequency can be clearly seen.

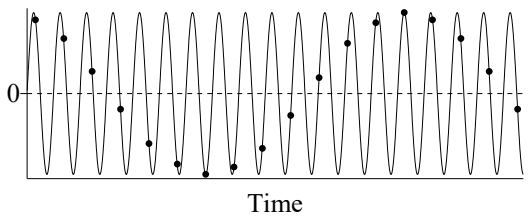


Fig. 17.3. Appearance of alias frequency by undersampling of sound in time.

The fact that the upper frequency limit of human hearing is less than about 20 kHz (kilohertz) means that the minimum sampling frequency for a sound of good quality should be at least 40 kHz. To avoid the sampling of too high frequencies of sounds the analog signals must always be *low-pass* filtered to attenuate the frequencies above the Nyquist frequency to an acceptably low level, which is known as **pre-filtering**. In section 8 about filtering you read that the slope of the filter always takes some frequency range: it cannot be made infinitely steep. What's more, the steeper the filter, the longer the reaction time, as mentioned at the end of section 8. Most signals are not steady and may change very rapidly, which means that the filter slope steepness should be limited. Therefore, the sampling frequency should be chosen sufficiently higher than the Nyquist frequency. In practice, however, this **guard band** (the frequency range allocated to the slope) can be relatively small (say, a few kilohertz) thanks to the development of sophisticated *linear phase* filters that have a constant time delay for all frequencies. The subject of analog linear phase filters falls outside the scope of the book. Suffice it to say that the commonly used sampling frequency in audio equipment is 44100 Hz or 48000 Hz, so that their frequency range is sufficient for high quality sound.

How can we, after the signal processing activities we carried out in the computer, transfer the digital signal to its analog version to make it audible? We cannot simply convert the numbers in the computer to voltages and feed them to an amplifier and loudspeakers. First of all, the sequence of voltages must be fed into the amplifier with the proper *timing*, which means that we must feed the voltages in exactly the same pace as the sample frequency by the insertion of a period of time between the successive amplitude values. Only then the frequencies will be correct. What to do with the voltage during these time insertions? If the voltage is zero then, theoretically, the amplifier will only receive voltages during infinitesimally short times so that the result is zero. Moreover, the frequency range of the amplifier and speaker would be too limited to process very short pulses.

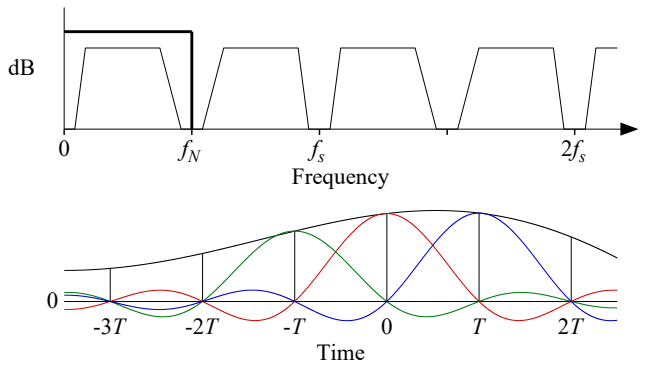


Fig. 17.4. Reconstruction of analog signal by ideal low-pass filter. In the time domain the samples are convolved with the sinc function of the filter.

In fact, we need an ideal low-pass filter to limit the repeated spectra of the sampled signal to the Nyquist frequency (see fig. 17.4). In the time domain this means a convolution with a sinc function, as we have seen. This results in a sinc function $\frac{\sin(\pi f_s t)}{\pi f_s t} = \frac{\sin(\pi t/T)}{\pi t/T}$ at each sample position, which is weighted with the sample's value. We can conclude from the sinc formula and the figure that the zero points of the sinc at a specific sample coincide exactly with all other samples so that each sinc is independent of all others. Each sinc curve contributes to the values between the samples and the sum forms the continuous time function version of the sampled one. This is known as Shannon's *reconstruction theorem*.

Because of the fact that the sinc function theoretically runs from $-\infty$ to ∞ it has to be truncated in practice (and preferably windowed as well). Therefore, because we need to use an analog filter, the complexity of performing sinc filtering is the reason that this method is not used in practice. A more practical solution is to sustain the voltage value of a sample until the next sample arrives. The signal will then look like the 'staircase' wave in fig. 17.5. (Because of the straight lines between successive values this is often called **zeroth order hold**, to distinguish it from other, higher order functions which are curved.)

To investigate its implications, we can see this signal as the convolution of two signals: one is the sampled signal with *zeros* between the samples, the other is a *single pulse* with duration of the sample time T_S, as also displayed in the figure 17.5. Therefore, according to section 14 where this convolution in the time domain was explained, to get the spectrum of the staircase wave we must *multiply* the spectrum of the sampled signal and the spectrum of the *single pulse* which is a sinc function. The zeros of the sinc function are positioned at multiples of $1/T_S$ and coincide exactly with the sampling frequency multiples. The original signal's spectrum is positioned in the range from 0 to

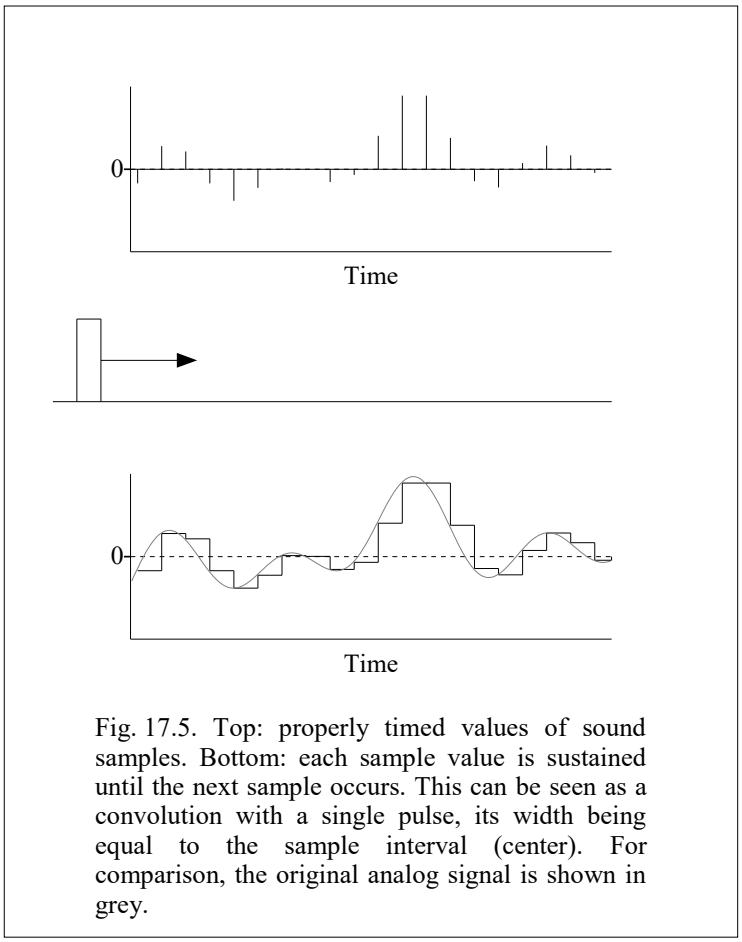


Fig. 17.5. Top: properly timed values of sound samples. Bottom: each sample value is sustained until the next sample occurs. This can be seen as a convolution with a single pulse, its width being equal to the sample interval (center). For comparison, the original analog signal is shown in grey.

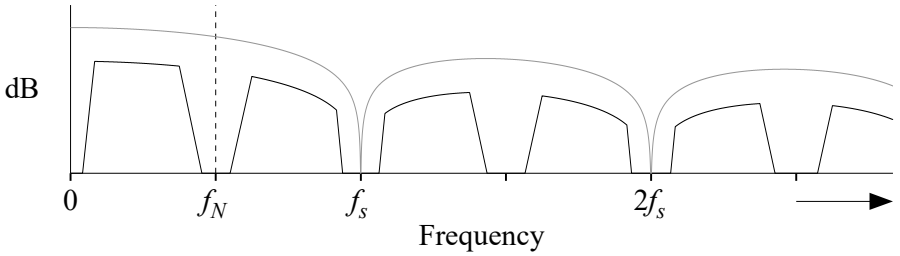


Fig. 17.6. Spectrum of sampled signal with sustained sample values. Spectral sinc function of single pulse shown in grey for comparison. (Scaling by multiplication results in a vertical shift on the log scale.)

the Nyquist frequency and repeated in the next spectrum-and-mirror image sets which are positioned around all zeros of the sinc function, see fig. 17.6. (The symmetrical negative frequency part of the spectrum is not shown, as we deal with power spectra.).

The obvious thing to do now is to low-pass filter the signal to suppress all frequencies higher than the Nyquist frequency to block their way to the input of the amplifier and

speakers. Although the higher frequencies are not audible for humans, most amplifiers will not behave linearly for very high frequency components which will produce unwanted audible shifted frequencies (called *intermodulation distortion* which will be described in part B). The low-pass filter needed, apparently, is identical to the low-pass filter used for the digitizing of the analog signal. Here, however, the filter is not applied for limitation of the original signal's spectrum (it was already limited before it reached the ADC) but to filter out the higher components that emerge by the nature of the sampling of the digital version of the sound. This is called **post filtering**.

Obviously, this pre-filtering and post-filtering must be done by using *analog* electronics. At the computer's input the signal only becomes digital *after* sampling by the ADC (analog to digital converter) and at the computer's output the filter is applied after the signal is made analog with a **DAC** (digital to analog converter). This pre-processing and post-processing of the signal is done in the sound card of the computer or in a separate ADC/DAC converter unit that is connected (digitally) to the computer.

One small problem still remains: as can be seen in fig. 17.6 the higher frequencies within the range from 0 to the Nyquist frequency are somewhat attenuated compared to the lower frequencies, caused by the shape of the *main lobe* of the sinc function. This roll-off (attenuation) is not dramatic: less than 4 dB for the Nyquist frequency. (This can be calculated from the sinc spectrum formula 13.2 by substituting $1/(2T)$ for *f*.) The highest frequency of the signal's range is lower because of the necessary *guard band*, allocated to the low-pass filter. For example: if this guard bandwidth is 10% of the Nyquist frequency the attenuation is at most about 3 dB. If we want to treat all audio frequencies equally, however, the frequency response of the DAC should be flat and horizontal. One solution is to high-pass the signal *in its sampled state* using a **digital filter**[1] with the right frequency response that compensates for the attenuation of the higher frequencies, the other is to perform this compensation using an *analog* high-pass filter connected with the output of the DAC. Because the amount of roll-off is not much of a problem, some computer sound cards do not compensate for this at all.

In practice there is no need to bother with this pre- and post-filtering: all the necessary electronics is included in the computer's sound card or converter unit. In order for us to understand the principles of conversion from analog to digital and vice versa, however, an explanation of why we need to filter is necessary.

Now we know about the limitations of sampling in time. We still have to look at the necessary sampling in the amplitude range, as mentioned earlier. The amplitudes cannot be represented by exact values because that would require numbers with an infinite number of decimals, or in computer language, an infinite number of bits. We must accept some inaccuracy which means we will end up with some discrepancy between

1 The name *digital filter* represents a special type of filtering, possible by the mere sampled state of the signals, rather than the 'classic' filtering of the sampled versions of the analog signals. This type of filters is described later on.

the real value and the value of its nearest higher or lower number, thus allowing for a specific minimum step in the amplitude range.

In fig. 17.7 the analog signal of fig. 17.1 is converted to a signal with a limited number of possible amplitude values, in this case 4 positive and 4 negative values and one for zero. The result consists only of amplitude values that can be represented by the numbers available. The difference with the real signal, the *error signal*, is also displayed (with its amplitude enlarged three times). It is obvious that the discrete value nearest to any analog signal value differs half a step at most, so that the amplitude of the error signal is always equal to half the step value.

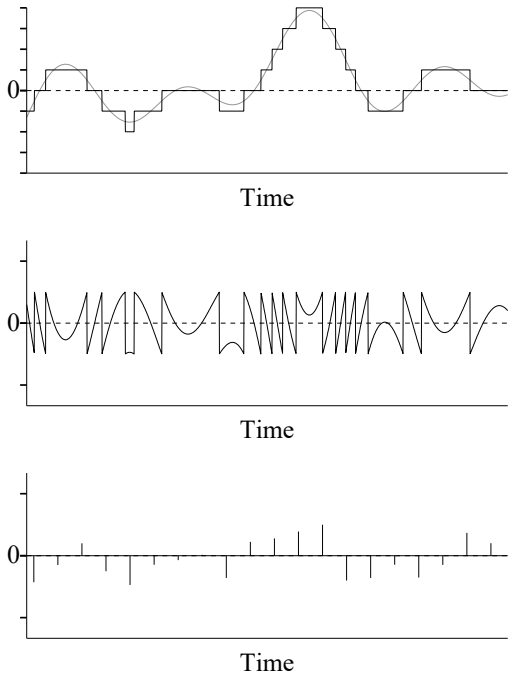


Fig. 17.7. Top: limited number of possible amplitude values. Original analog signal shown in grey. Center: amplitude error signal (enlarged). Bottom: sample value errors (also enlarged).

If we sample the signal in *time*, we know that its maximum amplitude error is also always a half step, no matter at which points in time where we sample the signal. Fig. 17.7 also displays the sample errors.

The error signals look like noise with limited amplitude. In fact, for truly periodic analog signals the error spectrum is related to the periods of the analog signal in a complicated way but the average spectra of these error sounds are very much like white noise. Mostly, signals in practice are not strictly periodic so that there exists no relation between the original signal and the error signal. The amplitude of the error signal can then be regarded as being completely unpredictable. To distinguish this type of noise from thermal noise the term **sampling noise** is commonly used.

To determine what error is acceptable, we can express the rms value of the sampling noise in relation to the rms of the analog signal: we would like to know the *signal to noise (S/N) ratio*. We know that the error amplitude will never exceed half the step value but we also know that every value between zero and the step value has the same probability

RMS of ERROR SIGNAL

Theoretically, we can put all absolute error amplitude values in order of magnitude. The result is a triangle function because the amplitude distribution is randomly uniform: $y = Ax$, where x = 0...1, A being the error amplitude. To calculate the root mean square value we first square the triangle function to get its power:
$P(x) = A^2x^2$ (x = 0 ... 1). To determine the overall power, we must calculate its mean which needs integration:
$P = A^2 \int_0^1 x^2 dx$. Evaluation leads to: $P = \frac{1}{3}A^2$
For the rms value we take the square root: $A_{RMS} = \frac{A}{\sqrt{3}}$.
A is equal to half the step value of the amplitude sampling so that we can write:

$$A_{RMS} = \frac{step}{\sqrt{3}}$$

of occurrence[1]. This distribution of values is called **random uniform**. For calculating the rms value of our error signal we should square each amplitude value, take the mean and then its square root (the box **RMS of ERROR SIGNAL** explains that the rms value is the amplitude step divided by 2√3). In our example the step value is 1/4 of the maximum signal amplitude, A_{MAX}. When, for example, the signal is a sine wave, its rms value is A_{MAX} divided by √2 (as explained in section 5). The signal-to-noise ratio is therefore:

$$S/N = \frac{A_{max}/\sqrt{2}}{A_{max}/(8\sqrt{3})} = 4\sqrt{6} \qquad (17.1)$$

We can conclude that the maximum signal to noise ratio of an amplitude sampled sound is equal to $m\sqrt{6}$ if m is the number of possible amplitude steps. The total number of steps needed, however, is twice as high because of the negative part of the analog signal, plus one step for zero. When the number of steps is much higher we may ignore this extra single step so that we can write:

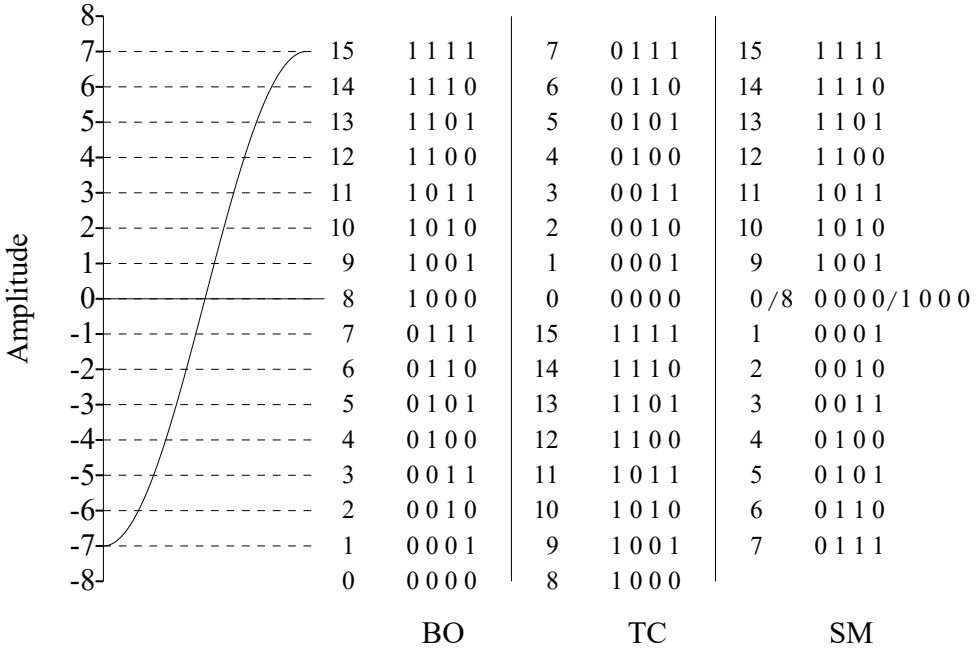
$$S/N = \frac{n\sqrt{6}}{2} \qquad (17.2)$$

where n is the total number of steps for the complete (*peak-to-peak*) signal.

The S/N value 4√6 from our example we can express in dBs using the formula 20·log(4√6). The result is about 19.8 dB which is not much, compared to a good quality sound intensity range of 90 dB or so. From formula 17.2 we can calculate that for a 90-dB signal-to-noise ratio, n should be about 25820. If the analog sound voltage, for example, has a maximum amplitude of 1, the 25820 voltage steps must cover a peak-to-peak range of 2 volts. One step is then less than 2/25820 volt, or 77.5 µV (microvolt). In our decimal number system, we can see that the *number precision* for this example of sound sampling requires 5 digits. As you know, the numbers in computers are *binary* where only two symbols for a digit are used (0 and 1) instead of 10 symbols. The binary equivalent for the decimal number 25820 is 110010011011100 which comprises 15

[1] Strictly speaking, at very high sampling rates the error values will be more related to the signal values.

digits (15 bits). For practical reasons in programming and electronics the number of digital digits always is a multiple of 4 so that actually the number of bits applied in sound cards and almost all audio equipment is 16. If all possible numbers within this range are used (which amounts to 2^{16}) the maximum S/N ratio is about 98 dB, more than enough for good quality sound. (See part B for a practical discussion about S/N ratio.)



	BO		TC		SM
15	1111	7	0111	15	1111
14	1110	6	0110	14	1110
13	1101	5	0101	13	1101
12	1100	4	0100	12	1100
11	1011	3	0011	11	1011
10	1010	2	0010	10	1010
9	1001	1	0001	9	1001
8	1000	0	0000	0/8	0000/1000
7	0111	15	1111	1	0001
6	0110	14	1110	2	0010
5	0101	13	1101	3	0011
4	0100	12	1100	4	0100
3	0011	11	1011	5	0101
2	0010	10	1010	6	0110
1	0001	9	1001	7	0111
0	0000	8	1000		

Table 17.1. 4-Bit example of different codes for ADC: binary offset (BO), two's complement (TC) and signed magnitude (SM).

The analog signal having a positive and a negative part, how are *negative* voltages represented by binary numbers? The first method that can be used is shown in the column marked BO of table 17.1 where a 4-bit code is used. In the first column the numbers simply run from the maximum negative amplitude to the maximum positive amplitude. The zero value of the sound is then 1000, halfway the number sequence. This is why this kind of coding is called **binary offset**, the offset here being 1000. It would be practical, however, to represent 0 volt by the digital number 0. This is possible by applying the coding from column two. Compared to binary offset, the most left bit (the most significant bit or **MSB**) is *inverted here* so that 0 volt is coded as 0000. This coding is called **two's complement** and is commonly used in computers and all digital audio equipment. One of the advantages is that the code of negative values can be achieved by simply *counting the digital numbers back from zero*, which can be practical in digital electronics. The general name for this audio coding in computer files is **PCM** (Pulse Coded Modulation).

Many different coding systems have been developed. In fact, any coding can be used provided that it is applied consistently in all signal analyses and manipulations within

one system. As an example, a third method is shown in the next column. Here the *magnitude* of the sound is represented by the three bits at the right of the codes and the sign of the sound wave is represented by the MSB (here 1 for positive and 0 for negative values). This is called **signed magnitude**. It is sometimes used in stand-alone digital electronic equipment for its symmetry: for inverting the value, simply the sign bit needs to be changed.

Because the smallest step of the amplitude is equal to the value of the bit most to the right (the least significant bit or **LSB**) of the code, the amplitude step is often referred to as LSB.

It is important to realize at this stage that the commonly used number representations in computers have a much greater precision than the 16 bits of the sound values. For calculations, analyses and processing the *integer numbers* have 32 bits or 64 bits which cover the ranges 0 to $2.1 \cdot 10^9$ and 0 to $9.2 \cdot 10^{18}$ respectively. Apart from that, in the computer *real numbers* are also used (which are not limited to the integer ranges), by applying *floating point arithmetic*, a subject that this book does not cover. However, the sample values coming from the sound card, or *from a sound file*, usually have a maximum precision of 16 bit[1], covering the range of numbers from 0 to 65535.

Now that we know how the analog signals are represented in the computer we should consider a few of the consequences. The next examples are illustrations of these consequences.

Example 1.

Even when the sampling frequency and number precision are sufficiently high for a proper audio frequency range and S/N ratio, some effects may surprise us. Have a look at fig. 17.8, where a sine wave is generated with a frequency of 400 hertz with a sampling frequency of 44100 Hz. Its period is 2.5 ms. When we take 6 complete periods of the sound the total length is exactly 15 ms. Its DFT therefore should show only one component at 400 Hz. but the DFT picture shows many spectral components that spread over the whole frequency range and are quite high around this frequency. (The components are multiples of 66.7 Hz, being the bin of 15 ms signal length, but the program interpolates the values in between). In the lower part of the figure, the same is done, but now with a fancy sampling frequency of 41600 Hz. Its DFT looks correct: only one component emerges at 400 Hz. How is that possible?

Let's look at the continuous spectrum of a once-occurring signal of, for example, three sine periods of 300 Hz, see fig. 17.9. The DFT of this signal produces only multiples of 100 Hz which all are zero, except at the center of the main lobe which produces only one component at 300 Hz, a mechanism already mentioned in section 13.

1 Some sound cards or other digital audio units are equipped with 24 bit precision ADC and DAC converters. In part B is explained why this precision is unnecessary in practice.

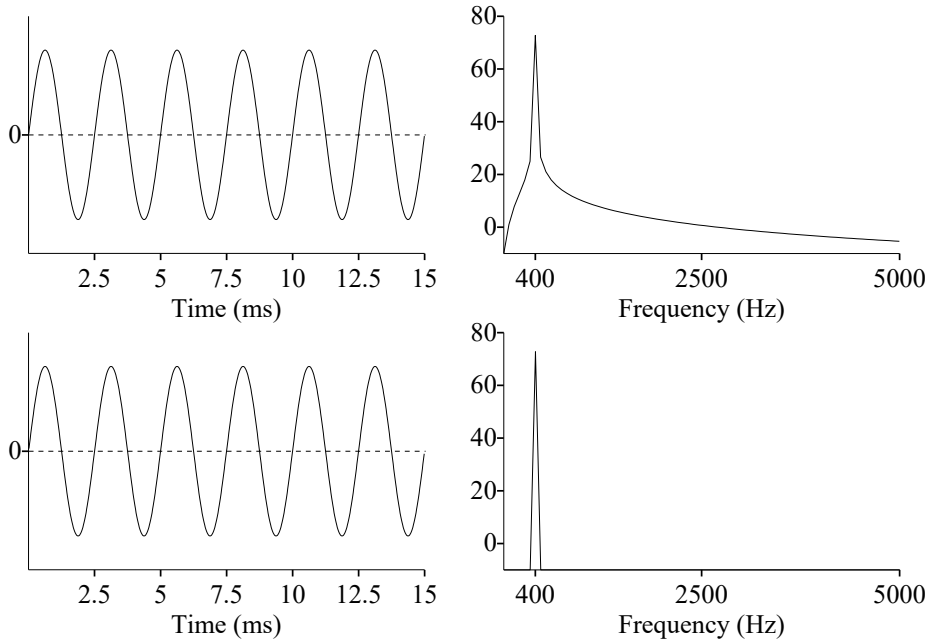


Fig. 17.8. Six whole periods of a sine wave of 400 Hz and its spectrum. Top: sampled with 44100 Hz, bottom: sampled with 41600 Hz.

Sampling of this signal at, for example, 1500 Hz repeats the spectrum with its mirror image around this sampling frequency and around all multiples of the sampling frequency. You can see that the DFT components (the multiples of 100 Hz) of this sampled sound coincide exactly with all zeros of the *repeated* spectra as well (and are positioned at the centers of the main lobes, of course). So far so good. If, for example, instead of 1500 Hz we applied 1750 Hz as sampling frequency, all repeated spectra with their zeros would shift a factor of 1750/1500 to the right and the DFT multiples do not necessarily coincide with these zeros any longer. Because the area of the side lobes of the sinc function is so broad (theoretically it is infinite) *alias frequencies* emerge everywhere at the DFT multiples and also in the range from 0 to the Nyquist frequency (spectral *leakage*). Therefore, the sampling frequency should be a multiple of the DFT *bin* (here 100 Hz). Indeed, in fig. 17.8 the bin is 1000/15 Hz and the second sampling frequency (41600 Hz) is a multiple of this bin, the first sampling frequency is not (44100 divided by 1000/15 is 661.5).

In practice, when sound is selected (perhaps taken from a longer sound) for analyzing its spectrum, the DFT bin width (which is $1/T$ if T is the signal part's duration) can be of any value, and the sampling frequency will not necessarily be a multiple of the bin width. To avoid the spectral leakage to some extent one should multiply the signal selection with a window that has low side lobes (Blackman, Kaiser) or no side lobes at all (Gauss) as described in section 13 on windows, of course at the cost of frequency resolution due to the spectral width of the window.

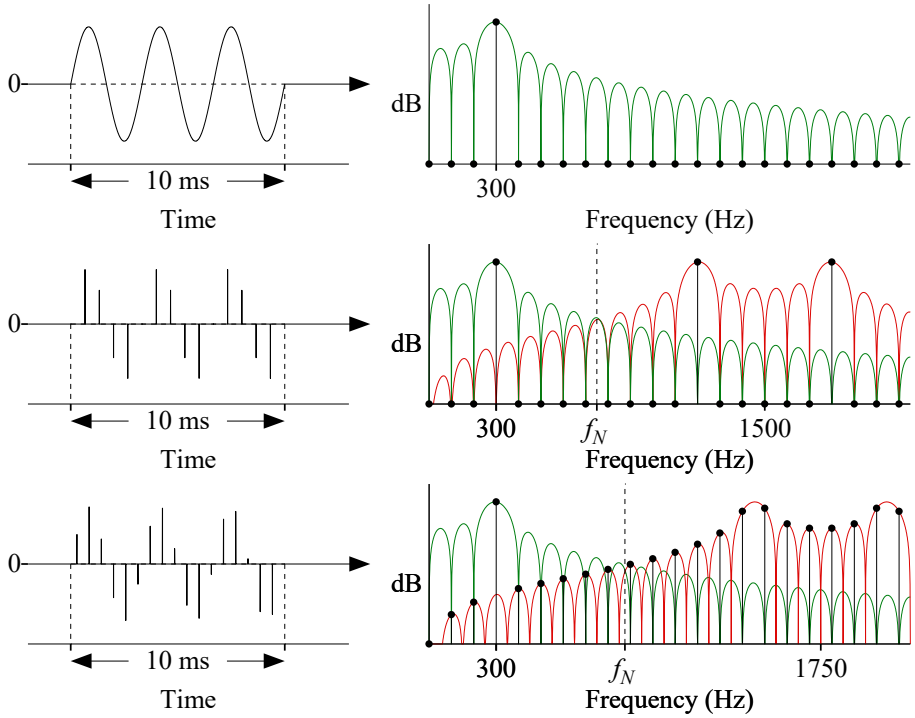


Fig. 17.9. Influence of sampling frequency on spectral leakage. Top: once-occurring signal of three periods of 300 Hz and its continuous spectrum (green). The DFT contains only a 300 Hz component. Center: same signal sampled with 1500 Hz. The repeated spectrum around the first harmonic of the sample frequency (red) causes no spectral leakage in the DFT. Bottom: signal sampled with 1750 Hz. The repeated spectrum (red) causes spectral leakage in the DFT.

In general, the frequencies of the components in the signals under consideration are not known so that a requirement to select an exact integer number of periods is not practical. The spectral leakage that emerges when the selection does not contain an integer number of periods is explained in section 13. Avoiding leakage caused by this phenomenon needs proper windowing in the first place.

Another way to suppress the effect of side lobes is to select a long section of the sound so that the side lobes are very narrow and will vanish for the most part, even when still close to the spectral components of the signal. Of course, this option is only valid for signals of which it is known that the components are steady (machines, musical instruments with sustained tones, etc.). For speech, for example, we mostly want to analyze short time parts and then windowing is inevitable.

Example 2.

The precision of the numbers in the computer is much higher than the 16-bit precision of the sound format mostly used. Almost all programs for signal analysis use this high precision for manipulation and calculation of the sound samples. When, for example, a

sound is generated in these programs the result contains sample values with this high precision. When the resulting sound is saved as a sound file with 16-bit samples, however, this high precision is lost.

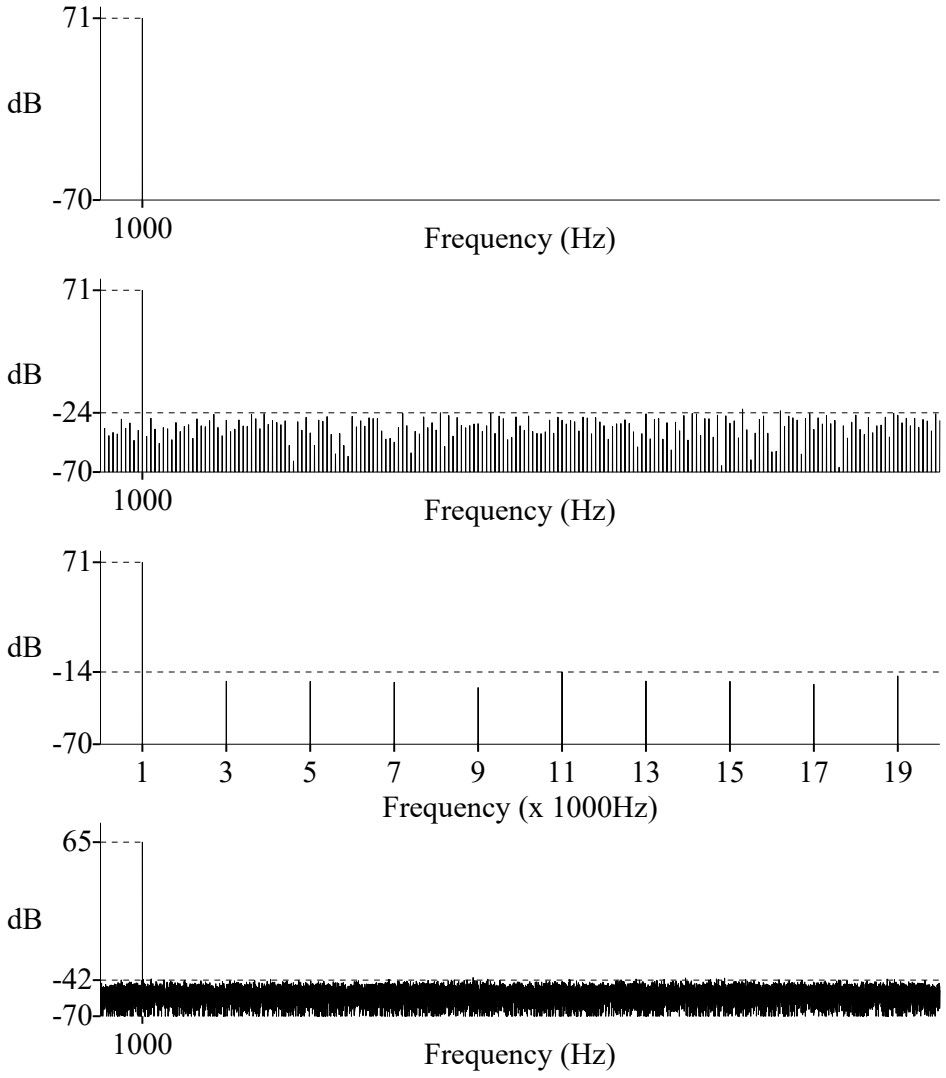


Fig. 17.10. Limited quantization precision of 16-bit sampling. From top to bottom:
- Spectrum of 1second sine wave of 1000 Hz with amplitude 0.1, sampled with 44100Hz, generated in computer.
- Spectrum after saving in 16-bit samples and reading back the sound again.
- Spectrum after same procedure, now with 50000 Hz sampling.
- Same as previous one, now a priori windowed with a Hann window.

Fig. 17.10 displays the spectrum of a signal generated in the computer with Praat (a 1000 Hz sine wave for 1 second with amplitude 0.1 and a sampling frequency of 44100 Hz). The visual dB range is extended to 150 dB and even on this extended scale its DFT shows a nice single component at 1000 Hz. (The choice of the frequencies and time ensures that no spectral leakage occurs.) After saving this sound as a standard format

sound file (which uses 16-bit precision), and reading it back to the program again, its spectrum shows a sample noise level with its peaks about 95 dB lower than the signal peak, spread over the whole frequency range (22050Hz, being the Nyquist frequency). The only cause of this can be the limited precision of the 16-bit sample numbers. Now the same is done for a sampling frequency of 50000 Hz. The choice of this sampling frequency, which is a multiple of the signal frequency, causes the samples within *each period* of the 1000 Hz sine wave to be placed on identical time positions. Fig. 17.10 shows the result: instead of the white noise of the former example, all odd multiples of 1000 Hz occur, of which the peaks are about 85 dB lower than the signal peak. Here the sampling noise results in a *waveform distortion* (apparently the sampling inaccuracy is symmetrical with respect to the zero line so that only odd harmonics occur, see section 6 about symmetry).

The rms values of both sample noise types are equal, as the noise level is determined by the 16-bit precision and the signal level, but the spectral peaks of the odd harmonics are about 10 dB higher than those of the other -white- noise.

Obviously, the white type of sample noise is preferable, so that in this respect it seems advantageous to apply a sampling frequency that is *not a multiple* of the signal frequency. Mostly, however, one must adhere to a standard sampling frequency for compatibility. A better strategy is to *window* the signal *before saving it* as an audio signal with some window other than the rectangular one. The bottom part of fig. 17.10 shows the spectrum of the same signal, now windowed with a Hann window. The sample noise looks much more like white noise and the odd multiples of the signal frequency are gone. Moreover, the sample noise peaks are about 107 dB lower than the signal's peak level. Thus, here is another reason to window the signal selections. It is important to bear in mind, however, the necessity of *a priori windowing* of the signal, which is possible because of the generation of the signal within the computer. If the signal were saved unwindowed, then read-in again and windowed, it would give no improvement. This is caused by the fact that the *a posteriori* windowing leaves the waveform shapes unaltered: it only multiplies by factors. All waveform sampling inaccuracies remain at exactly the same positions in the waveform so that in our 50000 Hz sampling example the waveform distortion remains the same. This means that windowing of a selection of a sound already recorded will NOT improve the S/N ratio. (But windowing remains necessary for the other reasons explained previously.)

Example 3.

When the amplitude of the sound generated in the computer is low, the S/N ratio of the sound, saved as a sound format file, lowers too. In the last example the amplitude is 0.1. In Praat the value 1 means **full scale** (maximum amplitude) so that only one tenth of the total number of steps is used. The magnitude of the steps remains the same which causes a decrease of the S/N ratio with 20 dB. In a 16 bit format the S/N ratio then becomes 78 dB instead of 98 dB. In general, one must be aware of the fact that the

absolute quantization error amplitude is constant and therefore independent of the signal amplitude. In part B of the book there is more about the practical aspects of the S/N ratio.

18. Digital filters

Once we have our sounds in digital form in the computer, we will be able to perform all kinds of analyses and manipulations in a much more flexible way compared to analog processing. For example, Fourier transforms and correlations would be almost impossible in analog electronics. In former days, approximations of Fourier transforms were done by *narrow band filtering*. The development of analog filtering in its long history has reached a high level and can be done in sophisticated ways. Nevertheless, the problems of stability when steep slopes or very narrow bands of the filter functions are involved, remain.

In digital form the signals can be analyzed and manipulated using a vast number of methods and with stability factors that at the moment are some thousands of times better than that of analog electronics because the only source of instability is the computer's number precision, which in principle can be raised at will[1]. Additionally, the filter properties that are possible in digital form greatly surpass the analog filter properties. Also, all (linear) signal processing can be done in the time domain as well as in the frequency domain.

We have already seen the convolution mechanism in the time domain using small steps in section 14 (DEMOs 14.1 and 14.2). Theoretically we need infinitesimally small steps. In our computer the smallest possible steps are equal to the sampling interval, as there is no information between the sample positions. The convolution is carried out *sample by sample*. Obviously the same applies to autocorrelation and cross-correlation: the mechanisms are similar.

Filtering in the time domain can be done, as explained in section 14 and before, by convolution of the signal with the filter's impulse response. In digital filter jargon this (sampled) impulse response is called the **filter kernel**.

The relative simplicity of performing the digital signal manipulations in the time domain offers a big advantage of digital filters over their analog counterparts. In addition, the achievements of digital filters can reach high levels. For example, look at the performance of the low-pass filter and high-pass filter of fig. 18.1.

The low-pass filter is realized by defining a sinc function as impulse response, which was already mentioned in section 6 about even functions. As explained in appendix II.3, for even functions the time function and its Fourier transform function are interchangeable. So, the spectrum of a rectangular pulse being a sinc function, a sinc function in the time domain causes a rectangular function in the frequency domain which is an *ideal low-pass filter*.

---

[1] As later to be explained, so-called *recursive* digital filters, however, need a very high number precision as these filters are based on feedback of former values which multiplies the existing number inaccuracy.

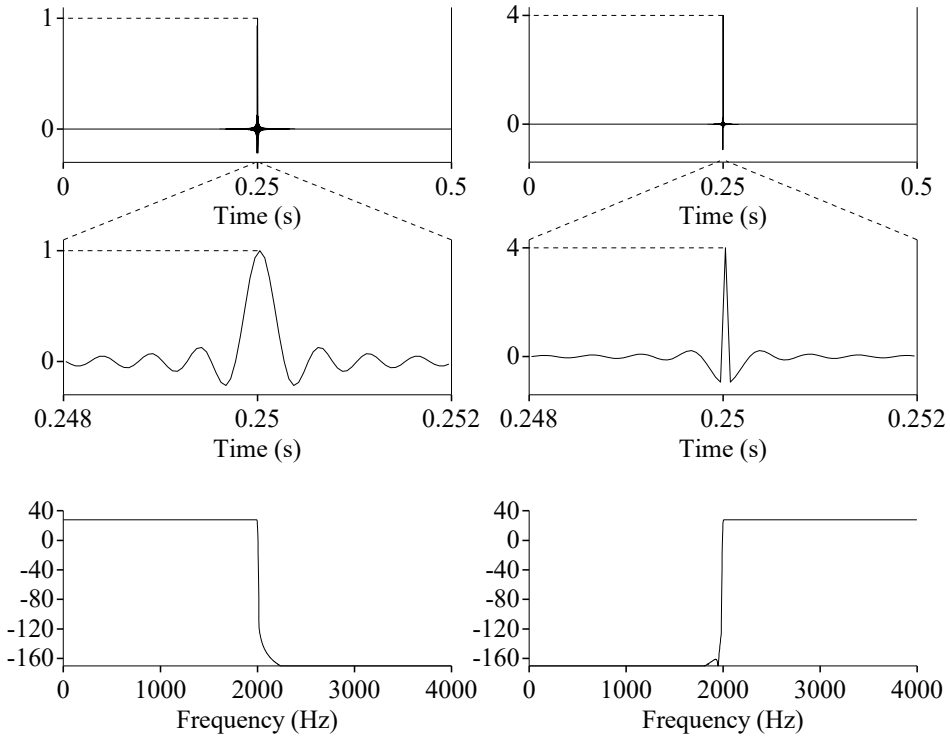


Fig. 18.1. Very high performance low-pass (left) and high-pass (right) digital filters realized by applying windowed sinc impulse responses. The high-pass filter combines an all-pass filter (one sample of value 4) with an inverted low-pass filter.

Because of the fact that the side lobes of a sinc function spread out in time from -∞ to +∞ it has to be truncated in practice. This example uses an impulse response of 0.5 s duration and is overall windowed with a (tampered) Gaussian window so that the side lobes of the sinc function gradually reach zero amplitude at the left and right edges.

You can see that the ratio of *pass band* and *stop band* levels is enormous: about 200 dB which is equivalent to 10^{10} to 1. A ratio like that is completely unnecessary in practice: It outranges the human ear dynamics with a factor 10000 and no electronic audio equipment could process a signal ratio like that. Needless to say that, in practice, a filter like this is not realizable by analog electronics (which, however, is no problem because nobody needs it).

The high-pass filter in the figure uses a modified sinc function in its kernel. In fact, it is a combination of an *all-pass filter* and an *inverted* low-pass filter. Fig. 18.2 shows the working principle.

An all-pass filter has an impulse response which is a delta pulse because its spectrum is formed by a straight horizontal line. This is not surprising if you look at the time domain as well: if you shift a delta pulse through a signal, its convolution is equal to

the signal itself (remember the convolution mechanism with a short pulse in section 14). In the sampled signals a delta pulse means that the kernel consists of one sample having the value 1 and all other samples having the value 0. The combination of the two filters can be applied within one filter kernel. In the high-pass filter impulse response of fig. 18.1 you see the addition of a single value in the center added to the inversed sinc function. (For the *stop band* of the high-pass filter it is necessary that the all-pass filter and the inverted low-pass filter cancel each other, which requires the scaling factor of 4 in this case.) For exact canceling of the outputs of the two filters in the stop band it is crucial that their phase difference is zero. For analog filters, this requirement would be almost impossible to fulfill; for the digital filter this is quite trivial: there is no phase shift, only a constant response time for all frequencies.

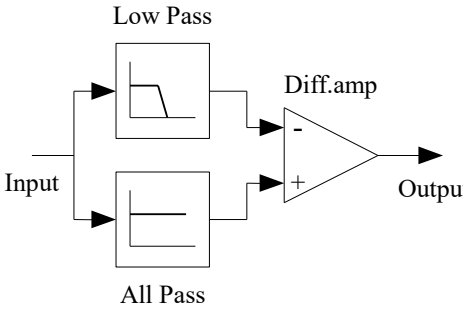


Fig. 18.2. High-pass filtering by (in-phase) subtraction of low-pass filter output from all-pass filter output.

Of course, filters with slopes like these will have very long response times: here the impulse response used has a duration of 0.5 seconds. That is always the price to pay: a great resolution in the frequency domain means a poor resolution in the time domain and vice versa.

Because of the windowing of the impulse response this kind of filters is called **windowed sinc** filters.

The time domain way of filtering (i.e. the convolution of the input with the filter's impulse response) lends itself perfectly for the **real time** filtering of long signals. Real time means here that the filter produces its output at the same speed as the samples of the input signal enter the system. Of course, there will be a delay caused by the length of the impulse response but the computer is fast enough to calculate the result of an incoming sample of the input signal before the next sample in time occurs so that the output rate (samples per second) can be equal to the input rate. Generally, the filter's impulse response duration for many filters used in practice is not more than a few tens of milliseconds so that it seems as if the variations of the result follow the input signal immediately.

The filters described so far compute the output from a number of input sample values. Each new sample at the input shifts the calculation of the new sample at the output from a set of input samples that is shifted one sample further: the oldest sample is abandoned and the newly entered sample is included in the calculation. Sometimes partial results of calculation on the intermediate samples can be used for computation of the next output sample to limit the calculation steps in the filter algorithm. Let us take a simple

example of such a filter that is very suitable for application of this idea: the **moving average** filter (see fig. 18.3).

Each sample of the output of this filter is computed by taking the mean of the current input sample and a number of preceding input samples. Averaging over 5 samples, for example, is defined by the formula:

$$y_n = \frac{1}{5}\left(x_n + x_{n-1} + x_{n-2} + x_{n-3} + x_{n-4}\right) \tag{18.1}$$

where y is the input, x the output and n the rank number of the incoming sample. If we define m as the number of samples to be averaged the moving average filter can be generally defined as:

$$y_n = \frac{1}{m}\sum_{k=0}^{m-1} x_{n-k} \tag{18.2}$$

From fig. 18.3 you will recognize this filtering as a convolution of the input signal with a rectangular window. Therefore, the filter spectrum is the sinc function. So, in some sense it is the opposite of the windowed sinc low-pass filter. The frequency resolution is poor but the time resolution is optimal. Looking at formula 18.1 we can now write the next sample rank as n+1:

$$y_{n+1} = \frac{1}{5}\left(x_{n+1} + x_n + x_{n-1} + x_{n-2} + x_{n-3}\right) \tag{18.3}$$

Compared to formula 18.1 you can see that 4 out of 5 preceding input sample values are re-used. One value (the oldest) is abandoned and the new input sample is added.

Instead of adding up all 5 samples again, we can use the preceding *output* value y_n if we subtract this old sample value x_{n-4} and add the new one x_{n+1}:

$$y_{n+1} = \frac{1}{5}\left(x_{n+1} + 5y_n - x_{n-4}\right) = y_n + \frac{1}{5}\left(x_{n+1} - x_{n-4}\right) \tag{18.4}$$

For *any* number of samples to be averaged the general formula for the filter kernel remains equally simple:

$$y_n = y_{n-1} + \frac{1}{m}\left(x_n - x_{n-m}\right) \tag{18.5}$$

where m is the number of samples averaged. If, like in this filter example, besides input samples, one or more previous *output* samples are used for estimation of new output samples, the filter is called **recursive**.

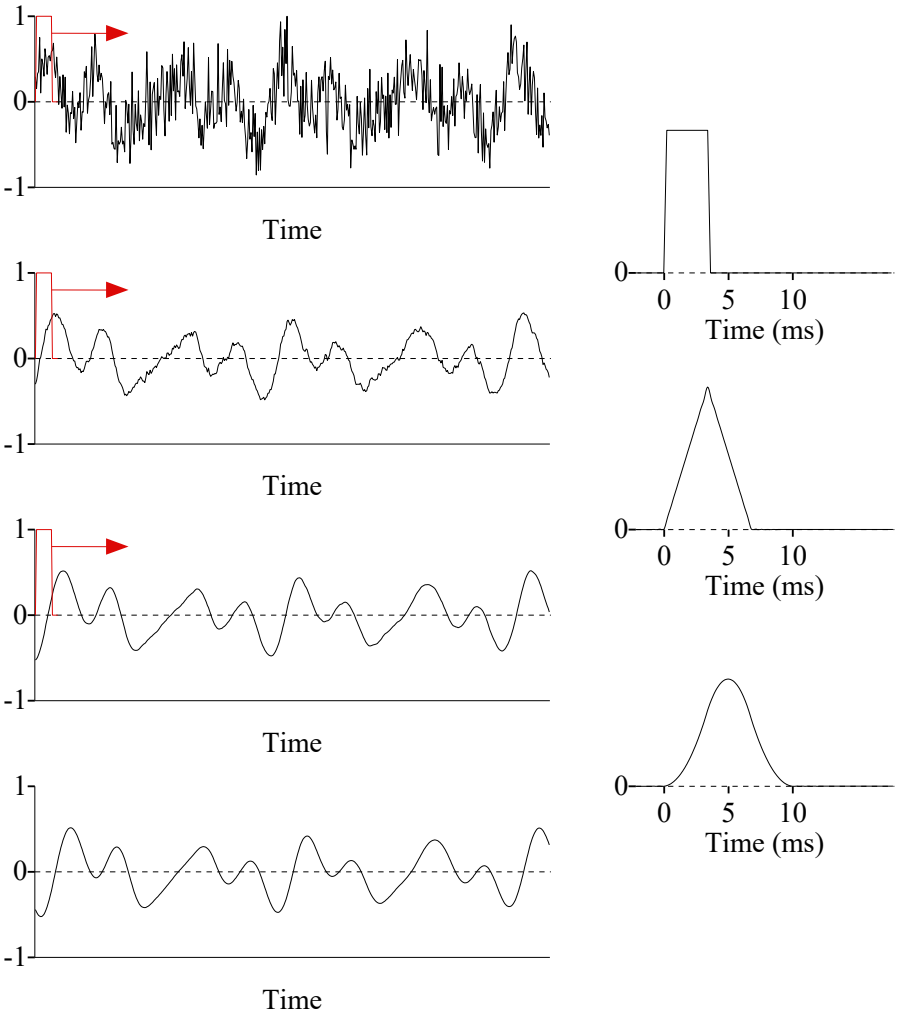


Fig. 18.3. Moving average filter. From top to bottom: a noisy chord sound is low-pass filtered one, two and three times respectively, by averaging 15 samples (0.34 ms) within the moving block. Right part: the resulting impulse responses for one, two and three filter runs.

In fig. 18.3 the moving average filter is applied to a noisy sound (the chord sound from section 3 with added white noise). The delay of the filtered sound is quite low thanks to the short impulse response of the filter (here only 0.34 ms), while the noise suppression is quite reasonable. When m is the number of samples averaged, the high frequency noise components are suppressed by the factor $\sqrt{m}$.

The figure also shows the effect of filtering a *second* time and a *third* time. Each run suppresses the noise by the same factor. When the number of filtering repetitions increases the resulting impulse response approximates a Gaussian function[1] and the

[1] It can be proven mathematically that the sum (or mean) of a sufficient number of random equally distributed values approaches a “normal distribution” or Gauss function.

spectral lobes are gradually suppressed. Naturally, the resulting delay becomes greater (each added repetition multiplies the previous delay by √2). Multiple filtering with this filter is sometimes preferred over a single convolution with a Gaussian approximation or other window function thanks to the low "computation cost" which is *independent of the kernel length*. This makes this filter suitable for fast signal processing (like graphical signal manipulations).

In the reality, a filter will only be able to respond from the moment the input signal starts. In other words, the filter is *causal*. Our moving average filter gives the first proper output value after *m* input samples have occurred, which causes a delay equal to the impulse response time. The input signal present in the computer's memory provides the opportunity to shift the output signal 'back in time' relative to the input signal. In this filter case, the averaging occurs over *m*/2 preceding samples and *m*/2 next samples. Therefore, digital filter kernels are usually centralized so that there is virtually no time shift between corresponding input and output samples. Of course, the reaction time remains the same; it is only distributed symmetrically around the time positions of the input samples. The requirement for this symmetry is that the number of kernel samples must be odd in order for it to have a central sample. Our formula for the moving average filter for a centralized kernel is thus slightly modified:

$$y_n = y_{n-1} + \frac{1}{m}\left(x_{n+p} - x_{n-q}\right) \tag{18.6}$$

where $p = (m\text{-}1)/2$, $q = (m\text{+}1)/2$ and m is an odd number.

As mentioned already in section 8, a symmetrical impulse response around $t = 0$ produces only cosine components so that it is a *zero-phase filter*. So, the filter kernel according to formula 18.6 is a zero-phase filter. When the kernel is symmetrical but the center is shifted to the left there is a constant delay which implies that the filter is a *linear phase filter*. (A constant delay means a phase which increases linearly, proportional to increasing frequency.) When the *form* of the filter kernel is asymmetrical around its mid position it is a *non-linear phase filter*. Obviously linear phase filters and non-linear phase filters both have sine and cosine components in their spectra.

When we omit the subtraction of older samples, we simply add all sample values:

$$y_n = x_n + y_{n-1} \tag{18.7}$$

Because there is now no need to refer to older x values we can apply this cumulative addition of sample values to an existing array of samples so that it is modified "on the spot":

$$y_n = y_n + y_{n-1} \tag{18.8}$$

(It is customary that the y variables refer to the output sample array and the x variables refer to the input array. In this case the input values are copied into the output array

prior to the processing of the contents.) Of course, this formula should not be regarded as a pure algebraic relation (in that case the conclusion would be that $y_{n-1} = 0$) but it means that in each step the value of y_{n-1} is *added to* the *former* value of y_n. To avoid unlimited increase of the sum value (the more samples in the signal, the higher the sum may become) we have to divide the sum values by the sample frequency to make the function independent of it. This makes it equivalent to the sample-by-sample *integration* of the *y* function. (Integration being an estimation of waveform *areas*, the computed values depend on the time step length, i.e. the sample frequency, and have to be multiplied by the sample interval which is the same as dividing by the sample frequency.) So, the *new* sample values in formula 18.8 must be divided by the sample frequency:

$$y_n = \frac{y_n}{f_S} + y_{n-1} \tag{18.9}$$

which transforms the array with *y* values into its integral function.

Using a similar reasoning the formula for *differentiation* of values in an array with samples turns out to be:

$$y_n = (y_{n+1} - y_n) \cdot f_S \tag{18.10}$$

The integration and differentiation methods described here are very fast as they work within the arrays that contain the input values. The original data are lost so that it is necessary to copy the data array before starting the integration or differentiation process. This method of signal manipulation within the data array is called **in-line processing**. In general, however, digital filters will read values from the input array and place the result in the output array.

The general formula for a recursive filter can thus be stated as:

$$y_n = a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2} + ... + b_1 y_{n-1} + b_2 y_{n-2} + ... \tag{18.11}$$

By convention the coefficients of the inputs are represented by *a*'s and the coefficients of the outputs by *b*'s. In this format the formula for the integrator, for example, becomes:

$$y_n = a_0 x_n + b_1 y_{n-1} \tag{18.12}$$

where $a_0 = 1/f_s$ and $b_1 = 1$.

Let us now have a look at the in-line integrating filter $y_n = b_1 \cdot y_{n-1}$ and activate it with a unit pulse (so, $y_0 = 1$ and all next samples are zero). If we change b_1 to a value lower than 1 we only use a *part* of the older *y* value. It will be clear that each next output

sample value will become the value of its predecessor multiplied with b_1 so that an *exponential relation* exists between sample number and value:

$$y_n = b_1^m \cdot y_{n-m} \tag{18.13}$$

For $b_1 = 0.9$ the result is shown in fig. 18.4. Because *parts* of former outputs are used for computing the new output, this impulse response is indeed infinitely long. Hence this kind of filters is called **Infinite Impulse Response** (IIR) filters, as opposed to the windowed sinc and the moving average filter mentioned before, which are **Finite Impulse Response** (FIR) filters. In fact, recursive filters are usually IIR filters because the feedback of parts of former output values to the input causes an endless process. The recursive moving average filter is an exception, the only reason being that *exact* values of former samples are subtracted.

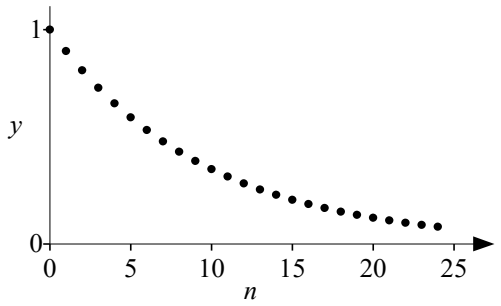


Fig. 18.4. Exponential relation between sample values and rank numbers in formula $y_n = 0.9 y_{n-1}$ after a unit pulse is applied $(y_0 = 1)$

If $b_1 < 1$ the function will decrease exponentially and approach zero. If $b_1 > 1$ the function will "explode" to infinity. We see that a value of b_1 between 0 and 1 will produce an *exponential decaying* function as impulse response of the filter. Its spectrum we have already seen in section 13 about windows (see fig. 13.4). As a filter, this works as the simplest possible low-pass filter. The *order* of the filter is 1, which refers to the number of sample steps back in history (m = 1 in this case).

An example of an order-two digital filter is the recursive version of our familiar resonator filter. If we define the resonance frequency as f_r, its bandwidth as B, and the sample frequency of the kernel as f_s, the filter can be defined by the following formula:

$$y_n = x_n + b_1 y_{n-1} + b_2 y_{n-2} \tag{18.14}$$

where $b_1 = 2e^{-\pi B / f_S} \cos(2\pi f_R / f_S)$ and $b_2 = -e^{-2\pi B / f_S}$. For a resonator filter with 1000 Hz resonance frequency and 50 Hz bandwidth, for example, b_1 should be 1.9727 and b_2 should be -0.9929 when the sample frequency is 44100 Hz. (The reason that the values for B and f_r are expressed as *parts of the sample frequency* is that the formulas like 18.14 refer to *sample intervals* instead of seconds.) Because the constants b_1 and b_2 can be estimated in advance, the filtering itself requires only two multiplications and two subtractions per sample which means very low *computation costs*. Compared to a non-recursive filter which convolves the signal with a damped sine wave kernel (which has to be truncated in practice) this recursive version is much more efficient. IIR filters generally have high execution speeds because of their low computation costs.

How can we estimate the values of the necessary coefficients like b_1 and b_2 from a desired filter function? For this purpose, the tailor-made **z-transform** can be used. Unfortunately, the mathematics necessary for understanding this transform is not very simple. It requires at least calculation with complex functions. In appendix IV the z-transform is explained in some detail. Its global principle is based on the following way of reasoning.

As you learned when the resonator was explained, its spectrum shows a peak at the resonance frequency. When the impulse response, which is a damped sine wave, has a low damping α, the peak in the spectrum becomes high and narrow, and a high damping causes a low and broad peak. Now the clever thing is that the spectrum of any filter can be seen as having been assembled by a set of resonators, each contributing to a peak in its spectral function. So, the spectral filter components are *damped sine components* instead of continuous sine or cosine components. The filter's impulse response is then a combination of damped sine waves with various resonance frequencies and damping factors, called the *poles*. In the same way a Fourier transform component is defined by frequency and phase (or sine and cosine values), the z-transform components are defined by frequency and *damping*. This means that z-transform components have starting points in time as opposed to Fourier transform components, which are continuous. Mathematically, one spectral peak consists of two poles in a *complex pair* which is explained in appendices III and IV.

An example of a 10-pole low-pass filter is shown in fig. 18.5. It is a so-called *Chebyshev filter*, as it is based on the work of Chebyshev, a Russian mathematician (1821-1894). The spectra of the 5 individual peaks are displayed as well. The multiplication of these 5 specific separate pole pairs results in the straight horizontal line in the pass band[1]. Compared to the windowed sinc low-pass filter from the beginning of this section this filter does not come near its performance as regards the frequency response. The slope of this filter is 'only' about 120 dB per octave. The delay, however, is about 1/25 of the windowed sinc's.

This example shows also that a higher order filter function (order 10 here) can be broken down to lower order sections. The z-transform mathematics available is able to define the relation between the filter transfer function and all recursive a and b coefficients and can be seen as a great tool for dealing with the complexity of this digital manipulation.

In addition to poles, *zeros* can also be applied in the z-transform. Together with the poles, they can be used to design any practical filter, including notch filters with highly suppressed outputs at specified frequencies. Explanation of further details about the

[1] In fact, in this example a very small *ripple* of 0.5% of the amplitude in the pass band is allowed for, which is not visible in the picture on this amplitude scale. It is very well possible to make this ripple zero, at the cost of steepness of the curve. Then the filter is a Butterworth filter, named after an English engineer who published the math in 1930.

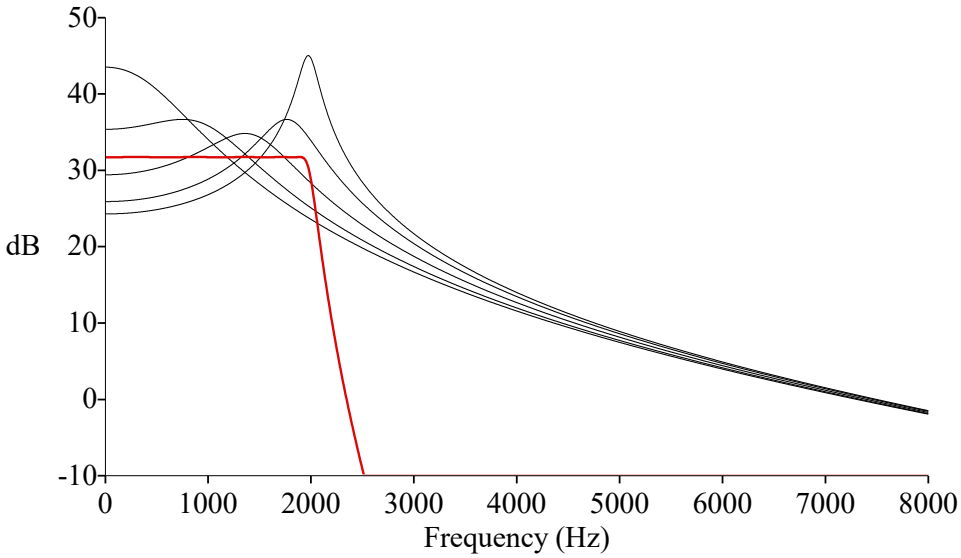


Fig. 18.5. Low-pass 10-pole Chebyshev filter (red line) as a result of cascading five second order filters, having one pole-pair each.

mathematics will need complex representation of signals and transfer functions, which is explained in the appendices II and III.

In the beginning of this section, the zero-phase possibility by convolution with a symmetrical filter kernel was explained. Theoretically, in the case of IIR filters, the impulse response is infinitely long, and in practice much longer than the IIR filter kernel. Nevertheless, there is a possibility to realize zero phase filtering with IIR filters. If the sequence of all rank numbers of the samples is reversed, which means playing it backward in time, the phase is opposite in sign compared to the non-reversed filter. A cascade of these two filters, therefore, cancels all phases resulting in zero phase. The magnitude is the square of the magnitude of one filter. Because the signal (or chunks of a longer signal) is held in the memory of the computer, the two filters can be run one after the other. Finally, the square root of the result can be taken to scale it (or their dB values halved).

Another digital filtering subject worth mentioning is the possibility of **custom designing** a filter by defining the desired frequency response directly in the frequency domain. Any practical filter function defined in this way has its own impulse response: it is the inverse Fourier transform of the filter function. We can use this impulse response as a digital filter kernel for convolution with the signal which we want to filter. So, when we have, as an example, a bizarre filter function like in fig. 18.6 we take its

inverse Fourier transform and use it as a digital filter kernel. There is a complication, however. Because of the fact that the filter function shown refers *to the magnitude only*, the phase is not defined. The inverse Fourier transform then regards the function as a *zero-phase* filter with only cosine components. The inverse transformed time function is therefore *symmetrical around sample zero* as you can see in fig. 18.6. (The 'negative' time samples wrap around.) To use it as a manageable filter kernel we should shift all samples so that this zeroth sample is placed in the center. In programming terms: it is *rotated to the right or left*. As you now know, the only effect is a filter output delay of half the kernel length.

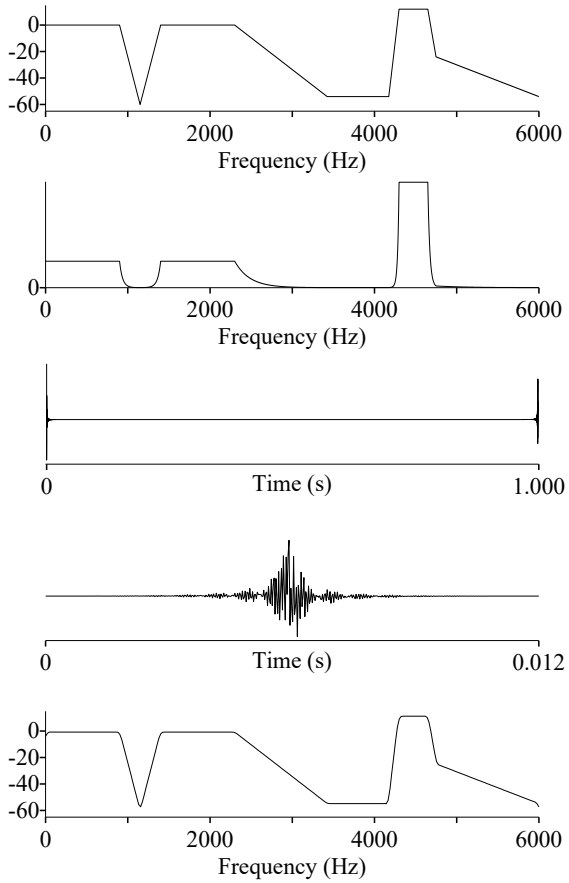


Fig. 18.6. Custom designed filter. From top to bottom: desired log spectrum; linear version; its inverse Fourier transform; its center shifted (rotated), zoomed in, and Hann-windowed; resulting spectrum as a check.

The second complication is that the digital representation of the filter function in the computer has to be time-limited so it cannot be a continuous function: the frequency step (the frequency **bin**) has to be defined. If we set it to 1 Hz, for example, the reverse transformed time function has a duration of 1 s. (It sets also the theoretical lowest frequency limit to 1 Hz.) The higher the resolution in the frequency function the longer the kernel length. In practice, the length can be reduced by truncation and windowing, just as in the case of the windowed sinc filter described above. This is done in fig. 18.6. The kernel length is reduced from 1 s to 12 ms and windowed with a Hann window. The resulting filter function, checked by taking its Fourier transform, is a fair approximation of the original defined filter magnitude function. A further shortening of the kernel length will be at the expense of the approximation, due to *spurious ripple* (unwanted ripple) and greater smoothing of the resulting function.

This custom filtering leads to the final filtering method we will discuss. A signal could also be filtered by directly *multiplying its spectrum* with a desired filter magnitude function. Of course, this can only be done for sounds that are stored completely in the

computer prior to the filtering: the spectrum of the complete sound has to be available before the filtering can be performed. The multiplication with a magnitude function is equivalent to zero phase filtering so that the filter responds to, for example, local parts of the sound 'before they occur'.

To complete this section, we will go into a subject that might give rise to some confusion. When we activated the digital filters, we applied a 'unit pulse' which was realised by making one sample 1 and all next samples 0. In fact, this could be seen as a pulse with a length equal to T (The sampling interval). The spectrum of a pulse like that is a sinc function which causes a spectrum which is not flat from 0 to the Nyquist frequency. Should it be necessary to compensate for the sinc's roll-off? The answer is "no". To explain this, we can use a similar way of reasoning as that for Shannon's reconstruction theorem when we exchange the time and frequency domains. To convert all discrete frequency points of a horizontal spectrum to a continuous spectrum we apply a sinc function at each frequency point and add it all (see for this construction fig. 17.4 of section 17). The result cannot be different from a straight horizontal line which forms a continuous spectrum. This is the spectrum of a pure Dirac pulse! Its time function, therefore, has zero length. In fact, the values between the samples can't, and don't, exist.

This is the end of part A. You should now have gathered sufficient basic knowledge to understand the why and how of the practical sections of part B.

Part B. Practical considerations

19. The principle of speech production

Many things we learned from the theory in part A of the book can be applied to the analysis of sounds that are produced by the human voice. Even if your focus of interest does not lie primarily in the field of speech, a lot of practical information about various sound analyses can be explained on the basis of speech sounds.

In a nutshell, speech sounds are generated by pushing air from the lungs through the *vocal folds*, also called *vocal cords*, which will start to vibrate. The sound generated by the vocal folds activates the cavities of the throat and mouth (together forming the *vocal tract*) which act as resonators. Therefore, the sound departing from the lips can be seen as the source sound filtered by all vocal tract cavities (see fig. 19.1). By moving the tongue and jaw into various positions the shape of these cavities can be altered so that the filtering of the sound coming from the source can be varied. In this way the differences between the *vowels* are realized. For example, the sounds of the words "feed" and "fed" only differ in the mid parts of their waveforms where, in these examples, the vowels are positioned. The *soft palate* (or *velum*) can be moved to open or close the entrance to the nasal cavity which results in *nasal* or *oral* sounds respectively.

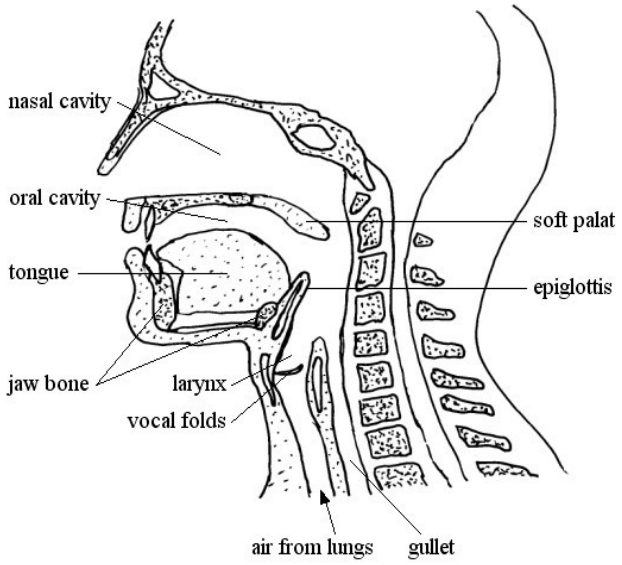


Fig. 19.1. View of the human speech organs.

The vibration of the vocal folds can be explained by looking at the following sequence of events: starting from the vocal folds held close by the controlling muscles, the air pressure from the lungs forces the vocal folds to part from each other so that air passes through. The air pressure between the vocal folds then decreases (caused by the so-called *Bernoulli effect*) which causes the vocal folds to close again, due to their elastic properties. The air pressure below the vocal folds is then built up again by the sustained air flow from the lungs and the cycle starts again. The result is a periodical stream of separate air packets which form the sound source of (*voiced*) speech, also called *glottal pulses*. The frequency of the resulting sound depends mainly on the length and mass of the vocal folds and the different forces applied by this set of muscles. A typical waveform of the air flow as a

function of time can be seen in fig. 19.2, which is made by using a commonly accepted artificial glottal pulse generation model.

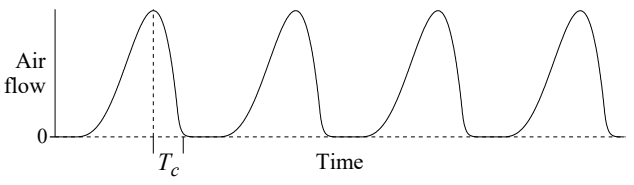


Fig. 19.2. Glottal pulses of human voice.

In the case of *whispered* vowels, the vocal folds do not vibrate; instead, they are opened a little, causing air turbulence, thus producing a noise sound. Additional noisy speech sounds can be made by adjusting gaps between teeth and tongue tip or making other narrow passages for the air in the vocal tract using the *epiglottis* or the tongue. All these noisy speech sounds are independent of the vocal folds but can also be combined with vibration of the vocal folds (*voiced consonants*). In this section we will focus only on voiced sounds.

A typical spectrum of a periodical glottal pulse signal is depicted in fig. 19.3. In this example the roll-off is about 16 dB/oct, a value mainly defined by the maximum speed of change that occurs in the waveform curve, which depends on the time it takes to close the vocal folds: the *collision time* (T_c in fig. 19.2). The shorter the collision time, the weaker the roll-off and thus the stronger the high harmonics. The longer the collision time, the greater the roll-off. The collision time is roughly proportional to the F_0 period. In the example T_c is about 20% of the F_0 period. Commonly it is assumed that the roll-off of the average human voice source is about 12 dB/oct. In practice, however, great variations of this ratio may exist from speaker to speaker.

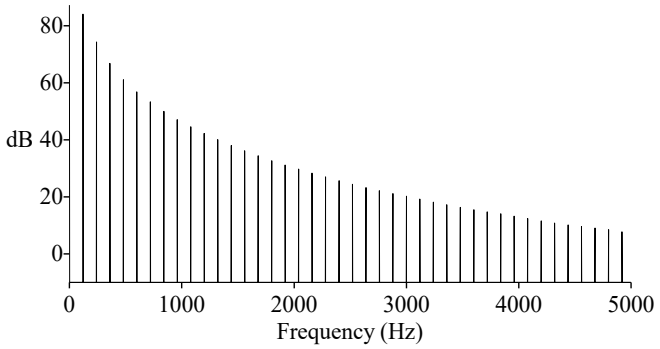


Fig. 19.3. Spectrum of glottal pulses (F_0 = 120 Hz).

In this case, the DC level at zero frequency (which is equal to the mean amplitude value, as mentioned in section 6) is about 88 dB. This relatively high value can be expected because the complete waveform of the glottal pulse exists above the zero level, here resulting in a mean value of about 1/3 of the peak value.

The sound waves passing the lips are propagated in the surrounding air. The transition of the sound from the enclosure of the vocal tract area to the unrestricted surrounding air can be regarded as a high-pass filter, called the **radiation filter**, having a slope of about + 6 dB/oct. The explanation of this phenomenon is rather complicated. With a bit of a stretch of the imagination one can regard the mouth opening in the skull as a small hole in a baffle (sounding board), where the skull is seen as a baffle with a relatively large surface. The shorter the wave length, the better the surrounding air is activated. The result of this phenomenon is a high-pass filter. You can find a more extensive explanation in Rabiner and Schafer, p.71...74 [11]. Combining the nominal slope of - 12 dB/oct of the glottal source with the +6dB of the radiation filter results in an overall slope of - 6dB/oct.

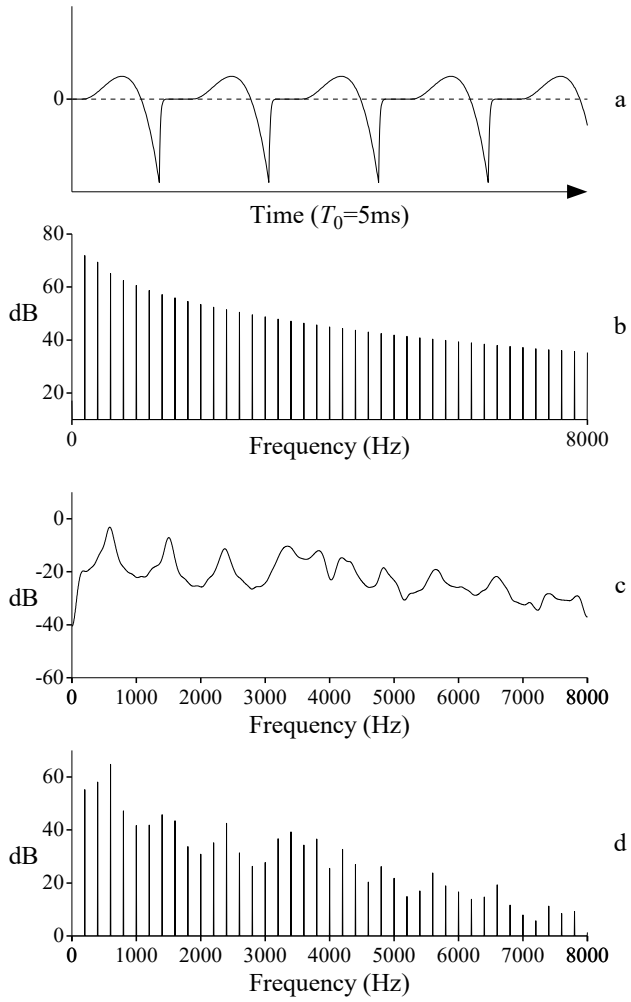


Fig. 19.4. a: derivative of glottal pulses; b: its spectrum; c: vocal tract filter function of natural /ɛ/ sound; d: spectrum of resulting speech signal.

Thus, we can conclude that a voiced speech sound signal entering the microphone can be regarded as the result of a source *substitute*, having a spectrum with a slope of - 6dB/oct, activating the vocal tract's filter. For this reason, the source waveform is often presented as the *derivative* of the glottal pulse (see fig. 19.4a), because the spectral effect of the derivative is that it alters the spectral slope with + 6 dB/oct (see Appendix II).[1]

As a consequence, the differentiation of the glottal pulse causes the DC component being zero (starting from any frequency in the direction to zero, each octave attenuates the spectrum level with 6 dB; the number of octaves to zero is infinite so that the level at zero frequency is -∞ dB).

[1] In Praat, the default settings of artificially generated glottal pulse substitutes result in spectral slopes of about -11 dB/oct instead of -6 dB/oct, caused by smoothing of a discontinuity in the glottal function derivative.

The *shape* of the glottal pulse is of limited importance. In Praat, many parameters of the generated artificial glottal pulse can be adjusted, like open/close time ratio, order of the time function formula, collision time, etc. Many different varieties of the glottal pulse waveform settings produce more or less the same spectrum (apart from pathological glottal pulses with can have very irregular spectra). The main parameter of the glottal pulse is the collision time, which has some influence on the slope of the spectrum.

An example of the vocal tract filter properties for the vowel sound "/ɛ/" within a word like "fed" is displayed by fig. 19.4c. This vocal tract filter activated by the derivative of the source waveform results in the spectrum as shown in fig. 19.4d which, as explained in section 8, equals the multiplication of the spectra of fig. 19.4b and 19.4c.

The peaks of the vocal tract's filter function are called the **formants**. The frequency positions and relative amplitudes of these formant peaks will vary when different vowel sounds are produced, depending on the positions of the jaw, the tongue, the soft palate, etc.

As a model for the vocal tract filter, the vocal tract area is often regarded as a combination of two or more resonant tubes. See fig. 19.5a for an example of a dual tube model of an [a]-like sound. The output of the left section is the input for the right section so that the dual tube model can be regarded as a cascade of two filters. Both tube sections in this example can be regarded as resonant tubes that are open at one end. On the left side of the picture you see that the back tube is closed, this is the position of the source (glottis). The front tube (which ends at the mouth) is much wider and can be regarded as (almost) closed on the left side by the narrow back tube. The tubes resonate in such a way that there fit only odd numbers of *quarter wavelengths* within the tube length. For a single tube then, if l is the length of the tube and c is the propagation speed of the sound, the resonance frequencies are defined by:

$$f_n = \frac{(2n-1)c}{4l} \tag{19.1}$$

So even with only two tube sections there will be a great number of formant peaks. To complicate matters a bit further, a *constriction* can be made in the vocal tract so that the back part no longer has an open end (see fig. 19.5b). This means that it must be regarded as a **Helmholtz resonator**, named after the German physicist Herman von Helmholtz (1821-1894), who investigated, among other things, frequency components of musical instruments. Now for the Helmholtz resonator the resonance frequency is defined by the (simplified) formula:

$$f = \frac{c}{2\pi}\sqrt{\frac{S}{VL}} \tag{19.2}$$

where c is the propagation speed of sound, S is the cross-section area of the constriction, V is the volume of the body and L is the length of the constriction. Whereas the lowest

frequency of the open-ended tube is determined by its length, the resonance frequency of the Helmholtz resonator does not have this limitation by length (theoretically, if S decreases to zero, the resonance frequency becomes zero too). This is the reason why the lowest formant for an [I] sound as in *deep* is so low that it may at times be not much higher than the fundamental frequency.

Because the back part of the vocal tract, in the case of a constriction, still has the shape of a tube (see fig. 19.5b), the air within it will *also* resonate like a tube and will produce resonance frequencies with wavelengths that depend on the tube length. In the case shown in fig. 19.5b, however, the tube must be regarded as (almost) *closed at both ends* (not taking into account the small opening of the place of constriction). For this type of tube resonance there is an integer number of *half wavelengths* so that the frequencies are:

$$f_n = \frac{nc}{2l} \tag{19.3}$$

As for the bandwidths of the resonance peaks, we must bear in mind that they depend on the type of material used. In case of physical tube models, the material used is often *plexiglass* which has a low absorption coefficient, causing low damping factors, i.e. small bandwidths. The tissue of the human vocal tract, on the other hand, has much higher absorption coefficients. In addition, during the open phase section of the glottal pulse, part of the vibrating air will be absorbed instead of reflected, which considerably increases the bandwidth of low formants. As a rule of thumb, the bandwidths of the formant peaks relate to the formant frequencies and are roughly equal to 1/20 of the formant frequencies, with the restriction that the absolute value of bandwidths will generally not sink below, say, 50 Hz or so. This implies that the bandwidths of formants lower than 1000 Hz will remain about 50 Hz.

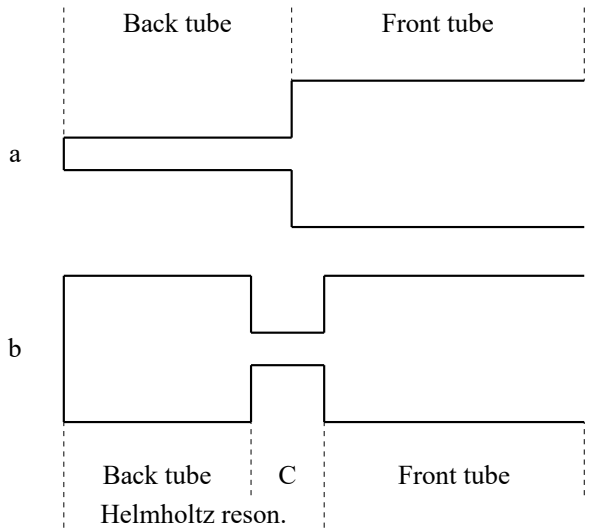


Fig. 19.5. Resonating tubes model of vocal tract. The back tubes of a and b resonate in different modes. In b the back tube together with the constriction (C) together act as a Helmholtz resonator as well.

The bandwidth values occurring in natural speech, combined with the generally small distances between formants, are the cause of the relative shallowness of the troughs between the peaks of the vocal tract filter function. However, sometimes the filter function shows low energy areas caused by 'side tubes' which are closed at the end. These may cause absorption of specific frequencies, sometimes called *antiformants*. The nasal cavity, for example, plays an important part.

Further details of tube models for vowel sounds fall beyond the scope of this book. Besides, there are many books on this subject (for example, see Rabiner and Schafer, [6]). For now, we can conclude that, in practice, the vocal tract filter function can be complicated considerably by the great number of peaks which may occur at unpredictable distances. In addition, the bandwidths of the peaks often cause the peaks to be relatively indistinct.

In normal speech, the relative bandwidths of speech formants are much greater than the relative bandwidths of many types of musical instrument resonators. For example, if the string of a piano or other plucked string instrument is regarded as a resonator, its damped sine wave typically has a relative bandwidth of about 1/1000. This characteristic implies that the 'resonators' of musical instruments as such (piano, guitar, harpsichord) can produce a very long sustained tone per string. In speech, the resonance energy has often almost damped out before the next glottal pulse excites the vocal tract filter again (at least, when the fundamental frequencies are not very high). These higher bandwidths are mainly caused by the higher absorption coefficients of the tissue of the vocal tract. In addition, because of the relatively fast repetition rate of the glottal pulses, the *tones* of speech sounds are determined by the glottal pulse frequency, not by the vocal tract resonances. The latter only determine the *timbre* of the tones, which can be adjusted to produce the different vowel sounds. Therefore, speech sounds differ greatly from sounds of musical instruments. (In section 25 we will discuss signals from musical instruments in some more detail.)

Because, in general, the damped sine waves of voiced speech formants are 'disturbed' at the start of the next F_0 period by a new start of damped sine waves, the spectral bandwidths of the undisturbed (i.e. once-occurring) damped sine waves are broadened by the spectrum of the fundamental period 'window'. From section 9 you may know that the spectrum of a periodical sound is equal to a 'sampled' version of the continuous spectrum of one period. Then, *one period* can be seen as a multiplication of the damped sine wave, from the start, with a rectangular window having a length of one F_0 period so that their continuous spectra are convolved. See fig. 19.6 where the 'underlying' continuous spectrum of a single F_0 period of an artificial one-formant 'vowel' is shown (black line), together with the spectrum of the untruncated once-occurring damped sine of the formant (red line). You can see that the peak of the spectrum of the damped sine is broadened by the main lobe of the sinc function of the rectangular window spectrum. The -3 dB width of the main lobe being about $0.9*F_0$, this means that, in principle, the

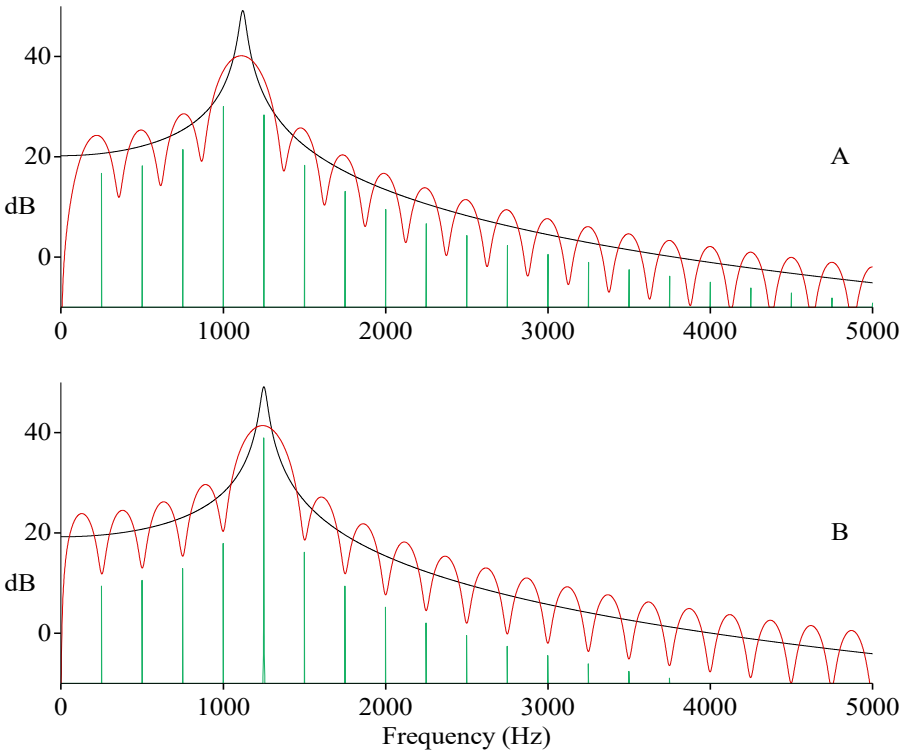


Fig. 19.6. Red: convolution of continuous spectra of the damped sine of one formant (black) and the rectangular 'window' of the F_0 period. A: the formant falls between two adjacent F_0 harmonics. B: the formant falls exactly on a harmonic of the F_0.

-3 dB bandwidths of formants cannot be lower than this value. As you can see, each side lobe of the continuous spectrum contains one spectral point of the discrete spectrum of the periodic vowel signal and the main lobe contains two of them. Only when the formant frequency is exactly a multiple of the F_0, the main lobe contains one spectral point and all other spectral points fall at the minima between the lobes. As, in normal speech, the formant frequencies do not depend on the F_0, this latter case is rather exceptional (more about this is discussed later), and it can be concluded that low bandwidth values relative to F_0 in speech do not manifest themselves as such and *therefore cannot be measured adequately*![1]

For a demonstration of this limited bandwidth influence, run DEMO 19.1. An artificial 'vowel' sound with only one formant is played, having a fundamental frequency of 220 Hz and a formant frequency of 1200 Hz. The formant peak thus falls roughly midway between two adjacent harmonics (i.e. the 5th and the 6th). First you will hear the sound with a 'normal' bandwidth of 150 Hz, then the same with a formant bandwidth of 50 Hz. The latter bandwidth even being only 1/3 of the former, you are unlikely to hear much difference. Next, the same is done, now the formant frequency coinciding with

[1] Linear Predictive Coding analysis (LPC) used for formant measurements can, in principle, present small bandwidths. However, these values are often not equal to the 'real' natural bandwidths but may result as a 'side effect' from the simplified mathematical model applied. In the next section this method is described in some detail.

the 5th harmonic. Although audible, the difference due to the bandwidth alteration in this case is rather small as well.

When the bandwidth B is low in relation to F_0, however, the remaining amplitude of the damped sine of the formant at the end of the F_0 period can 'disturb' the next period significantly. In that case, the intensity of a formant will be more dependent on the positions of the harmonics: it will have a higher intensity when it coincides with a harmonic as there is no phase shift from the end of the damped sine and the start of the next one. So, at low bandwidths (i.e. relatively high remaining formant amplitude), there will be some formant intensity dependence of F_0. As the bandwidth of normal speech is roughly proportional to the formant frequency, this intensity dependence vanishes at higher formants as, in these cases, the formant peak encompasses more harmonics and the amplitudes at the end of the F_0 period are practically zero. In that case, the intensities of the higher formants depend more on their bandwidths only because, within the F_0 period, the lower the bandwidth, the longer the train of periods of the damped sine wave before it peters out, resulting in higher formant peaks, and vice versa. In normal speech, however, the higher the formants, the lower their intensities, due to the roll-off of the glottal pulse, so that, in general, a possible bandwidth influence of the higher formants will play a minor role. This is indicated to some extent by the last part of DEMO 19.1: there, an artificial a-like vowel sound is generated containing four formants with a sweeping fundamental frequency, varying from 190 Hz to 110 Hz, at first with 'normal' bandwidths of 1/12 of the formant frequencies, then with low bandwidths of 1/36 of the formant frequencies. You can judge for yourself the influence of these large bandwidth differences on the vowel-like sound. (Although a low formant bandwidth causes greater amplitude fluctuations during the F_0 sweep than a high formant bandwidth, these increased amplitude variations are not strongly perceived.)

From this all, we may conclude that the bandwidth feature of formants plays *no important part in the perception of normal speech*, because of the spectral broadening by repetitive activation of the vocal tract filter. (In singing, however, some people can use the effect of increasing the intensity by 'tuning' a formant frequency so that it coincides with the fundamental frequency or one of the harmonics. Especially in the case of the high-pitched sounds that female singers produce, the amplitudes of the damped sines of the formants at the end of the F_0 periods are relatively high so that harmonics in the spectrum at both sides of the *tuned* formant (at the throughs of the lobes) will have a relatively low intensity, resulting in a distinctive spectral peak at the formant frequency. Some people can produce **overtone singing**, a very precise tuning ability where a formant is 'stepped' from one harmonic to another, such performing a simple melody while the F_0 is held steady.)

20. Formant measurements

If we want to estimate the formant frequencies from a periodical part of the speech signal which the microphone picked up, the vocal tract's filter function should in some way be reconstituted from this signal. Referring to the example of fig. 19.4: the speech signal from the microphone is represented in fig. 19.4d. For measuring the formants, we should try to 'convert' this signal into the graph of fig. 19.4c. In general, the following problem emerges: in case of high fundamental frequencies the filter function cannot be reconstructed reliably from the signal's spectrum. This is because the distance between the spectral 'samples' (the harmonics) is too large. When this occurs, low frequency formants may even become invisible (in some cases a formant frequency may even fall below the fundamental frequency), and higher formant positions cannot be determined accurately. Also, two formants close to each other may be detected as one formant.

If we apply the sampling theorem, mentioned in section 17, to the discrete 'samples' of the *line spectrum* of a periodic signal, we can conclude that the underlying continuous 'signal' (in this case the continuous spectrum) can only be reconstructed by frequency-domain sinusoids which have 'periods' of $2F_0$ hertz or higher. This means that the shortest distance between peaks of the continuous function which can be reconstituted from the discrete function should be at least about $2F_0$.[1] In other words: the formant/F_0 ratio should be higher than 2. This limits the formant measurement range at the low side. In addition, the 'formant measurement resolution' is limited to about $2F_0$ as well.

The range of the formants for average male voices goes from roughly 250 Hz to 5000 Hz and the F_0s of male voices range from roughly 70 Hz to 300 Hz (with an average of 120 Hz). This means that the lower formants cannot be detected reliably when the F_0 is higher than, say, 125 Hz. In many cases, therefore, this formant/F_0 ratio requirement is not fulfilled. For female voices the formants range from about 350 Hz to 5500 Hz and their F_0s go, roughly, from 100 Hz to 500 Hz (with an average of 220 Hz) so that their formant/F_0 ratios for low formants are even lower than the male values.

Infant speech formants depend very much on the age of the child. On average, infant formant/F_0 ratios fall somewhere between those of males and females. The average F_0 is not very different from the female value but the formants are much higher, mainly caused by the smaller dimensions of the infant vocal tract[2]. (The bandwidths are higher as well, as the *relative* bandwidths roughly do not change. So, the formant peaks encompass more spectral 'lines' compared with female formant peaks.)

---

1 In practice, this shortest distance can be somewhat lower than the $2F_0$ value, as the underlying continuous spectrum function varies relatively gradually so that the probability of *aliasing* is limited, as we will see later.

2 It seems that not all speech researchers are fully aware of this formant/F_0 ratio aspect: it is often assumed that measuring infant speech formants is more difficult than female speech formants although in general the opposite is true.

There are many methods for measuring formants from the spectrum of (voiced) vowel sounds. Let us look at some of them in some detail.

a. **The swept band filter analysis**.

Although the following method stems from old analog hardware analyzers not used any more, its working principle is worth mentioning because of the insight it may give. For this method it is necessary to either select sustained vowel sounds, or make the speech vowel sound you want to analyze perfectly periodical (this can be done by, for example, repeatedly replaying one specific fundamental period). As the name already implies, the speech signal is filtered by a band filter which is (slowly, because of the necessary response time) tuned through the frequency range of interest so that its center frequency shifts from zero frequency to the highest frequency of the range.

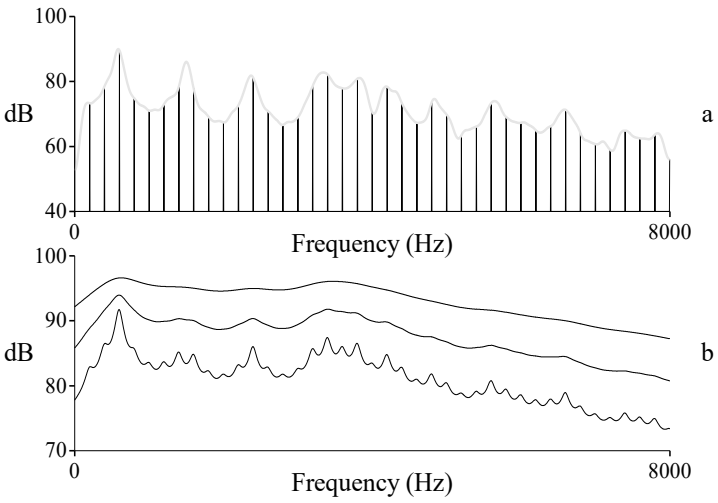


Fig. 20.1. a: Line spectrum of vowel from fig. 19.4 (slope corrected for source roll-off) and vocal tract filter function (grey line); b: 2nd order band filter spectra; from top to bottom: B = 600 Hz; B = 260 Hz, B = 100 Hz.

Viewed from the frequency domain, in each shift position the spectrum of the speech signal is multiplied by the shifted 'spectrum' of the filter. Thus, the total result can be seen as a convolution directly in the frequency domain of the speech line spectrum and the filter amplitude spectrum. In each shift position the filter output power can be seen as a measure of the spectral power of the speech signal at the current frequency band, just like the example of the radio tuning, mentioned in section 12. To get the spectral values, the average output power is extracted from the filter output for each shift position.

It will be clear that the result depends on the shape of the filter function, in particular its bandwidth. In fig. 20.1 this filtering method is applied to the vowel signal from fig. 19.4, using a 2nd order sweeping band filter. Fig. 20.1a shows the line spectrum of the signal, compensated for the roll-off of the source spectrum; fig. 20.1b shows the result, applying three different bandwidths. The bottom curve emerges when the

bandwidth is small compared with F_0; the mid curve occurs when the bandwidth is $1.3F_0$ and the upper curve corresponds with a bandwidth of $3F_0$. Obviously, for the measuring of formants the filter bandwidth should not be too low, otherwise the spectral harmonics become dominant. At the opposite end, when the bandwidth is too high, the spectral definition of the underlying continuous spectrum becomes poor. There is an optimum bandwidth which offers a best compromise between F_0 ripple and spectral definition. In the case of this filter type the optimal bandwidth is displayed by the mid curve, i.e. when the bandwidth is about $1.3F_0$. (This value has been found by trial and error.) The fact that the optimal bandwidth is lower than the $2F_0$ limit which you would expect according to the sampling theorem is caused by the gentle slopes of the band filter spectrum so that a wider area than the 3 dB bandwidth is used, thus making the interpolation between the spectral lines more smoothly. The consequence is that these gentle slopes mean great overlapping areas which, for a great deal, 'fill the valleys' in the curve so that the dynamic range of the amplitude values of the spectrum is very poor. (In fig. 20.1b the displayed dB range is even limited to only 30 dB to zoom in on the amplitude range.) Of course, many other filter types are possible but all alternatives are compromises between amplitude resolution, frequency resolution and side lobe suppression. A better compromise can be achieved by a *Gaussian filter*, which will be dealt with later on.

CHOICE OF FREQUENCY SCALE IN SPEECH SPECTRA

The frequency resolution of spectra of voiced speech parts is limited by the 'sampling' of the vocal tract's filter function at multiples of the fundamental frequency. Its implication is that, in principle, the peaks in the *measured* envelope spectrum cannot have bandwidths smaller than about $1.3F_0$. This value is constant at steady F_0, which means that the frequency resolution remains constant over the complete formant frequency range. The bandwidths of the peaks in the 'real' vocal tract filter function, however, will be roughly *proportional to* the frequencies of the formants. The measured bandwidths, therefore, will approximate the real bandwidths better at high values of the formant frequencies while the bandwidths of low formants are limited by the F_0.

To benefit from the high resolution at high formants the spectra of speech sounds are usually displayed at a *linear* frequency scale.

Commonly, in speech vowel spectral analysis the frequency axis is linear and the bandwidth applied is constant, as is also applied in this section. In some cases, a logarithmic frequency scale and a constant *percentage* bandwidth are used. See the box **CHOICE OF FREQUENCY SCALE IN SPEECH SPECTRA** for a general discussion about the choice of frequency scales for speech analysis.

b. **Band filter bank analysis**.

This method is practically equivalent to the swept band filter analysis. The only difference is that, in band filter bank analysis, many band filters run simultaneously while each band filter is permanently tuned to a different frequency section (frequency

band), together covering the entire frequency range. The output amplitude of each band filter represents one point of the measured spectrum (see fig. 20.2). In the figure, the output power envelope of each band-pass filter (BPF) is acquired by squaring and integration (Penv.) For a good approximation of the convolution as mentioned in method a. there should be a sufficiently high number of (overlapping) band filters to obtain enough frequency points.

For the choice of bandwidth, the same applies as mentioned in method a. A wide bandwidth causes greater frequency overlapping areas than a narrow bandwidth, so that the total number of frequency points in the range does not need to be as high as in case of a narrow bandwidth. In practice, for 2^{nd} order band filters the distance between the center frequencies can even be increased to as much as the 3-dB bandwidth to retain a smoothly overall curve.

Obviously, for this method there is no need to make the speech sound periodical: it is an 'instant analysis' which reacts on the incoming speech sound in real time. Of course, the band filters have a non-zero impulse response time so that the resulting spectra are delayed a certain amount of time.

c. **Band filter analysis by software**.

Methods a. and b. naturally can be much more easily performed by software after digitizing the speech signal that is to be measured. For example, you can easily construct repeating signal periods to make a long and steady signal. In addition, digital processing also makes it possible to apply all kinds of filter types, including digital

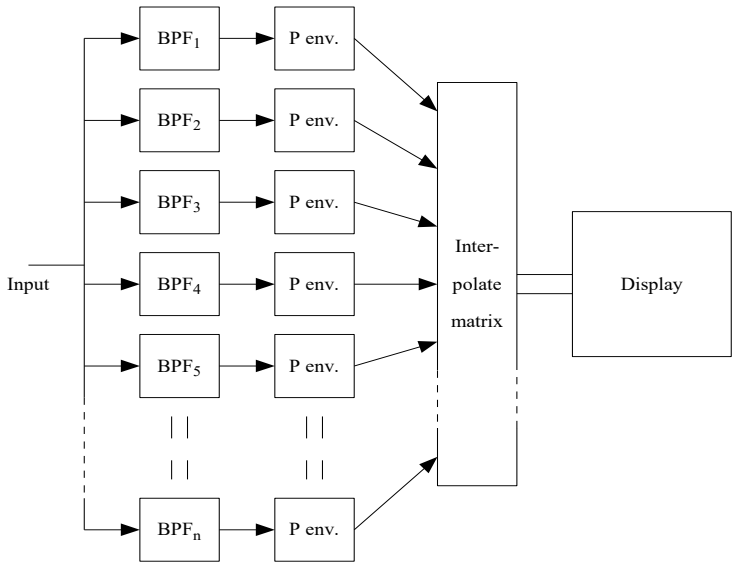


Fig. 20.2. Spectrum analysis by bandpass filter bank.

filters, which are very efficient (see section 18). The band filtering can be carried out in the frequency domain, using convolution as described above.

As an example, analysis using software makes it easy to apply real Gaussian filters. (Strictly spoken, a Gaussian filter cannot be realized by using analog hardware. In practice, however, it is possible to realize very close approximations.) Compared with the 2nd order band filter spectra mentioned above, the Gaussian band filtering produces much better results. See fig. 20.3 for a Gaussian band filter spectrum of the vowel shown in fig. 19.4. The bandwidth applied is $0.9F_0$ which seems to be the best compromise between F_0 ripple and spectral definition for the Gaussian band filter.

d. **Windowed sinc cepstral smoothing (WSCS).**

Another method is reconstructing the filter function in the manner which is described in part A about sampling (section 17). There, the reconstitution of the continuous time function from the sampled version was carried out by convolution of the sampled function with a sinc function (see fig. 17.4). We can do likewise in the frequency domain: reconstructing the continuous frequency function from the sampled frequency function (the line spectrum) by convolution with a *frequency domain* sinc function. In practice, the sinc function has to be truncated or, even better, windowed, thus minimizing the effects of the discontinuities. This leads to the *windowed sinc filter* as mentioned in section 18 (see fig. 18.1).

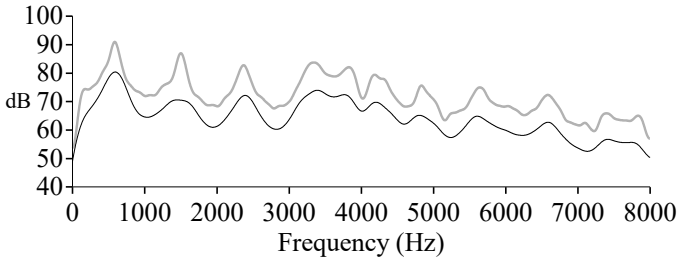


Fig. 20.3. Vocal tract filter approximation of vowel from figs. 19.4 by Gauss filtering with 3 dB bandwidth of $0.9F_0$. The vocal tract filter function is shown as well (grey line) for comparison.

However, when the formant/F_0 ratio criterion described above is not met, the continuous filter function will be *undersampled* and the reconstructed continuous function may be a poorer approximation of the vocal tract filter function. This can be seen in fig. 20.4a where the line spectrum of the vowel sound of fig. 19.4 is convolved with a Gauss-windowed sinc function with 17 lobes (8 side lobes at both sides of the main lobe). The deviations from the vocal tract filter function (the grey line), stem from *aliasing* caused by undersampling of the continuous spectral vocal tract function. When the vocal tract filter is sampled with a 300 Hz fundamental instead of a 200 Hz one, the deviations from the original continuous function become quite severe, as shown in fig. 20.4b. To

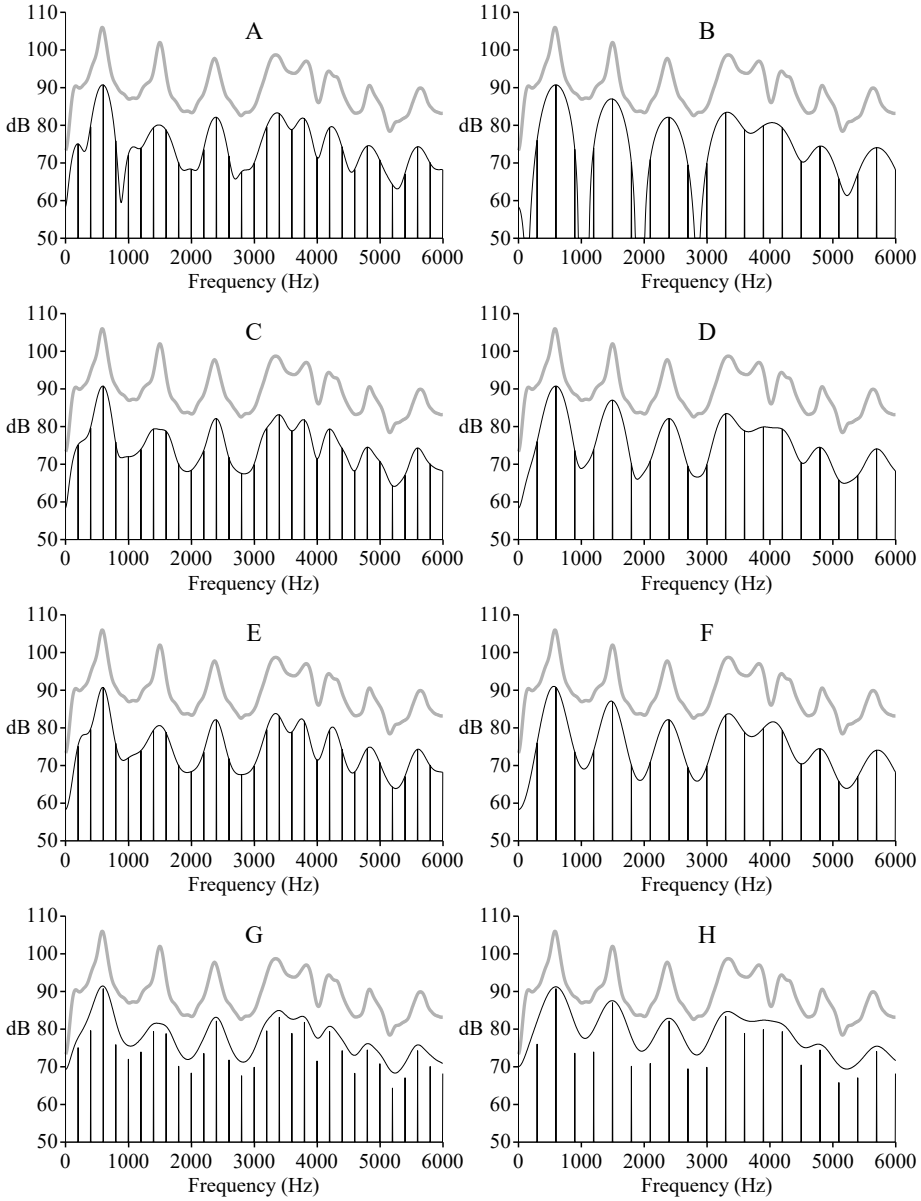


Fig. 20.4. A through F: Different windowed sinc filters applied to line spectra of the vowel sound of fig. 19.4. Left column: F_0 = 200 Hz; right column: F_0 = 300 Hz. In A and B, the width of the sinc function is $18F_0$; in C and D: $8F_0$; in E and F: $22F_0$. E and F show the result of width $22F_0$ when the *log values* of the spectrum are filtered, equivalent to *Cepstral smoothing*. In G and H, the result of Gauss filtering with $B = 0.9F_0$ is shown for comparison. The original vocal tract filter is shown by the grey lines shown above the black curves in the figure.

limit the occurrences of these aliasing effects we may attenuate the components that are close to the Nyquist by making the sinc filtering less steep. Obviously, the price to pay for this is a lower spectral resolution of the result. Figs. 20.4c and fig. 20.4d show the result when a sinc filter with only 7 lobes (3 side lobes at both sides of the main lobe) is Gaussian-windowed. Especially in the case of the 300 Hz fundamental, the result is greatly improved.

A further improvement can be achieved by filtering the *logarithmic* spectral values instead of the linear ones: the logarithmic function is usually much smoother than the

linear one so that less aliasing can be expected. This is done in figs. 20.4e and 20.4f where a sinc filter with even 21 lobes is applied, and we can indeed see some improvement. In fact, this sinc filtering of the log spectrum of a periodic signal is called **cepstral smoothing**, the word *cepstrum* referring to the *forward* Fourier transform of the power spectrum, as if the log spectrum were a time function. Some people even use words like *quefrency* and even *liftering*. In spite of the words specially invented for this analysis, the method stems from the sampling theory and, in fact, is similar to *Shannon's reconstruction theorem* as mentioned in section 17. [1]

To compare the results of figs. 20.4A through 20.4F with the Gauss filtering described in method c, the 'ideal' Gaussian filter, with a bandwidth of $0.9F_0$, is applied to both fundamental frequency examples and displayed in figs. 20.4G and 20.4H. As you can see, the difference between this one and the windowed sinc filtering of figs. 20.4E and 20.4F is not very great, however, in all sinc filtering examples the resulting curve is seen to go through all sample values, whereas the Gaussian filtering fills the throughs to some extent so that the peaks are less prominent than those of the windowed sinc filtering.

Theoretically, the reconstruction applies to spectral *lines*, i.e. the DFT components of exactly one period or an integer number of periods. However, the reconstruction method described works also on a windowed part of the vowel sound. In that case there are no pure spectral lines but *spectral functions* of the window, due to the convolution

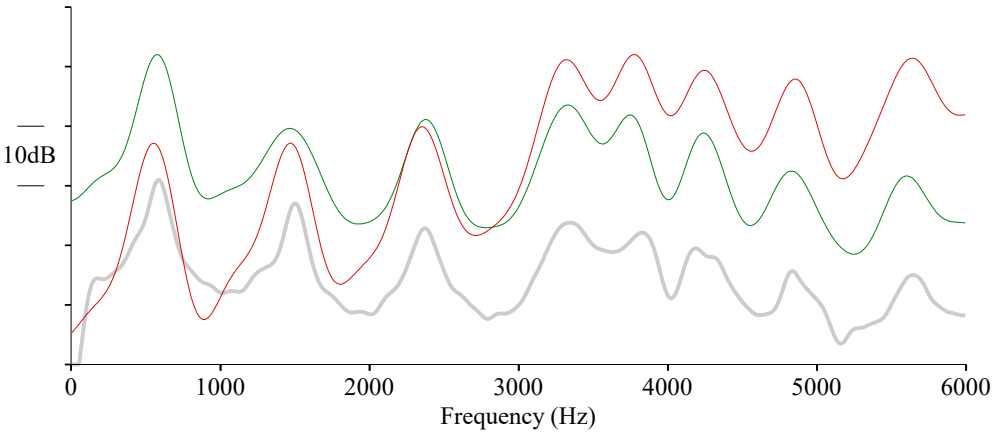


Fig. 20.5. Windowed sinc cepstral smoothed spectra of 100 ms windowed part of vowel sound of fig. 19.4. Green line: constant F_0=200Hz. Red line: F_0 shifts 10 Hz within 100 ms window. The vocal tract function is shown as well for reference (downshifted grey line).

[1] In Praat the possibility of cepstral smoothing has been built-in. Instead of a windowed sinc function, however, a Gauss function is applied. This means that, if the proper bandwidth is chosen, the result is similar to the 'Gauss filtering' mentioned earlier, but applied to logarithmic values. However, this has the consequence that the troughs between the peaks are less 'deep' caused by the relatively high influences of the harmonics at the sides of the peaks which means that this is not the best method for formant measurements.

in the frequency domain (see section 10). In this way, there is no need to isolate exactly an integer number of periods. Even when the fundamental frequency varies somewhat within the selected signal window (which in practice will always occur to some extent), the method works reasonably well if the sinc function applied is matched with the mid value of the F_0. See fig. 20.5 where a 100 ms Hann windowed part of the signal is analyzed. The green line is the result when the F_0 is constant and looks almost the same as the spectrum of the interpolated DFT components of one period, as shown in fig. 20.4.E. The red line shows the result when the fundamental frequency of 200 Hz is varied 10 Hz within the 100 ms window length. The result is not very different from the green graph, apart from a certain spectral slope. This is caused by 'filling the gaps' between the spectral lines in the higher region: the shift of the n^{th} harmonic is n times the shift of the fundamental so that, for example, the 20^{th} harmonic is shifted up to 200 Hz in this case. In fact, this spectral lift can be compensated, as the F_0 variation within the window can be measured. It is important that the peaks remain at practically the same frequencies.

(In the sinc filtering examples described the convolution was performed using the symmetrical spectrum, i.e. including the negative frequencies, thus avoiding transient effects after filtering of the sudden start of the spectral function at 0 hertz.)

As can be seen from fig. 20.4, the deviations from the original continuous vocal tract function in the case of the high F_0 of 300 Hz are substantial. This may disappoint phoneticians who want to extract formant frequencies, but one should not expect a spectral analysis method to extract local formant frequency information which is not present in the isolated signal segment. (If a method does, it will generally produce wrong results!) It may be wise to realize that very weak or absent formant peaks in the speech signal cannot be perceived by listeners as well. (And people learn to control the various speech vowel sounds by listening to them in the first place.)

e. **The spectrogram**.

As you may recall from part A, if we multiply the speech signal with a window function which is the impulse response of the filter applied, the spectral components of the signal will be convolved with the spectral function of the window. Therefore, analysis of the local spectral components of the signal can be achieved by choosing the correct window length. This opens the possibility of analyzing a complete speech utterance (a sentence for example) by moving a window along the time axis and computing the local spectrum of the windowed part of the signal at each position in time.

To display this information, three dimensions are needed (time, frequency and intensity/amplitude). In general, this is done using a graph called a **spectrogram**. See fig. 20.6 for an example of a spectrogram made with the Sound editor of the program Praat of the phrase "his lips glued together". The horizontal axis represents time, the vertical axis frequency and spectral intensity is expressed by the intensity or greyscale

of each point. Analysis programs that produce spectrograms like these almost always apply (stepwise) moving windows as described above. The spectrogram offers a very useful way to make visible the different sounding speech parts (**phonemes**).

In Praat, the default settings of spectrogram parameters use a moving Gaussian window, causing no visible side lobes in the spectra. In each window position in time, the Fourier Transform is computed. The bandwidth can be chosen at will, by setting the window

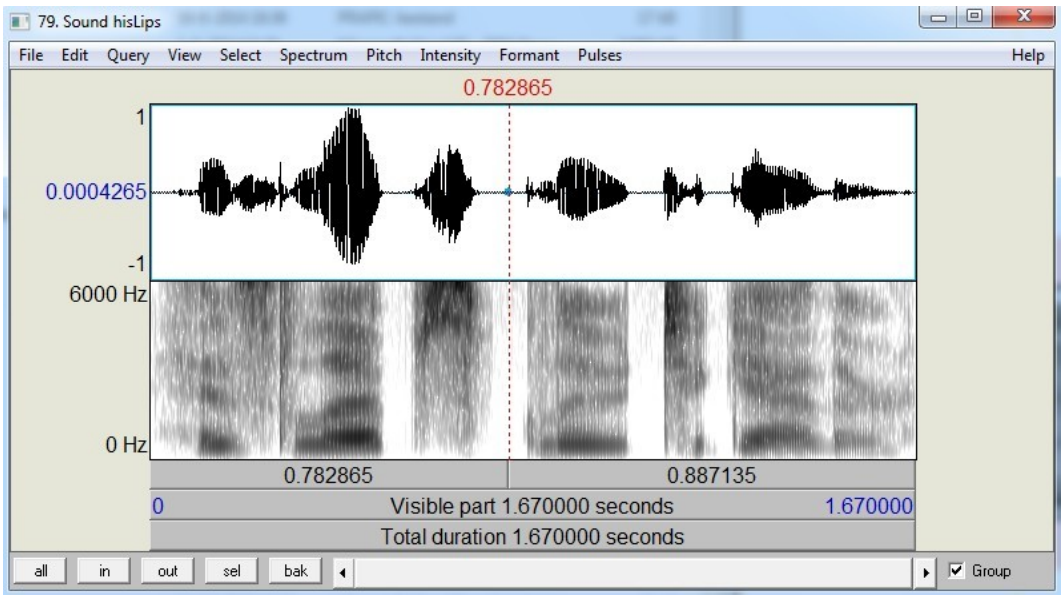


Fig. 20.6. Praat's Sound editor, displaying the waveform and spectrogram of the English phrase "his lips glued together".

length[1]. (For the 'optimal' bandwidth of $0.9F_0$ for measuring formants, the window length in the Praat spectrogram should be set to about $1.4/F_0$ seconds.) The steps are much smaller than the window length, so that the overlap causes smooth transitions from one spectrum column to the next one.

In fig. 20.7 two Praat spectrograms of the same phrase as used above are displayed. In the top spectrogram the bandwidth is small with regard to the fundamental frequency so that the separate harmonics of the vowel's F_0 can be seen as *horizontal* lines (which are bending gradually caused by the changes in the F_0). In the lower spectrogram of the figure the bandwidth is much wider than the F_0 so that the separate harmonics are not visible any more. Instead, there are (somewhat vague) *vertical* lines. These occur in the positions where the vocal tract filter starts to react on the steepest parts of the glottal pulses, where the high frequency spectral components have the most energy. With the

[1] Because the Gaussian window has no time limits, Praat defines the Gaussian window length used in spectrograms as the time between the positions where the function values are 5% of the top value. Using formulas 13.3 and 13.4 from section 13, it can be calculated that the spectral bandwidth of the untruncated version of this window function is about 1.3/ (*defined window length*) hertz. The actual window length applied by Praat is twice as long, which limits the side lobes due to truncation to negligible values.

band being as wide as it is, the impulse response of the band filter applied is very short.

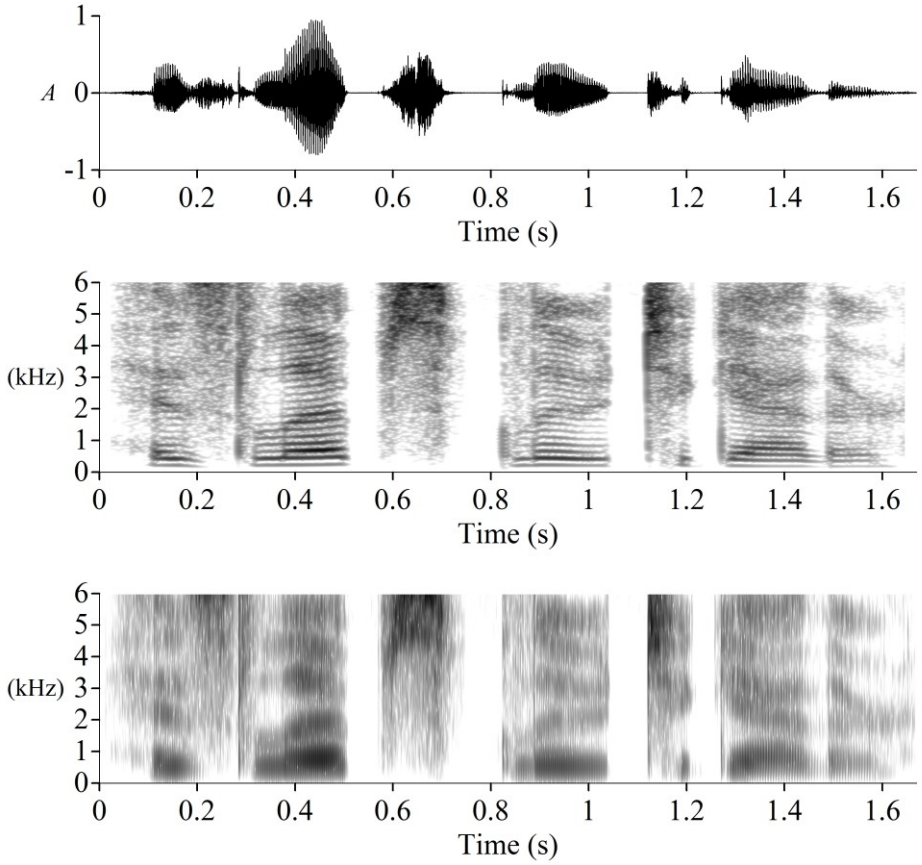


Fig. 20.7. Waveform (top), narrow band spectrogram (mid) and wide band spectrogram (bottom) of the English phrase "His lips glued together".

In other words: the *time resolution* is great while the *frequency resolution* is poor. It is the other way around when the bandwidth is small. (The old-fashioned hardware speech *spectrograph*, therefore, had a bandwidth switch to select either *narrow band* or *wide band* analysis.)

There is an important difference between the moving window principle used in spectrograms of this type and the swept band filter or band filter bank analysis: in the latter cases the spectral component values are computed over a relatively long part of the signal (in the case of the swept band filter analysis the signal is made steady by repeatedly replaying its fundamental period during the complete frequency sweep). In other words: it produces a **long-term spectrum**. In the moving window method, however, the result is a **short-term spectrum** when the window applied is only as long as a few F_0 periods. Now the result is highly dependent of the exact time position of the window. The shorter the window, the greater the spectral dependence of the position will be. You may run DEMO 20.1 for a display of consecutive spectra from a

spectrogram of the sustained vowel as described above. Here the bandwidth chosen is about 0.9 times the F_0, the 'best' value for formant measurements with Gaussian windows. You can see that the spectra vary greatly and that in many positions the spectra deviate much from the original vocal tract function!

This problem of the moving window in the spectrogram teaches us an important fact: *for measuring formants* it is not a good idea to use a spectrogram and simply apply the 'optimal' bandwidth of, say, $0.9F_0$. Although the values between the spectral harmonics are interpolated, the resulting spectra depend highly on the window position in time within the steady vowel part. If we look at the spectrogram as a whole, our estimation of the formants can be done quite reliably due to our brains' ability to do a global filling-in of the 'weak points' using the surrounding values, but the spectral intensities and exact spectral peak positions cannot be estimated accurately from the picture. This is why automatic formant measurements using data from spectrograms, like these, suffer from these 'weak points' and we cannot do this without being confronted with a vast amount of false formant data. The only way to avoid these spectral fluctuations is by using a greater moving window length. Of course, the F_0 ripple will then emerge, which in turn will hamper the formant estimations. Consequently, where formant estimation is involved, the task remains to extract the envelope spectrum from the F_0 multiples.

We saw that by using a swept Gaussian band filter or a band filter bank we were able to approximate the continuous vocal tract filter from the periodical signal spectrum of the vowel sound quite satisfactory (see fig. 20.3). As we explained in section 10 the swept band filter result is equivalent to the convolution of the spectrum of the signal and the frequency domain filter function. (In fact, the graph of fig. 20.3 is constructed like this.) For *formant measurements*, therefore, it would be advisable to convolve the Fourier transform of sufficiently long Gaussian windowed signal selections with the frequency domain function of a Gaussian window, having the 'ideal' width of $0.9F_0$. Of course, the computation time to display a spectrogram in this way would be much higher. The advantage, however, is that the formants could be extracted much more accurately and the width of the frequency domain Gaussian window could be automatically tuned to the local F_0 values so that the spectral filter width is optimal in each moving window position. Automatic formant measurements could be done very well from these spectrograms by *peak detection* algorithms.

The fundamental frequency in speech fragments normally varies a lot. Having examined the methods described we may conclude that optimal estimations of the continuous envelope spectra require that the measuring parameters should be adapted to the local F_0.

f. **LPC**.

The abbreviation LPC stands for **Linear Predictive coding**, a name that can be clarified by the description which follows. During speech, the changes of the vocal tract over

time will be relatively slow w.r.t. the speech wave form fluctuations: a substantial alteration of the vocal tract filter function will usually take at least a couple of F_0 periods. Thus, because the changes in the vocal tract occur relatively slow, taking a short piece of the signal will present us with a seemingly unchanged filter function. Therefore, a relatively short time interval of the speech signal can be regarded as the result of the filtering of the glottal pulse (or a noise signal in case of whispered vowels, for example) using a filter with steady parameters, resulting in a number of formants with constant frequencies and bandwidths. Let's assume that the spectral slope has been compensated for by the right amount of pre-emphasis of the speech signal. This means that its source signal has a flat spectrum then and can be seen as either a Dirac pulse train or white noise while the spectrum of the speech signal is completely defined by the filtering properties. As a thought experiment, suppose that the speech signal is the *result of filtering* of the source signal with a *digital recursive filter* as described in section 18. There the general formula (18.11) for a digital recursive filter was stated as:

$$y_n = a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2} + \ \ldots + b_1 y_{n-1} + b_2 y_{n-2} + \cdots \qquad (20.1)$$

where the x terms refer to the input samples and the y terms to the output samples. Because all sample values are caused by the filtering (whereby the overall power level depends on the source power), the sample values can be seen as the output of an *in-line* filter which simplifies the formula by leaving out all x terms. By convention the LPC filter formulas use a's instead of b's:

$$y_n = a_1 y_{n-1} + a_2 y_{n-2} + a_3 y_{n-3} + \cdots + a_k y_{n-k} \qquad (20.2)$$

That means that each sample value can be seen as the sum of the values of k preceding samples, each multiplied with a coefficient a_i where i is the number of sample steps before the occurrence of the n^{th} sample. The number of preceding steps k can be chosen to define the order of the filter. Because each formant peak can be seen as a 2nd order filter section, the order k of the complete filter should be twice the expected number of formants. During the relatively steady time intervals of the speech (the chosen *window length*) we may assume that the values of the filter coefficients (the a factors) will not vary much. Therefore, the way of thinking is that each sample could be *predicted* to a certain accuracy using its k preceding sample values. Because of the facts that the filter with order k will not be exactly equal to the real vocal tract filter function and the vocal tract filter will vary somewhat, there will be a difference e between the predicted new sample value and the real new sample value:

$$y_n = a_1 y_{n-1} + a_2 y_{n-2} + a_3 y_{n-3} + \cdots + a_k y_{n-k} + e_n \qquad (20.3)$$

So, the error can be stated as:

$$e_n = y_n - a_1 y_{n-1} - a_2 y_{n-2} - a_3 y_{n-3} - \cdots - a_k y_{n-k} \qquad (20.4)$$

Thus, the error function can be expressed as a function of the set of *a* coefficients, each multiplied with its current sample value. To find the *a* coefficients, the procedure is to minimize the error function during the filter steps through the window. Mathematically, the error is defined by the average of squares of all *e*'s.[1] To find its minimum the 'standard' mathematical method can be used (taking the derivative and set it to 0) which produces an equation with the variables a_1, a_2, ... a_K. For a signal that is sampled with a sufficiently high sample frequency, the number of samples within the steady interval is much greater than the order *k* of the filter, so that a 'new' error equation arises for each shift in time, with a new set of *a* coefficients. The result is a set of *k* equations, each with *k* variables, which can be solved mathematically in a number of different ways. (The exact procedures are beyond the scope of this book. There exist many books on this subject. See for example Rabiner and Schafer [11] and Weenink [14].) So, the set of *a* coefficients can be updated with each new step of the in-line filter, always using *k* formerly calculated *a* coefficients.

Going through the speech signal with steps in this way the time domain filter function will be updated with each step. Finally, the frequency domain function can be extracted according to the time domain formula 20.2, using the *z transform* (see Appendix IV), which will produce the frequency positions of the peaks (the *poles*) and their bandwidths. Apart from the peaks, the vocal tract filter may occasionally have minima (*zeros*) as well. Although this LPC filter is an *all pole filter*, having no zeros, it is generally assumed that the formants can be reasonably analyzed by this type of filter, as the peaks are more important than the dips with regard to human perception. (I fact, the LPC filter is a *cascade filter* which means that the output of one filter section is the input of the next. In practice, the oral and nasal cavities of the vocal tract form two parallel signal paths which causes the LPC analysis producing greater errors for nasal speech sounds.)

The structure of the error formula (20.4) shows that the error signal can be seen as the result of a certain digital filtering of the speech signal. Assuming that the speech signal spectrum contains all vocal tract filter information, the error signal must have a flat spectrum (because the spectral slope has been corrected). This implies that the filter of formula 20.4 apparently 'filters back' the speech sound to the source signal which must have a flat spectrum, theoretically either Dirac pulses or white noise. So, this filter must be the *inverse* of the filter from formula 20.2. (Indeed, an **inverse filter** can be made by changing the sign of all *a* coefficients and adding a coefficient $a_0 = 1$.) The error signal can be regarded as the source signal, separated from the speech. The pitch as well as the voiced/unvoiced property can easily be extracted from this error signal.

The values of the *a* coefficients will vary only slowly with time, as long as the speech signal is relatively steady. Therefore, for *coding* of the speech it is sufficient to process only the average formant information per chosen window. Together with the pitch data

[1] The method of squaring the errors instead of taking the absolute values is chosen for its easier mathematical manipulation, in particular for the possibility of taking the derivative for calculating the maxima and minima of a function.

and voiced/unvoiced info, the complete speech signal can thus be coded. This obviously offers a substantial data reduction compared to the sampled speech version. Of course, this compression method may decrease the signal quality considerably, dependent on the chosen filter order. (More on signal compression in Part B section 24.)

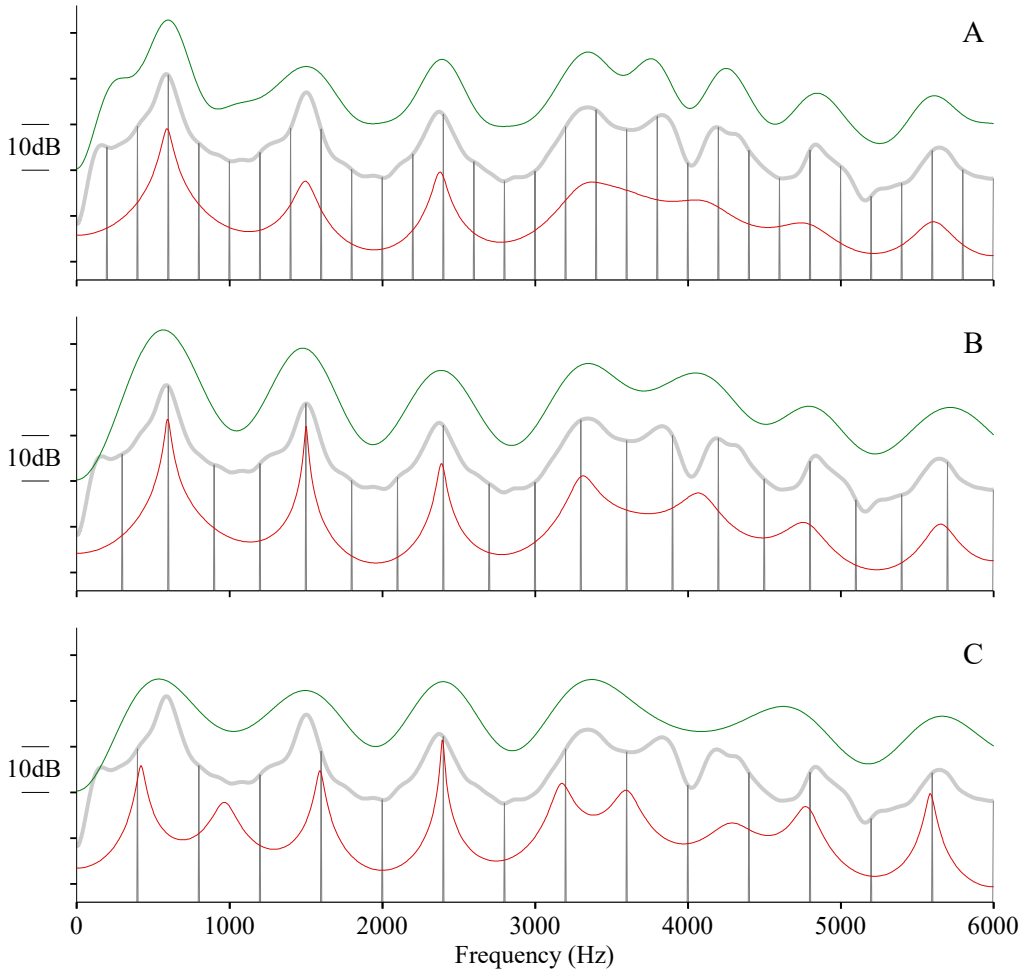


Fig. 20.8. LPC filtering of vowel sound of fig. 19.4 (red lines). A: F_0=200Hz. B: F_0=300Hz. C: F_0=400Hz. The windowed sinc cepstral smoothing results (green lines) and vocal tract filter function (grey lines) are shown as well (graphs vertically shifted apart for clearness).

To compare the LPC method to analyze *formants* with the methods mentioned before, see fig. 20.8, where the LPC analysis is applied to the vowel sound of fig. 19.4. Three values of F_0 are tested: 200 Hz (A), 300 Hz (B) and 400 Hz (C). The lower graphs (the red lines) represent the LPC filtering. No pre-emphasis is applied as there is no roll-off in the example. The green lines represent the windowed sinc cepstral smoothing, as mentioned in method d, and the grey lines show the vocal tract filter for the purpose of reference. The harmonics are drawn as well, to show their possible influence on the peak positions. The frequency range for the formants (the *formant ceiling*) has been

limited to 6000 Hz and the filter order chosen is 18 (for a maximum of 9 formant peaks in the range 0.6000 Hz).

As you can see, the bandwidths of the LPC peaks are often much smaller than those of the windowed sinc cepstral smoothing *and the vocal tract itself* which implies that the LPC bandwidths cannot be used for measuring the 'real' bandwidths of the vocal tract peaks.

It appears that, often, the heights of the LPC peaks bear no relation to the corresponding vocal tract peaks. Apparently, they depend to a large extent on the fundamental frequency: the peak heights are raised considerably when the peak frequencies coincide with the positions of the harmonics.

The frequency positions of the LPC peaks are more or less in agreement with the vocal tract peaks (except in part C of the picture). In both cases, when F_0 is 200 Hz and F_0 is 300 Hz, there are three peaks visible in the range 3000...5000 Hz where the vocal tract shows four peaks. The order of the LPC filter should allow for 9 peaks totally, however, so in that case we should expect a display of all 4 peaks in this range.

When F_0 is as high as 400 Hz (fig. 20.8C) the LPC analysis can even detect four formants in this range. However, the peak positions of the first two of them are shifted considerably. Furthermore, the most important first formant of about 600 Hz (the lowest in frequency) has been shifted a lot (almost to the first harmonic) in this LPC graph, and a 'new' peak between the first and second peak emerges. Apparently, as in spectrum B, here the LPC spectral peaks tend to shift to the nearest harmonics as well. This LPC graph of the 400 Hz vowel is therefore of very limited use in practice.

Of course, the *undersampling* of the vocal tract at high F_0 (especially in the B and C parts of the picture) causes lack of details of the vocal tract in the (green) graphs of the windowed sinc cepstral smoothing so that in the range 3000...5000 Hz only three peaks in B and two peaks in C are visible. The deviations from the values of the vocal tract graphs, however, seem smaller than in the LPC cases. It appears that this type of cepstral smoothing is more reliable when measuring spectral energy distribution of the vowel signals, especially in case of high fundamental frequencies.

In addition, there are a number of criteria for the use of LPC for formant estimation, as described in the literature about LPC analysis. To mention some in simplified form:

1. The order of the LPC filter must be chosen in advance. In the example above the LPC order was adapted to the number of vocal tract filter peaks, but in practice, the vocal tract filter is not known. The number of (prominent) formants depends on the type of vowel (an /u/ vowel usually has fewer formants than an /a/ vowel, for example).
2. The frequency range has to be limited (by downsampling of the signal), otherwise a great number of formants must be 'placed' beyond the practical

formant range to cover the entire frequency range to the Nyquist frequency, needing a very high filter order which would increase the chances on false positions of formants under consideration (the limitation of the *formant ceiling*).
3. Nasal vowel sounds can cause dips (*zeros*) in the vocal tract filter. The LPC filter defines nothing but peaks and then may produce errors when 'trying to imitate' the vocal tract filter.
4. The spectral slope has to be compensated by pre-emphasis of the speech signal. That would not be a problem if the spectral slope is known but in practice it is not and may vary a lot, depending on the speaker and the way the vocal folds are used. The standard 6 dB/oct. often would not be the right compensation.
5. There must be a sufficient number of samples within the window applied (the LPC *frame*). Therefore, a window length of one F_0 period, for performing a period-by-period analysis for example, can only be used when the F_0 is not too high.
6. Noisy sounds will often cause a great number of incorrect peak positions.

Adding the findings of the LPC experiment as described:

7. Bandwidth data are not reliable in practice.
8. Peaks may be totally absent or peaks may emerge in wrong positions when the F_0 is high.
9. 'Prominences' of peaks cannot be estimated reliably from the LPC filter. There seems to be only a weak relation between the peak heights and shapes of the analysis and those of the vocal tract functions of the vowel sounds.

In relation to the last item a general remark. When trying to reconstruct 'difficult' formants from speech sounds one should realize that if the analysis used does not show information about the formant frequencies one expects to be present, this should be inherent to the signal itself and preferably not be caused by some limitation of the method of analysis. One would like to detect as much as possible of the underlying vocal tract function in the signal part of interest. So, when the spectral 'undersampling' limits the information about a peak of the vocal tract filter function, this should not result in a peak at a different frequency, and also not in a peak with a very small bandwidth.

Summing up the advantages of the windowed sinc cepstral smoothing over the LPC method for formant measurements:

1. No limitations to the setting of the frequency range (no *formant ceiling*).
2. No critical setting of filter parameters (apart from the necessary adaptation to the local F_0).
3. Spectra of nasal sounds can be measured without difficulty.
4. No need to compensate for the spectral slope.
5. It is possible to make a period-for-period tracking of the spectrum by isolating one period and smoothing its DFT components. High F_0 values result only in low frequency resolution, not in wrong peak positions and heights.

6. The susceptibility to noisy sounds is much less than that of LPC and can even be limited by applying a relatively long window in case of a steady signal part so that the noise is attenuated by averaging.
7. The bandwidth data are much more reliable and the 'prominence' of the peaks can be easily valuated. The spectra seem to be related more directly to the perception of speech sounds.

(This summary of advantages also applies to some extent to the Gaussian filtering method. In that case, as regards item no. 5, an isolated period can be recirculated to make a signal segment of sufficient length.)

The limitations of the windowed sinc cepstral smoothing can be summarized as:

1. The filter parameters (i.e. the sinc function) must be adapted to the local F_0.
2. When applying a window to select a part of the vowel sound, its length should be limited, dependent on the F_0 variation.
3. The measured bandwidths of low formants are increased, depending on F_0. (which is a property of the signal, not of the analysis method).
4. The computation efficiency is low.

We may conclude that, from the methods described, the *windowed sinc cepstral smoothing* is the best method for extracting formant peaks from *voiced* speech vowel sound sections, provided that *the cepstral filter properties are matched with the (local) F_0 values*.

The methods for estimating formants mentioned so far are only a few of the many measuring strategies that have been developed over time. More recent investigations often focus on the *higher level* of speech perception in the brain and analyze larger parts of speech so that the context of the word or message plays a major part in formant estimation. Having knowledge about the limitations of what can be measured from local segments, however, is important for separating the analysis results caused by context information from those caused by signal properties.

No doubt, of all the methods for speech formant estimation, the LPC analysis is by far the most popular. This is probably caused by its high processing efficiency and assumed high accuracy. Although the comparison of LPC analysis with other methods as described here is not very extensive, it may be wise to give a warning to be careful in the selection of a spectral analysis method for speech signal segments and not 'blindly' use LPC, especially when speech perception is involved.

By using the link "View cepstral smoothed spectrum at cursor script" on my home page you may download a script to experiment with the harmonics interpolation method. The script contains an instruction on how to add a button in Praat's sound editor for this purpose.

21. Unvoiced speech

Many speech sounds are produced without vibration of the vocal folds: these are the **voiceless** sounds. Some of them are produced by building up some air pressure by blocking the air flow for a short time, followed by a sudden release of the obstruction. The resulting sounds are thus a combination of a short silent interval and a 'popping' noise. This is why they are called *voiceless plosives* (mainly the sounds of p, t and k). In the spectrograms you may locate them by the presence of relatively strong high frequency components during a short time. Other voiceless speech sounds are the voiceless *fricatives*: the vocal tract is narrowed at some place, leaving only a narrow opening, which causes the air flow to become turbulent. This produces a hissing noise, its timbre (i.e. spectral properties) depending on the position of the constriction in the vocal tract, and the size of the opening left. When this constriction is realized by the vocal folds (without letting them vibrate!) the entire vocal tract will filter the sound of the source in the same way as described before (section 19). This is how *whispering vowels* are produced.

To estimate formants of whispered vowel sounds, or spectral properties of fricatives in general, we should take into account the random fluctuations of the airflow, causing random variations of its spectrum. In other words: we have to handle the analysis *statistically* by proper averaging, as mentioned in section 15 concerning noise. See fig. 21.1A where a spectrum of 0.1 seconds of a sustained whispered /a/ sound is displayed (black graph). Obviously, the peaks of the spectrum, i.e. the formants, cannot be extracted accurately from the huge number of peaks. In addition, in spite of proper windowing of the signal (a Hann window is applied), the spectrum shows many very 'deep' dips: apparently, some of the bins of the spectrum contain almost no energy. As the time length is 0.1 s the spectral bins are only 10 Hz wide. First of all, we need wider frequency bins to increase the probability that they contain components with some energy. In other words: we need smoothing in the frequency domain for averaging frequency components. The red line in fig. 21.1A shows the same spectrum smoothed by a Gaussian window of 100 Hz bandwidth. The number of peaks has been limited greatly although the number of 'formants' seems implausible. Besides, in fig. 21.1B the spectra of *a number* of separate 0.1 seconds parts of the whispering /a/ have been smoothed by the same 100 Hz Gaussian window. The differences between the individual spectra are quite high, showing that the formants are not estimated with reasonable accuracy. In fig. 21C the lengths of the whispered separate /a/ parts were increased to 2 seconds. Now the number of formants as well as their positions seems much more reliable. (The remaining differences are mainly caused by the 'natural' speech production variations.)

We may conclude that for analyzing fricative speech sounds there is a necessity for averaging spectra of steady parts with *sufficient lengths* of similar phonemes. In most cases the signal of only one fricative in a normally uttered sentence will not produce enough information for estimating its spectrum reliably. For estimating formants of a

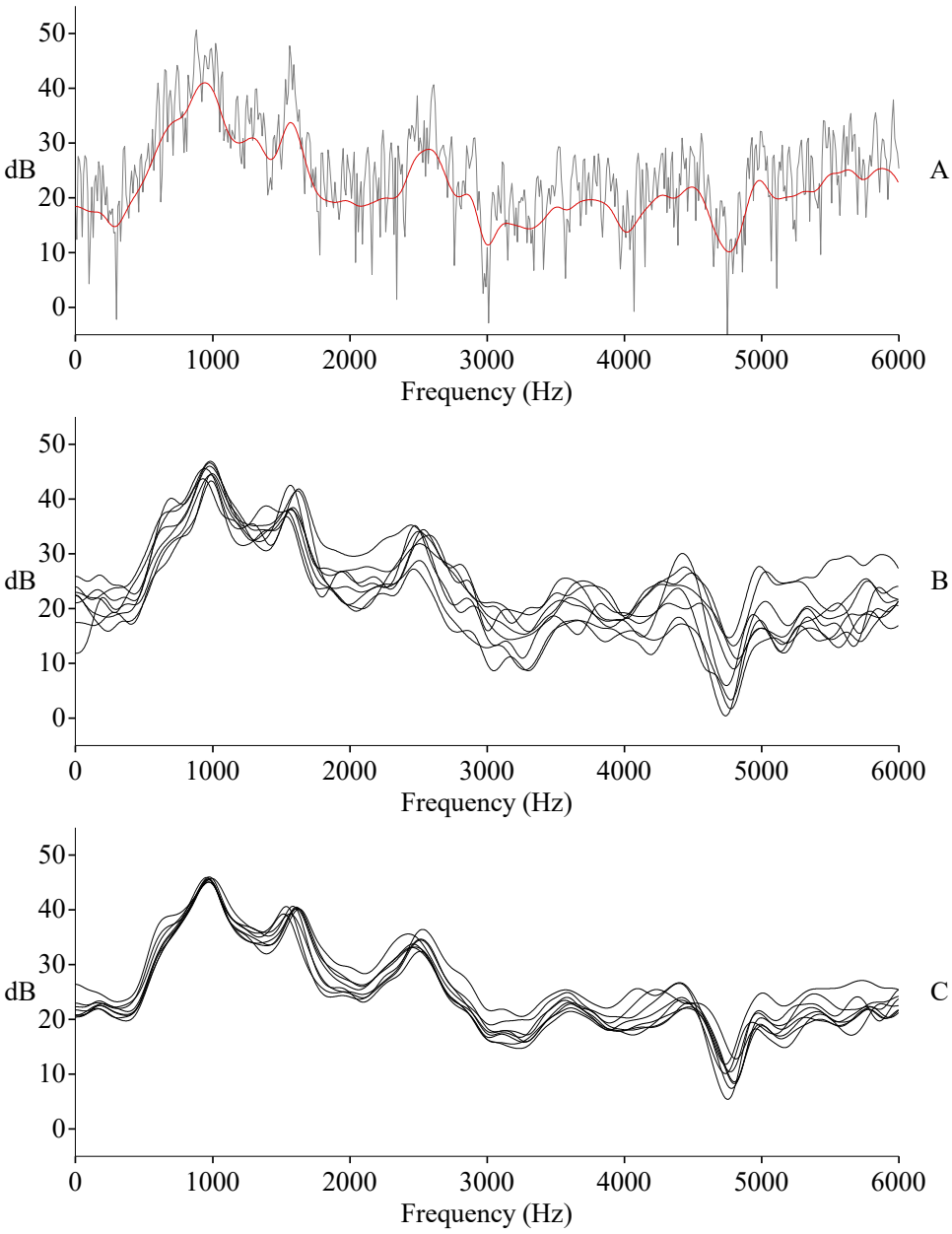


Fig. 21.1. A: Spectrum of 0.1 s of whispered /a/ (black line), and its Gaussian smoothed version with *B*=100Hz (red line). B: Gaussian smoothed spectra of eight separate 0.1 seconds parts of whispered /a/ on top of each other. C: The same as B, now of 2 seconds parts.

speaker's fricative phonemes with some accuracy, many measurement positions of identical fricatives should be used, while accepting some broadening of the peaks caused by the natural differences.

22. Prosodic features

In addition to the spectrogram, which shows the varying spectrum along the time axis of an utterance, there are a number of other acoustic speech measurements as functions of time. Often, the variations in time of these acoustic measurements occur on a larger time scale than those in between the phonemes or segments. As they give the sentence or utterance its structure, these speech properties are called *prosodic features*. Because of the larger time scale of these features compared with phonemes or segments, they are also called *suprasegmental* variations. Naturally, these variations are not absent within the phonemes or short segments but within these short time intervals they are much smaller and often regarded as being constant. In fact, many types of measurements can only be performed from a certain length of signal (the *window* used) so that the value obtained must be regarded as some kind of average over the window length at a specific position in time. For variations of a single variable along the time axis one speaks of a *contour* like *pitch contour*, which shows F_0 as a function of time, and *intensity contour* which shows the logarithmic intensity as a function of time.

In linguistics some terms are used to describe prosodic properties of speech which refer to the perception by the (generalized) listener, like *intonation*, *accent*, *pace*, and many more. Mostly, they should be regarded as subjective properties. At best they can be seen as a combination of objective measurements. For example, intonation depends on *pitch*, *intensity* and segment *length*, accent depends on *intensity*, *length* of syllable and *pauses*, pace (or speech rate) depends on the average number of words or syllables per unit of *time*, etc. In addition, the individual contributions of the various objective measurements to the 'strength' of a subjective feature are dependent on the language, the speaker and even on the contents of the spoken sentence itself. Obviously, this is no subject for this book, which will not go beyond the handling of basic objective signal measurements. There is no need to: there are many libraries full of books about linguistics. Only a few basic measurements of prosodic features will be described in some detail here.

22.1. Pitch[1]

Fig. 22.1 presents an example of a pitch contour of a spoken phrase, measured in Praat, together with its waveform. (The phrase is uttered by a male English native speaker.) The default method in Praat for pitch measurements is *autocorrelation*, a method already mentioned in section 16 (see fig. 16.4). The program calculates the autocorrelation of a Hann- or Gaussian-windowed part of the signal and takes the point of time where the autocorrelation has its maximum (of course excluding the maximum at T=0). If this time shift is T the pitch value is $1/T$. The window moves on along the time axis and the autocorrelation is calculated anew to compute a new pitch. The window moves with small steps to achieve sufficient overlap so that a smooth pitch curve can be displayed. The window length must be sufficiently long to contain a few periods of the lowest pitch but not too long to enable to display the curve with some detail. The Hann-window length that Praat uses is three times the lowest pitch period

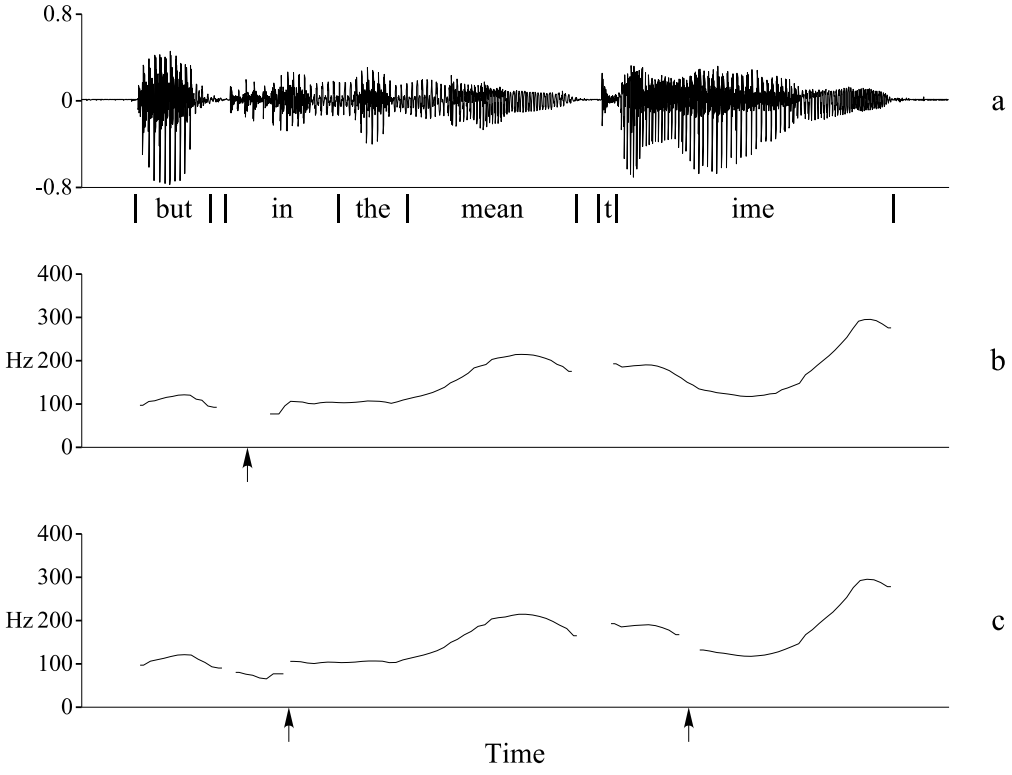


Fig. 22.1.1. a: Waveform of spoken sentence section. b: Its pitch contour measured with Praat's autocorrelation method with 75 Hz as lowest limit. c: As b. but lowest limit now 65 Hz.

that is defined by the user. Praat's default value of the lowest pitch is 75 Hz so that the Hann window has a length of 40 ms. The autocorrelation will also present peaks at

[1] As mentioned before in section 16, the word pitch indicates a subjective value. Nevertheless, the F_0 is meaned here.

shifts which are multiples of the period. When the periods of a signal part are highly identical, these multiple period peaks can be equally high or even higher than the first one. So, when the maximum correlation is found at twice the period, for example, the pitch found at that position will be one octave lower than the real value. To avoid this, there has been built-in a small bias to the higher pitch values. When the signal is noisy, or has a poor periodicity, these 'octave jumps' sometimes cannot be avoided completely, however. The lowest pitch, therefore, should preferably be set higher than half of the expected values.

There is a second reason why it is preferable to avoid very low values of the lowest pitch setting. In fig. 22.1.1.b the pitch contour was made with the default lowest pitch setting of 75 Hz. At the marker the pitch seems absent whereas in the waveform some periods can be seen at that position. Lowering of the lowest pitch to 65 Hz causes the pitch to emerge there, as shown in fig. 22.1.1c. However, at the two marked areas the pitch is absent, which obviously is not correct. How can that be? When the lowest pitch is set to 65 Hz the window length is so long that the pitch within the window (which is more than 46 ms now) varies so much that the autocorrelation peak is lower than the *voicing threshold* (another default pitch setting). Lowering this voicing threshold will solve this but will produce more spurious pitch values in other cases.

Obviously, apart from the problem that a too high setting of this lowest pitch will miss very low pitch values, a setting like this will also increase the probability of occurring upward octave jumps. This could occur especially when a strong low frequency formant is present. Avoiding upward octave jumps by lowering of the highest pitch setting cannot be applied, because the pitch range of natural voices covers more than one octave. The general message is that the pitch parameter settings, especially the lowest pitch value, should be defined with care.

In Praat, instead of the Hann window, the Gauss window can be chosen. It creates the possibility to measure pitch with very high accuracy in cases of steady parts of the speech. However, the effective window length is twice the length of the Hann window which means that the problem of the fast-varying pitch as described above will be greater. As the autocorrelation method itself already offers high accuracy pitch measurements, the Hann window will be a practical choice for most speech pitch measurements. [1]

In addition to the autocorrelation method for pitch measurements, Praat offers the *cross-correlation method*. While, in each measurement position, the auto-correlation method takes the correlation of the *complete windowed part* with itself, the cross-correlation method uses a variable window length. See fig. 22.1.2 for an explanation. At each side of the measurement position (mp) a window is placed such that both windows are adjacent to each other and have equal lengths. The windows start with the shortest

[1] The autocorrelation method in Praat has been made very accurate by modifying the ac data by dividing each ac function of the windowed part by the ac function of the selection window. See Boersma [1]

length necessary to contain the defined highest pitch period. From the two windowed signal parts the cross-correlation *factor* is computed, by multiplying and integration (see for an explanation of this cross-correlation factor section 6 about signal

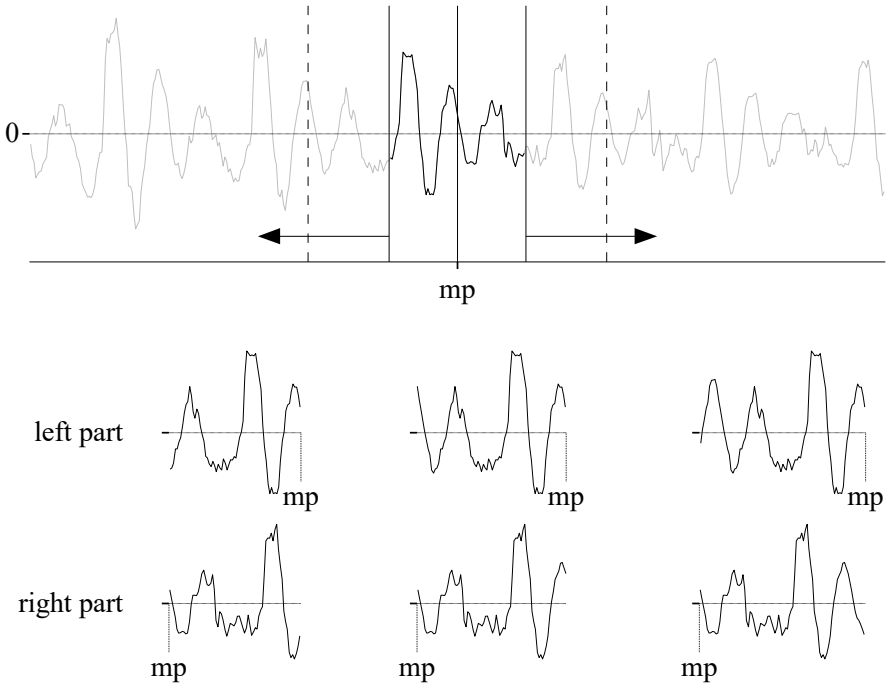


Fig. 22.1.2. Cross-correlation method for pitch measurement. The equal duration of the adjacent parts is varied from the lowest to the highest limits. The duration which gives the highest cross-correlation between the left and right part is the most probable pitch period.

comparison). Now the two windows are increased one step in length and the new cc factor is computed again. This process goes on until the window length for measuring the lowest pitch period has been reached. Now the window length where the highest cc factor occurred is equal to the period of the most probable pitch at the measurement position[1]. This procedure is repeated at each measurement position.

This cc method is able to follow the pitch movements in more detail than the ac method and therefore can better cope with the lowest pitch problem of the ac method. The noise independency and accuracy, however, are not as high as the ac method. In general, for normal voices the ac method is the best compromise whereas the cc is more suitable for pathological voices, where the pitch periods may vary irregularly.

[1] In fig. 22.1.2 the windows drawn are rectangular for simplicity of explanation. In reality, however, a Hann or Gauss window is applied, as mentioned before.

22.2. Intensity

Usually. the different speech sounds within a sentence have very different intensity values ('volumes') due to the way they are generated. In addition, intensity variations within a sentence are 'deliberately' altered by the speaker, to give the speech its dynamic structure. As we know from section 5 the intensity of a sinusoidal wave refers to its power, which relates to the square of the amplitude. The same applies to the *speech waveform* which means that all instantaneous values of the signal have to be squared, summed over a specific time period and averaged. What time period? If we want to see all fast-changing details of the **intensity contour** we might prefer a very short window which moves along the time axis. However, we are not interested in the short time intensity variations *within the periods* of the waveform. Therefore, the chosen setting will be a compromise between time resolution and allowed pitch ripple. See fig. 22.2.1 for an example of intensity contours of a speech vowel sound, applied with different window lengths. To avoid the pitch-synchronous intensity variation as a result of a short window we should average over at least two or three times the longest period that occurs in the signal, i.e. the period as determined by the lowest pitch. In general, a certain setting for the intensity measurements remains constant while the pitch period can vary a lot. Therefore, the chosen setting will be a compromise between time resolution and allowed pitch ripple. See fig. 22.2.1C where still some pitch ripple is visible in a low pitch part of the utterance, while the window length used here is 20 ms.

Because the setting of the window length should depend on the pitch period, in Praat the window length used for measurement of the intensity contour has been made equal to the window length defined for measuring the pitch contour, i.e. three times the defined lowest pitch period. In addition, a (pseudo) Gaussian window is applied to avoid sudden steps in the contour. Furthermore, to create a smooth intensity curve, the time step chosen of the moving window is shorter than the window length which creates some overlap. In Praat, the time step is ¼ of the window length. Together with the window type used, this results in a negligible residual ripple (0.00001 dB) for a periodical signal having the lowest pitch defined and a constant intensity. So, in practice, the ripple can be regarded as suppressed completely provided that the window length applied is derived from the longest pitch period. To avoid a big loss of time resolution, however, a certain F_0 ripple can be allowed for, like in the example of fig. 22.2.1C.

Commonly, the intensity contour values are displayed using a dB scale. As you may know from section 2 and 3 the linear values of the waveform are regarded as sound pressure levels (SPL) and the dB values refer to the hearing threshold of 20 μPa. However, as explained in section 3, there is no calibration of the waveform values due to the unknown amplification and sensitivities of the sound equipment used. The dB values of the intensity contour, therefore, must not be regarded as the representation of real sound intensities that occurred during the recording.

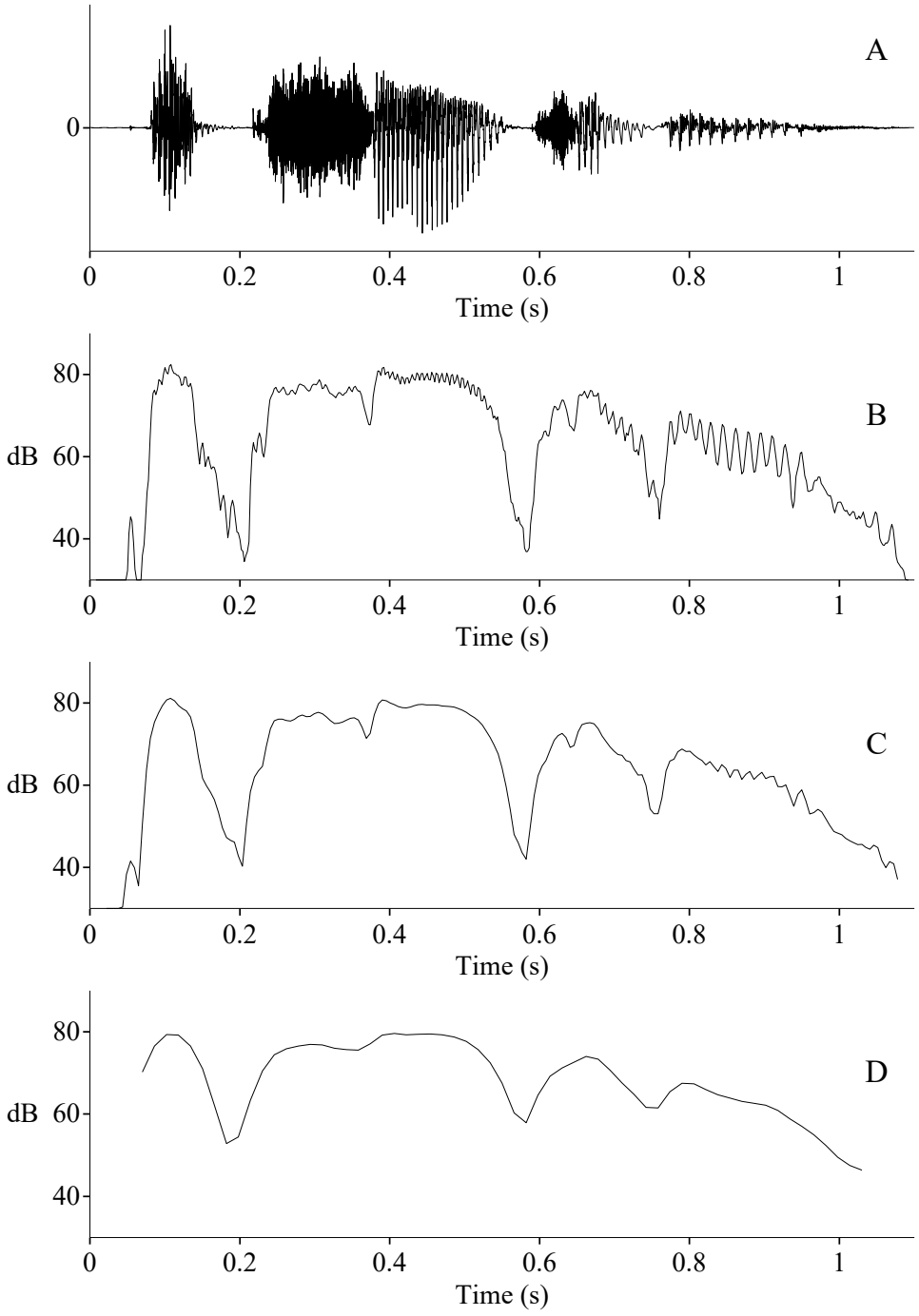


Fig.22.2.1. Praat's intensity contours of spoken word "excitable". A: waveform. B: intensity measured with 7.5 ms window. C: the same with 20 ms window. D: the same with 60 ms window.

Reminding note: because a sound recording in wav format cannot have values higher than 1 or lower than -1, the theoretical maximum value which can occur in the intensity object of Praat is 94 dB, corresponding to 1 Pa. In that case the waveform will contain only values equal to 1 or -1 as no values exist in between. What is the maximum

intensity if the wav sound is a *sinusoidal* wave? We know from the box **RMS** of section 5 that the intensity of a sine wave with amplitude A is $\frac{1}{2}A^2$. The maximum (*unclipped*) amplitude of the sine wave is 1 which means that the intensity value is ½. As log(½) = -0.3 the dB value is – 3, so the sine wave cannot have intensity values higher than 94-3 = 91 dB. As explained in section 5, the corresponding rms value is $1/\sqrt{2}$ which is about 0.7.

22.3. Speech rate

Some types of speech research require to get some figure about the speed of speaking by subjects, *the speech rate* or *pace*. In many cases this figure is determined by measuring the number of spoken syllables per unit of time. The dependence of the differences of syllable lengths then is minimized by estimation of the mean of a great number of measurements per speaker.

To detect the syllables of many utterances, a program is needed to perform the task in an automatic way. The program may use the intensity contour, possibly combined with a detection of *voicing* of the signals, to detect the presence of the syllables. Because of the great fluctuations of the intensity values within one syllable and the mostly close connection of adjacent syllables, the automatic detection of syllables cannot be accomplished without some errors. Fig. 22.3.1 displays an example of an output from a program (a Praat script, which is published by N. de Jong and myself, see ref. [6]) which has proven to work relatively accurate, applied to the spoken sentence of fig. 22.1.1. In this program, among other things, the intensity contour is measured in

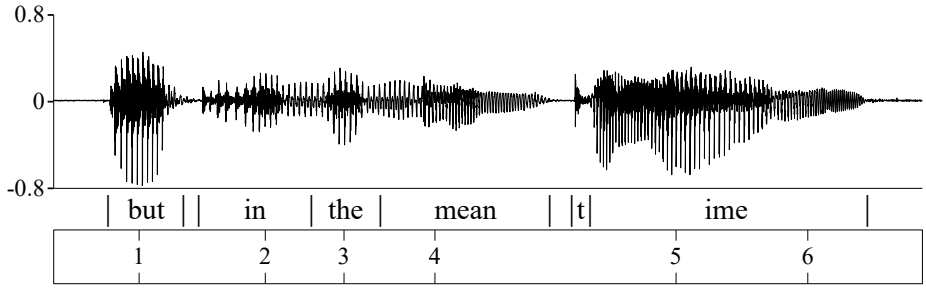


Fig. 22.3.1. Automatic detection of syllables in spoken sentence.

two steps: one with a long and one with a short window. In most cases the long window intensity contour will show one peak at each syllable and the short window contour will detect the minima between the syllables. By a combination of the contours the errors are minimized. You can see that the last syllable has been counted erroneously as two different syllables, as the long window intensity contour counts more than one peak due to the exceptional length of this syllable.

This problem must be seen as a general difficulty to handle complex tasks (like the assignment of syllables from spoken sentences) by the measurement of relatively simple parameters. Nevertheless, in many cases great improvements can be achieved by combining a number of basic measurements. Here, using the pitch contour and/or the spectrogram as well, for example, could avoid errors like the one shown.

On the other hand, errors cannot be avoided completely and, therefore, a great number of sentences should be measured to be able to apply statistical methods to obtain sufficient reliability of the results, even when using only simple parameters.

Anyway, the researcher (as always!) should be aware of the limitation of the measurement accuracy of the analysis applied.

23. Overall properties of speech signals

On a larger time, scale than that of segments and sentences, there are objective measurements possible which may give some statistical information about specific 'overall' speech signal properties. The four most commonly used are mentioned here.

a. Ltas

Naturally, apart from short speech segments, spectra can be made from long speech recordings like complete stories told. Hence the name: **Long time average spectrum**. Of course, all detailed spectral information has vanished then because the value of each specific frequency in the spectrum is an average of the intensity of that frequency over the whole spoken story. Nevertheless, comparison with Ltasses of other speakers may give some clues about their speaking behavior. Especially in cases of speech impediments or defects Ltasses can be of use. Important to mention here is, to be able to compare Ltasses, preferably all recordings should have been made in the same room, using the same equipment, because the overall spectral shape depends partly on the room acoustics and microphone properties.

As you may know from part A, the spectral bin width of the Fourier transform is equal to $1/T$ where T is the length of the time interval. Therefore, the Fourier spectrum of a long sound may contain a tremendous number of points. For example, when the speech recording has a length of 60 s, the spectrum from 0 to, say, 10000 Hz contains 600000 points! A lot of 'overkill' as the display of the spectrum will seldom need more than 1000 points or so. Also because of the great fluctuations of spectral energy along time, there is no need to get a very high spectral 'definition'. For this reason, the Ltas to perform can be specified by defining a spectral bin width, which usually is set to 100 Hz or so.

In Praat there is no restriction on the maximum or minimum of the sound length. It is very well possible to get a Ltas from a short segment. What is the difference then with a Fourier spectrum? In fact, there is no difference if you set the Ltas bin equal to that of the Fourier spectrum, apart from the underlying structure: the Fourier spectrum contains linear values whereas the Ltas contains dB values. Also, the Fourier spectrum is complex: it has cosine and sine values which makes it possible to inverse Fourier transform the spectrum to sound whereas the Ltas cannot be reversed transformed to sound again. But the displays of Fourier spectrum and Ltas are exactly equal if their bin widths correspond.

In the case of a short sound the 'long time" part in the name of the Ltas does not make much sense and you could regard the Ltas then as a 'rebinned' Fourier spectrum. The purpose of using Ltas for short segments could be a globalization of their spectra. In most cases, however, a spectral smoothing (i.e. as mentioned in section 20) is preferred instead which omits the sudden steps that may occur in the Ltas.

b. Jitter

When the pitch of a voiced speech sound, like a vowel, is held steady, all periods of the fundamental frequency will have not exactly the same duration. There exist some fluctuations of the pitch: the **jitter**. Usually these fluctuations occur *at random* and are very small. In some cases, these fluctuations follow a *pattern*: the glottal period lengths may alternate between two values (which results in a weaker subharmonic of half the nominal pitch as the lowest pitch comprises two 'original' periods). Also patterns which comprise more than two periods may occur so that lower subharmonics will occur. A third kind of deviation of the steady signal is generated when the vocal folds act like two independent sources, each having a (mostly small) different fundamental frequency. This latter category counts not as jitter.

When different pitches can be heard at the same time the effect is sometimes called **diplophonia** or **triplophonia**. In normal voices these regular patterns are mostly negligible. Besides, all natural voices will have some small *local* random variations of the pitch periods. These variations must be distinguished from the audible variations, generally *on purpose* by the speaker or singer, like **vibrato**: a periodic *modulation* of the fundamental frequency at a pace varying from 5 to 15 Hz or so. So, a complete 'vibration cycle' contains about 10 or more pitch periods and on this time scale one speaks not any more of jitter.

It is customary to express jitter not as pitch deviations but of the mean of the deviations expressed as percentage of the nominal pitch period:

$$\text{jitter} = \frac{100 \cdot \sum_{i=1}^{M-1} |T_{i+1} - T_i|}{(N-1)T_M} \tag{23.1}$$

where N is the total number of periods, T_i the current period in seconds and T_M the mean length of all periods. Obviously, the jitter can only properly be measured from sustained vowel sounds where the pitch is held unaltered. When that condition is met, the practical value of jitter for normal healthy voices is about 1 percent or lower. Because of the natural pitch gliding variations on a greater time scale, which cannot be avoided by the speakers, there exist some methods that take the differences of the current period and the mean of some *local periods around the current period*, instead of only the differences of adjacent periods. In this way a gradual increase or decrease of the pitch has less influence on the jitter figure because the reference is defined by the 'moving average' of the local periods. Mostly 1 or 2 neighboring periods are applied, which averages 3 or 5 periods respectively. In Praat there are built-in even five different jitter measurement methods. Probably, the method according to formula 23.1 (called *local jitter*) is the most widely used.

It will be evident that the reliability of jitter measurements depends highly on the pitch measurement method. Some commonly used pitch measuring methods, however, are

based on amplitude peak detection and, therefore, much more dependent on the presence of noise in the speech signal. To detect the small variations of the pitch due to jitter the influence of noise should be of great care so that it is important that the jitter measurements should be derived from autocorrelation pitch measurements like the pitch measurement method mentioned in section 22.1, which have great noise immunity.

c. Shimmer

Apart from the pitch fluctuations of a speech sound held steady, like a sustained vowel, also the *amplitude* fluctuations can be measured. From each period the peak-to-peak amplitude is measured and, likewise in the case of jitter, the mean difference of peak-to-peak amplitudes between adjacent periods is expressed as a percentage of the overall mean peak-to-peak amplitude:

$$\text{shimmer}(\%) = \frac{100 \cdot \sum_{i=1}^{N-1} |A_{i+1} - A_i|}{(N-1) \cdot A_M} \tag{23.2}$$

where N is the total number of periods, A_i the current peak-to-peak amplitude and A_M the mean peak-to-peak amplitude of all periods.

Here again, to limit the influence of 'natural' gradual amplitude changes on the shimmer figure, a moving average of the amplitudes of a few periods around the current one is used. In practice, the number of local periods applied for shimmer ranges to 3, 5 and 11.

Instead of the peak-to-peak amplitude ratio, sometimes the dB differences are used:

$$\text{shimmer}(\text{dB}) = \frac{\sum_{i=1}^{N-1} |20 \log(A_{i+1}/A_i)|}{(N-1)} \tag{23.3}$$

d. Harmonicity

The autocorrelation (ac) function of a sustained vowel sound offers the possibility to measure its *harmonics-to-noise ratio* (HNR), also called **harmonicity**. In case of a purely periodic signal without noise its autocorrelation exposes maxima at $t = 0$ and at all positions where t is a multiple of T or $-T$: the period. These maxima are all equal and represent the power of the periodic signal (at these positions the signal is multiplied by itself and the power is equal to the square of the amplitude). When the signal contains noise and the noise is not correlated with the signal (which usually is true in practice), the autocorrelation of the noise itself is added to that of the purely periodic component of the signal. Now the autocorrelation of the noise itself has only one maximum at t = 0

(because the noise itself is not correlated at all). This means that the autocorrelation component at $t = 0$ for a noisy signal is the sum of the signal component and the noise component. See fig. 23.1 for an example of a noisy periodic signal and its autocorrelation function. You can see that the maxima of the function are all equal, except the one at $t = 0$. So, for the harmonics-to-noise power ratio we simply have to divide the value of the 'repeated' maxima by the difference of this value with the value at $t = 1$:

$$HNR^2 = \frac{r_{xx}(T)}{r_{xx}(0) - r_{xx}(T)} \tag{23.4}$$

where r_{xx} is the autocorrelation function as described in section 16. Usually, just in case of the *signal-to-noise ratio* as mentioned in section 15, the HNR is expressed in dB's:

$$HNR(dB) = 10 \cdot \log \frac{r_{xx}(T)}{r_{xx}(0) - r_{xx}(T)} \tag{23.5}$$

In fact, the formula is only valid for stationery signals. As, in practice, a sustained vowel sound will never be strictly stationery, we need to select and window parts of the sound.

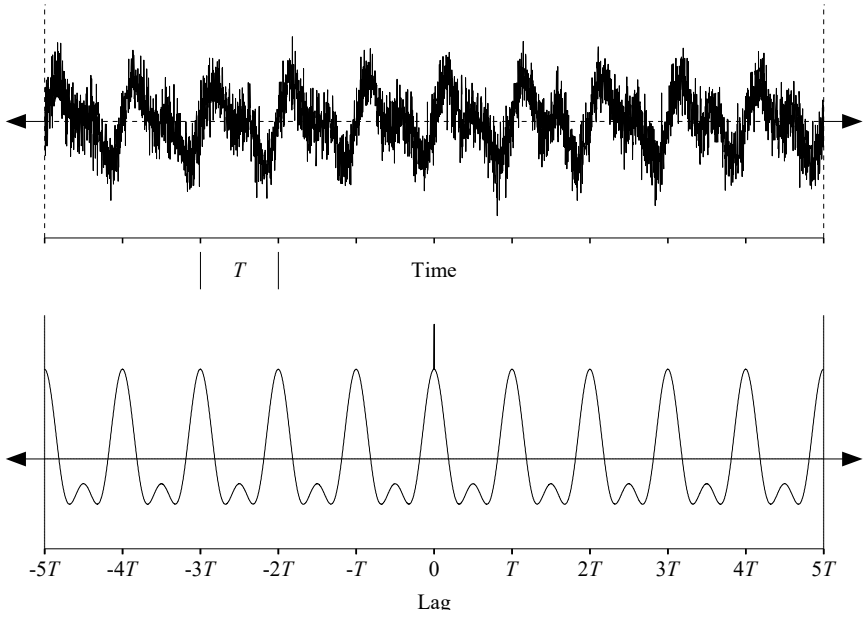


Fig. 23.1. Top: part of noisy periodic signal. Bottom: the autocorrelation is the sum of the ac of the periodic part (all equal peaks) and the ac of the noise (one peak at t = 0).

Moving the window through the length of the vowel sound produces the HNR as a function of time: the HNR *contour*. In Praat, the influence of the window on ac measurements is limited by dividing the local ac functions by the ac of the window used which results in a high accuracy of the HNR. (For a detailed explanation see Boersma [1].)

Of the types of overall measurements of speech described, the Ltas depends mainly on the global vocal tract properties whereas the jitter and shimmer give information about the glottal source only. The harmonicity is also basically a feature of the source but the output is somewhat dependent of the spectral properties of the vocal tract filter. Because of their focus on the source signal, especially the latter three types are mainly used in cases of pathological voices, generally the result of vocal fold functioning impediments. The jitter measurement, in addition, is also well-known in the general field of signal analysis, as a measure of oscillator (generator) stability.

The shimmer being a measurement where amplitude is involved, you may wonder why its output does not highly depend on the vocal tract filter properties. Naturally, the amplitudes of the waveform of different vowel sounds can vary considerably due to the change of the vocal tract filter. The shimmer, however, computes the amplitude *ratios* of a few adjacent local periods. Changes of amplitudes as a result of changing the vocal tract filter occur on a time scale that is longer than that of the local periods taken as the local reference. Besides, the types of overall measurements described are usually applied to sustained vowels so that the short term amplitude variations can be regarded as being caused by the source properties only.

24. Sound data compression

The digital representation of audio signals with high quality needs a considerable quantity of bits per second, as we have already seen in section 17. For a stereo audio file with CD quality the sample frequency is 44100 Hz and the number of bits per sample is 16. In that case the required bit rate is 2 x 44100 x 16 = 1.4112 million bits per second, or 176.4 kilobytes per second, as a byte contains 8 bits. Not surprisingly, many signal manipulations have been developed to limit the number of bits per second to represent the audio signal, i.e. to **compress** the sound. Formerly, the main reason to compress was the limited recording time on computer disks or removable media as the storage space was expensive. Now, the costs to store data are extremely cheap: at the present you can buy a 2 TB (terabyte) disk for less than 60 dollars, while 1 TB = 1000 billion bytes = 10^{12} bytes which can hold more than 1500 hours of CD-quality sound! However, in spite of the cheap storage space in computers the need to compress sound has remained because of the widely used downloading of audio and video via the internet. The internet communication speed has improved a lot but at the moment it is still limited to an average of about 15 Mbit/s or so. For audio alone, this is acceptable: downloading a 3-minute CD-quality audio file with this speed would take 3*60*1.4112/15 = 17 s. However, an uncompressed 3-minute video clip with audio, for example, contains 562 MB (Megabyte) so this would take 562*8/15 = 300 s or 5 minutes to download which is not acceptable. In addition, *streaming* of uncompressed video with audio (the data file is played while it is sent) would even not be possible: the downloading bit rate is lower than the playing bitrate of the video.

So, for video it is a must to compress the video signal and the pertaining audio, therefore, has to be compressed too. Even for audio alone, it is attractive to be able to record a lot of music and speech on a small portable device having a flash memory with limited capacity instead of a spinning disk, and to send and receive audio files almost immediately.

The subject of signal compression encompasses many areas (pictures, video, audio, data encryption, etc.) of which we are only interested in audio here. Even limited to audio data compression[1], about this field alone many books exist and it is beyond the scope of this book to describe all of the existing, often quite complicated, methods and algorithms. The only intention here is to explain the basic principles of some different methods to limit the number of bits to represent the original audio signal, and how to do the reverse: to reconstruct the original back again.

First of all, we have to distinguish *lossless* and *lossy* compression. The first category, as the name implies, are the methods to limit the number of bits in such a way that the

[1] In this section the term 'sound *data* compression' is used, as opposed to the term 'sound compression' which might cause confusion as the latter term is also used for the gradual limitation of the momentarily amplitude level of the sound *waveform*, for example to avoid overmodulation of a sound recorder by the peaks in the signal. This type of sound compression is usually performed by using a **limiter**, which is explained in section 27.3.

original can be reconstructed completely, without any difference. The second category allows for some differences in the reconstructed sounds which are not important or, if audio is involved, which are practically not audible by human ears.

As a digital representation of a sound section consists of a number of sample values, how can we limit the number of bits that can represent this bunch of values without loss of data? We know from section 17 that the amplitude of each sample is approximated by one of a fixed set of values. Each value of this set should be represented by a unique binary number. When all values are completely unrelated to each other lossless compression would not be possible. In that case the *probability* to occur is the same for each value. In practice, however, there is some *redundancy*: the values have some weak or strong relation to each other. As an example, fig. 24.1 shows a histogram for all amplitude steps of the spoken sentence as presented in fig. 22.1.1. You can see that the

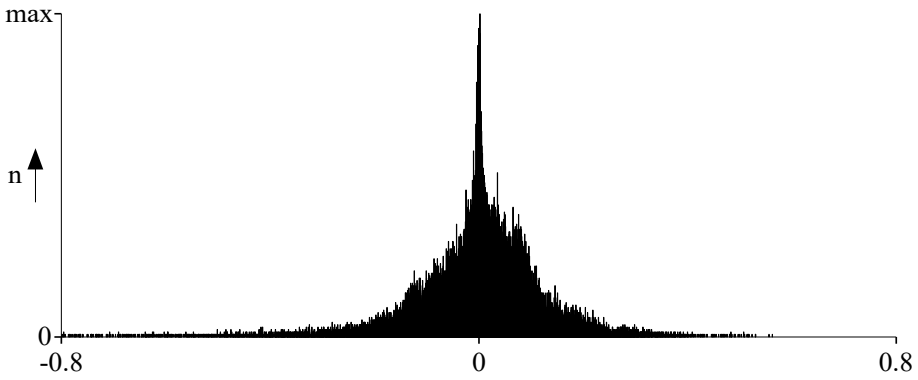


Fig. 24.1. Number of occurrences of amplitude values of all samples in the spoken fragment of fig. 22.1.1.

amplitude values nearest to zero occur by far the most frequently and that the frequencies of occurring of high amplitudes are very near zero. Now the idea is to apply a *variable length code*: fewer bits for amplitude values occurring at high frequencies and more bits for values occurring at low frequencies. Of course, there must exist some manner then to separate the individual codes from the sequence of bits of a coded data stream as the lengths of the individual codes vary all the time. The **Huffman encoding** offers a clever way to accomplish these tasks.

To understand the principle of this Huffman code a simple example: we will encode the lower-case characters of the alphabet and the space, which means that we must have at least 27 different *symbol* codes. Table 24.1 gives the relative frequencies of occurring of characters using this book as a reference, sorted from high to low frequency (scaled by a factor 2000).

sp	e	t	i	a	o	s	n	r	h	l	c	d	f	u	p	m	w	g	y	b	v	k	q	x	z	j
352	215	156	125	123	119	116	114	92	82	71	59	50	50	49	46	45	29	27	22	21	17	6	5	5	3	1

Table 24.1. Relative frequencies of lower case characters and space from this book (multiplied by 2000).

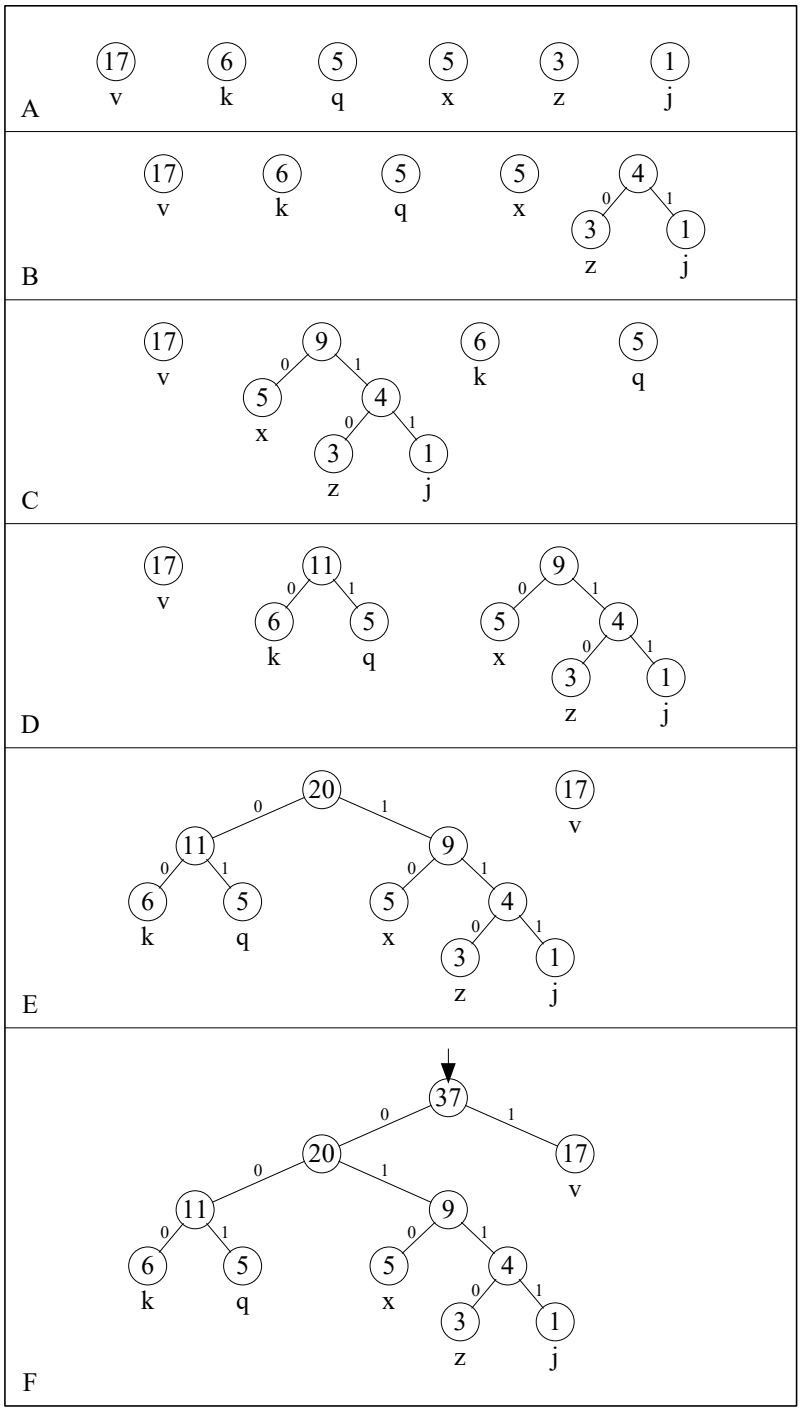


Fig. 24.2. First six steps to construct the Huffman encoding tree from the table 24.1.

The Huffman encoding constructs a tree consisting of only two-way nodes. Its leaves represent the characters. Each leaf has a unique position in the tree and a unique path to it via the nodes. The construction of the tree starts with the two lowest frequency characters. They are combined with a node. This combination will have a frequency of occurring which is the sum of the individual frequencies of the two characters. These two characters in the frequency table are replaced by the combination and the combination is put in the proper frequency position. Now the procedure is repeated and finally there remains only one node at the top (the root). In fig. 24.2 the construction of the tree from the six lowest frequency characters is shown.

From the sorted table of characters (panel A) the lowest two (z and j) are combined to the node so that it gets the frequency 4. From the new array (panel B) the two lowest frequency elements are combined and produce a node with frequency 9. After resorting the array, it looks like panel C. After three more steps the tree looks like panel F. The branches to the left are labeled as 0 and the branches to the right are labeled as 1. All texts existing of these six characters can now be encoded by the use of this tree. For example, the code of the sequence j k q v x z k v z x is:

01110000011010011000010110010

The procedure is repeated for the rest of the characters until the final root has been reached. As all leaves (the characters) exist only at the end of the paths, each character has its own unique path and thus unique binary code. The resulting tree is shown in fig. 24.3.

As you see, there is no separation between the individual codes of the characters. The Huffman coding, however, uses a variable length. Then, how can a Huffman code be decoded? The bits in a computer are stored in 8 bits or 16-bit chunks and there are no clues in the code to mark the starts or stops of the individual codes. Nevertheless, the decoding procedure is simple: handle the bits one at a time and follow the tree from the root. Each time a character has been reached, decoding of the next bits starts at the root again. As a challenge, you will be encouraged to decode the following sequence of 56 bits by using the tree of fig. 24.3:

10010111000010000111100010000000110001100001100100001000

If you made no mistake, you will discover that the message contains 14 characters (including the space). If the same message were encoded by using “normal” fixed-length codes we would need 5-bit numbers as the minimum number of bits necessary to represent one out of these 27 elements (there are only 16 numbers of 4 bits and 32 numbers of 5 bits). Formally, this can be expressed by:

$$b = \log_2(n) \tag{24.1}$$

where b is the minimum number of bits and n is the number of values in the set. For $n = 27$ the minimum number of bits is 4.75 so we would need numbers with 5 bits to represent all values individually. Thus, for the message involved we need 14 x 5 bits =

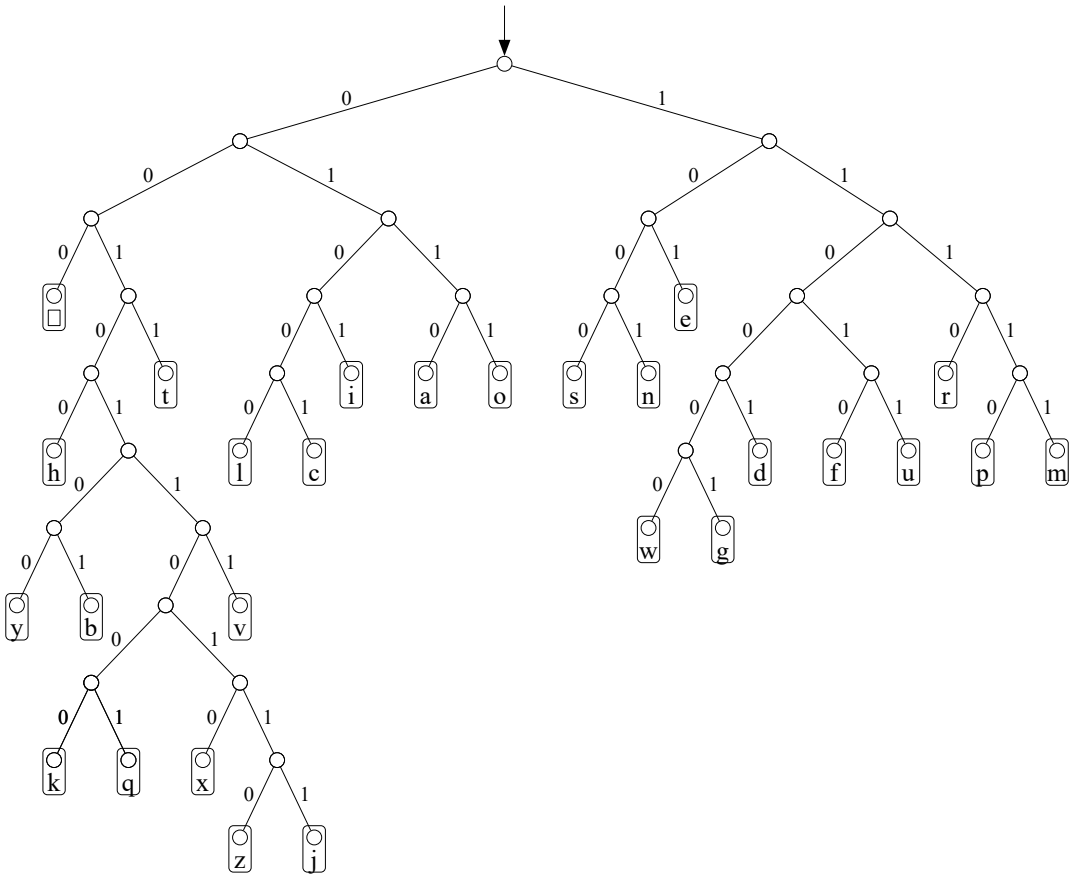


Fig. 24.3. Completed generation of Huffman tree from frequency table 24.1.

70 bits. The **compression ratio** for this message, therefore, is 56/70 or 0.8. Also, the **compression factor** is often used, being (70-56)/70 = 0.2.

In this example, the average number of bits per character is 56/14=4. It is possible to compute the theoretical number of bits needed to contain the information of each symbol from its frequency table:

$$b_i = \log_2\left(\frac{1}{p_i}\right) \tag{24.2}$$

Here is b_i the number of bits for the i^{th} symbol and p_i the probability of occurrence of the i^{th} symbol. This probability is equal to its relative frequency. If we divide each frequency in the table by the total sum we get the relative frequencies. The probability is a number between 0 (it never occurs and b_i is infinite) and 1 (it is the only element that occurs and b_i is 0). It is a well-known formula in information theory but the derivation of this formula falls outside the scope of this book. To find the theoretical

average number of bits of all symbols we must add all individual b_i's in the ratio of their relative frequencies, i.e. multiply each b_i by its p_i which leads to:

$$b_a = \sum_{i=1}^{m} p_i \cdot \log_2\left(\frac{1}{p_i}\right) \qquad (24.3)$$

where b_a is the *average* number of bits of all codes and m is the total number of different symbols. This formula, applied to the book text example of 27 symbols, renders a theoretical average of 4.1 bits/symbol. The practical average happens to be 4.14 bits/symbol. The practical average is always higher than the theoretical one because the Huffman code lengths do not consist of fractions of bits but have integer numbers of bits. Nevertheless, the Huffman codes approximate the theoretical average quite nearly for large sets of symbols[1].

Likewise, in case of sounds instead of texts, the frequency of occurring of digital amplitude values differs for all discrete amplitude levels, as you know from fig. 24.1. It will be clear then that also sound files can be compressed by Huffman encoding of the sample values. In the example of fig. 24.1 the number precision is 16 bits, which means that the whole range (from -1 to 1 pascal) consists of 65536 different values. In practice, the absolute peak (positive or negative) of a digital sound is kept below these boundaries and the sound of this example has a range of 40200 amplitude steps. From each of these steps the frequency of occurring has been measured. Using the formula 24.3 the theoretical number of bits turns out to be 12.6 which causes the compression factor to be 0.2125.

While Huffman encoding offers a completely lossless solution, the compression ratios are not very impressive. For texts, big improvements have been made by encoding *combinations of characters* and other methods to take advantage of the redundancy in texts. We will not focus further on text encoding as it falls outside the scope of this book. Besides, for sounds these methods are not very appropriate. Even the Huffman compression ratio found in our example above will generally approach a limit when the sound lengths increase. (For music sounds the distribution of probabilities of sample values will tend to a Gaussian of which the average variable-length code is only about 1.2 bit lower than the fixed-length code. On 16-bit precision sounds this means a compression factor of only 0.075.)

So, it is necessary to find out how redundancy in speech and music sounds can be used in a different way. One of the properties of sounds is the frequently occurring of intervals of absent signal or at least intervals having amplitudes that are below the first amplitude step from zero. In those cases, sequences of many samples with zero value occur. Instead of encoding sequences of many zeros, only one code for zero and the number of repetitions is encoded. Of course, for this **run length encoding** there must

[1] There are other coding systems that approach the theoretical average even closer, like *arithmetic encoding* which defines the **whole set** of symbols as only **one** fractional number that equals its probability of occurring. Generally, the number of digits of this number will be much greater than the number precision of computers can accept, so that the digits are distributed among many bytes. Details of this arithmetic encoding deviate too much from the subjects of this book.

be some strategy to discriminate between numbers of repetitions and the codes that are repeated, which will need some marking bits. In practice, when four or more elements occur in sequence, the run length encoding will already produce some compression. It will be evident that, especially in speech sounds, many zeros occur.

Another property that can be used for compression is that sample values are, to some extent, dependent of preceding sample values. In practice, big amplitude steps from one sample to the next one is quite exceptional so that it will be advantageous to encode the (mostly small) *differences* between consecutive samples instead of their actual values. (This works as a differentiator, as you may know from section 18.) As 'normal' digitizing is called PCM (Pulse Code Modulation, mentioned in section 17), encoding the differences is called **DPCM** (Differential PCM). Combining DPCM and Huffman encoding of our example sound yields a compression ratio of about 0.64 (compression factor = 36%). All encoding types described up to now are lossless: the originals can be reconstructed without any error.

All methods described so far take PCM samples as input. There are, however, methods that control the sampling itself, i.e. the analog to digital conversion (ADC), in different ways to limit the number of bits. Examples are the **µ-Law** or **A-Law** conversions which are based on the probability distribution of amplitudes of practical audio signals, as shown in fig. 24.1. The nearer to zero the amplitude values are, the more frequently they occur. In addition, a property of the human hearing is that low intensity signal components during the occurrence of high intensity components tend to be masked by the high intensities. Now, the idea is to apply a *variable amplitude step* depending on the magnitude of the signal. ((adapted step size)) For this reason, the formulas of the **µ-**

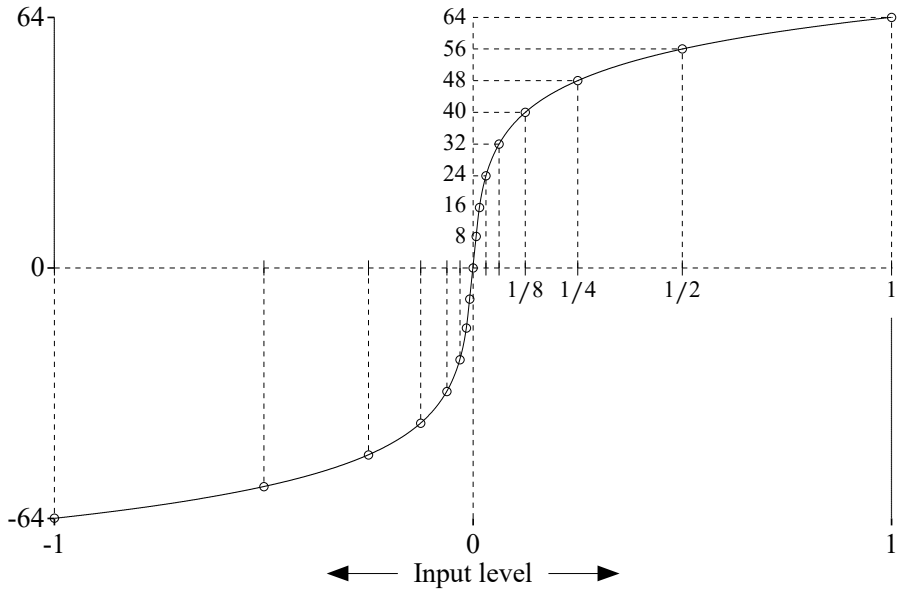


Fig. 24.4. AD conversion with variable step size according to A-Law function.

Law or **A-Law** conversions modify the analog input level according to a logarithmic relation.

The formula which modifies the analog input level according to the A-Law relation is:

$$f(x) = \frac{sgn(x)\cdot A\cdot|x|}{1+\ln(A)} \quad (0 \le x \le \frac{1}{A}) \tag{24.4a}$$

$$f(x) = \frac{sgn(x)\cdot(1+\ln(A|x|))}{1+\ln(A)} \quad \left(\frac{1}{A} \le |x| \le 1\right) \tag{24.4b}$$

Here represents x the analog input level. The constant A is 87.7 by convention. The values of $f(x)$ are quantized with the 'standard' PCM method. The total number of steps can be limited then without too much loss of quality. See fig. 24.4 for an explanation of its principle. In the figure, the A-Law converted output values are quantized into 128 discrete steps, i.e. into 7-bit numbers, as an example.

The **μ-Law** is a slightly different conversion which uses the formula:

$$f(x) = \frac{sgn(x)\cdot\ln(1+\mu\cdot|x|)}{\ln(1+\mu)} \quad (0 \le x \le 1) \tag{24.5}$$

Here the constant μ is set to 255 by convention. The two functions are almost identical, only in the vicinity of zero there is a small difference. While the μ-Law conversion is the standard for America and Japan, the A-Law conversion is the one for Europe mainly.

The principle of the *differential* PCM mentioned above can also be applied to the analog values: instead of the analog values themselves, the differences between analog values at regular adjacent time positions can be converted and quantized. In this case, the process is called **Delta Modulation** (DM). The number of bits applied for quantizing the differences can even be limited to *only one* provided that the time steps are sufficiently short, i.e. the sample frequency sufficiently high. For this 1-bit ADC, see the functional diagram in fig. 24.5 which shows one of the methods to realize a **delta-sigma** modulator. Here, the solid lines represent analog signals and the dotted lines represent digital signals. The analog input signal is compared with a reference voltage, coming from a capacitor (C). A capacitor (or condenser) can be seen as a device which can be loaded or unloaded to any voltage and hold it when the charging or discharging stops. When the input voltage is higher than the reference voltage, the comparator produces a digital '1' at its output. Reversely, when the input voltage is lower than the reference, the output is a digital '0'. The 'clock' produces pulses at regular time intervals with a high rate. At each clock pulse the comparator output is stored in the 1-bit data memory until a new clock pulse arrives. The 1-bit data stream controls a switch which charges the capacitor a small amount when the bit is 1 and discharges the capacitor the same amount when the bit is 0. When, after one or more clock periods, the capacitor has reached the input voltage, the comparator changes its

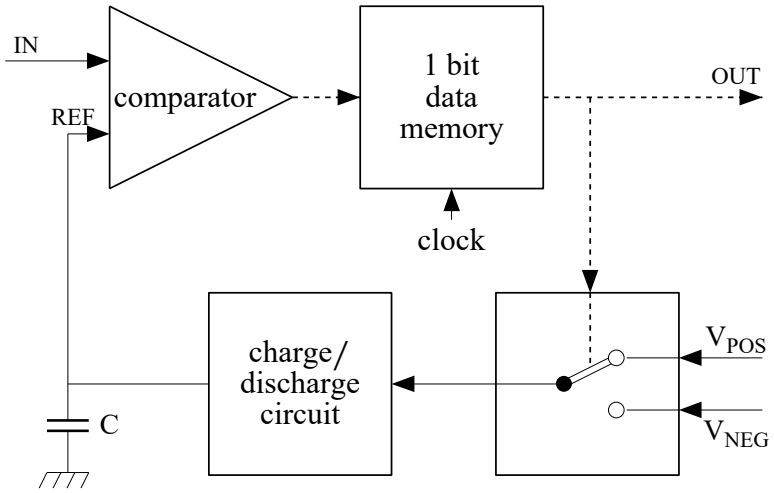


Fig. 24.5. AD conversion with delta-sigma modulator.

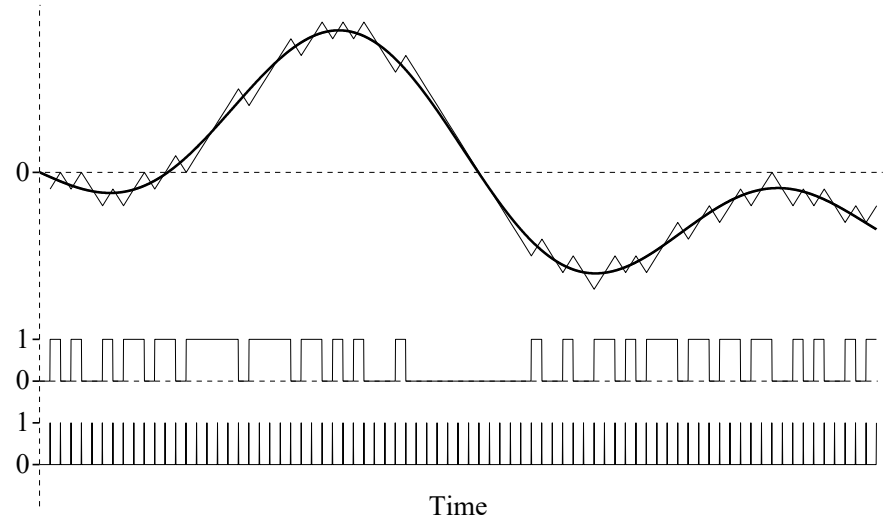


Fig. 24.6. Signals of delta-sigma modulator. Top: input signal (bold line), together with signal at capacitor (zigzag line). Center: 1-bit digital output stream. Bottom: clock pulses.

output and from the next period on, the capacitor voltage starts changing into the other direction. When the input remains zero, for example, the bit stream at the output changes at each clock period so that a continuous 1-0-1-0… sequence occurs at the output. The voltage at the capacitor then fluctuates with small values around zero while its mean is exactly zero. In fact, the voltage steps from the switch are *integrated*. The

Greek letter delta is often used to express a difference while the Greek sigma refers to addition. The ‘delta’ in the name refers to the difference between input and reference, while ‘sigma’ refers to the addition of a part of the difference to the reference voltage. Fig. 24.6 shows the wave forms at different positions in the delta-sigma modulator. For achieving some accuracy, the 1-bit data stream should have a very high sample frequency, in practice about several megahertz. This, in principle, makes the system suitable for feeding the data stream directly into serial inputs, like digital audio disks, recording onto SD cards and USB inputs. Nevertheless, the recording devices (pc’s or digital audio recorders) are designed in such a way that the data stream is converted into, for example, 16 or 24-bit numbers, as there is always the need to process the signal (like filtering and labeling) before it is recorded onto the medium.

To convert the 1-bit data stream back into an analog signal, obviously the lower part of fig. 24.5 can be used. It converts the data stream to the voltage at the capacitor which is an approximation of the original signal. The approximation can be made as accurate as needed by applying a sufficiently high clock frequency, i.e. sample frequency, and an appropriate charging/discharging rate.

The sigma part of the delta-sigma modulator can be seen as an *adaptation* of the differences. Therefore, the counterpart of the delta-sigma modulator in the digital domain is Adaptive Differential PCM (**ADPCM**). Here the magnitudes of the differences are decreased by taking into account the value or values from one or more earlier differences. This reduces the number of bits necessary to encode the differences. Using a number of earlier values to ‘predict’ the next difference is similar to the LPC (Linear Predictive Coding) mechanism as mentioned in section 20.

(Of course, the adaptation of the differences by using more than one earlier differences can also be done in the analog domain of the delta-sigma modulator.)

All these Delta Modulation and ADPCM compression types, however, are not lossless: decreasing the number of bits for encoding the differences, together with the size of the time step (the length of the clock period) cause that some deviations from the original will occur. (In principle, all analog to digital convertors are not lossless: the discrete steps always mean an approximation of the original continuous analog signal, as explained in section 17 about sampling.)

A great deal of the many *lossy* compression methods is based on the properties of the human hearing. As an example, when a strong frequency component and a weak one in the spectral neighborhood are present at the same time, the weaker component is **masked** by the stronger one. Even when two frequency components in a sound have equal intensities, the perception is as if only one (mean) frequency is present, provided that the frequency difference falls within the so-called **critical band** (cb). This critical band depends on the frequency and intensity but as a rule of thumb it can be stated that it is about 19% of its central frequency when this frequency is greater than about 1000 Hz. Below 1000 Hz the cb is found to be rather constant (about 100 Hz). The ability of

frequency discrimination (or **selectivity**) should not be confused with the ability to discriminate between frequencies of *separate* sounds. For example, when tuning musical instruments by listening to them in succession, people are able to hear differences of about 0.2 % or even less.

Masking occurs also in the time domain: a signal with a weak intensity within about 10 ms or less from the end of a signal with high intensity is masked as well.

To make use of the properties of the human sound perception for compression can be quite complicated. Many compression methods have in common that the frequency range is divided into bands, about the size of the cb. The time signals of them are then processed separately to achieve an efficient individual compression strategy for each band. Because of the huge number of different compression systems developed in the course of years we will suffice here to mention only a few principles of the **mp3** coding which offers a great compression ratio (usually 1/11) and a very satisfying fidelity of the sound quality. Mp3 is short for MPEG-1 Layer III where MPEG stands for Motion Picture Expert Group. It was originally developed by the Fraunhofer Institute in Munich, Germany, and in a later stage adopted by the MPEG as the sound part of their video compression format.

Although the details of the mp3 coding are very complicated, a few principles can be explained from the basic knowledge of the subjects earlier discussed in the book. First of all, a *filter bank* divides the spectrum of the PCM sampled sound to be compressed into a number of bands in order to be able to process the outputs of these filters individually. Then all limitations of the human hearing can be used optimally for each frequency band. The output signals from the filters are processed in limited intervals of time: the frames. From each frame, a *transform into the frequency domain* is performed. On these spectral components, various compression processes are applied by limitation of the quantizing steps for the spectral data, using the masking properties as mentioned above. For example, when spectral components are within the range of a specific cb, the number of the quantization steps for the components can be limited to the number which keeps the quantization noise just inaudible. The remaining stream of limited sample values, in turn, are compressed by Huffman coding to limit the data stream even more. The result is converted into parts of a fixed number of bits to represent the data in a decodable stream of numbers.

As you will remember from the first sections, the transformation into the frequency domain means that the result contains components for sines and cosines, or components for magnitudes and phases. The transform used here, however, uses only cosine components, hence it is called the Discrete Cosine Transform (DCT). How is it possible then that the original can be reconstructed from only cosine components? The DCT uses a simple trick: firstly, from the frame of samples, make a mirror by reverse it in time, then concatenate it with the original so that a symmetrical time function emerges. From section 6 you may remember that a symmetrical sound like this is an *even function* which has a Fourier transform with contains only cosine components. So, an inverse

Fourier transform produces the symmetrical function back again. Extraction of the non-mirrored half of this function returns the original. See fig. 24.7 for an explanation. (Of course, it is not necessary to calculate the cosine components for the whole

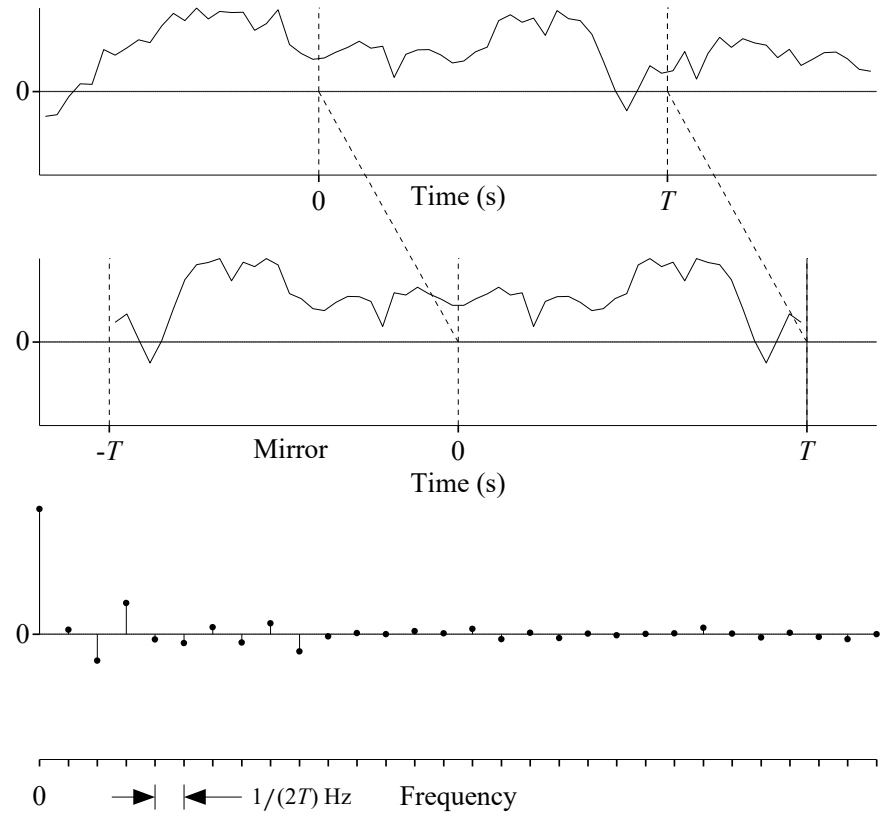


Fig. 24.7. Discrete Cosine Transform. Top: arbitrary selection of signal part. Center: addition of mirror of part doubles the 'period' length. Bottom: DFT of symmetrical sound contains only cosines.

symmetrical function. The multiplications of the negative time part render exactly the same results as those of the positive time part so that the number of multiplications within the transform can be halved.) As a consequence, the DCT uses the *double length* of the 'period', causing the 'harmonics' to be 1/(2T) Hz apart, i.e. the number of components is doubled, compared with the DFT. Whereas the DFT uses *n* cosine values and *n* sine values, the DCT uses 2*n* cosine values.

What is the use, then, of the DCT whereas the number of spectral components is equal to the DFT's? In fig. 24.7 you can see that most of the higher frequency components have quite low energy, which is a general property of spectra of practical sounds. Now the DCT, stronger than the DFT, concentrates almost all of the spectral energy in a small number of components, a property which can be used for limitation of the necessary number of bits for the quantization without too much distortion of the signal.

But the DCT offers an additional advantage. As has been explained in section 13, the boundaries of the selected signal parts effect their spectra. So, when the sound must be reconstructed from the spectral components, it is necessary to apply a suitable window but also *overlapping* of the sequential windowed intervals. This overlapping 'costs' extra encoding bits. Now, with the DCT it is possible to apply 50 % overlapping *without the cost of extra bits*. This overlapping version of the DCT is called the Modified Discrete Cosine Transform (MDCT). As the explanation of this trick is quite

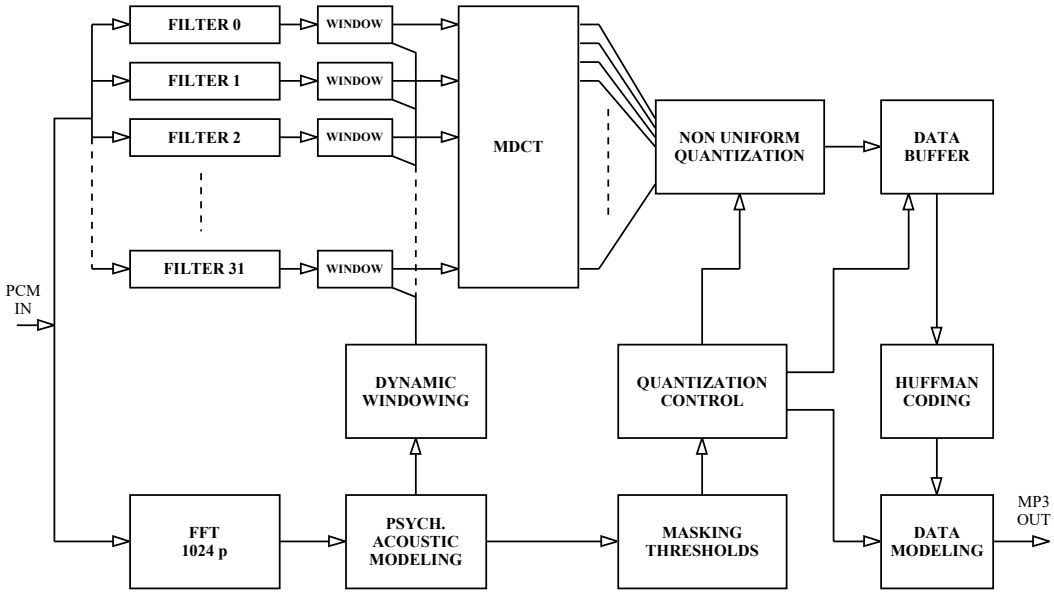


Fig. 24.8. Functional diagram of MP3 coding of one audio channel.

complicated we will omit it here. Besides, there exist many books about the details of the MDCT. This clever modification makes it possible to reconstruct the original from the consecutive spectral data without any loss. Nevertheless, the MP3 coding success is mainly due to the possibilities to limit the precision of the spectral components, using the various masking properties of the human hearing, so that the compression achieved occurs practically without audible effects. Fig. 24.8 shows a (simplified) functional diagram of the MP3 coding system. Consecutive parts of the PCM are shifted into buffers of 512 positions (the frames) and filtered by a filter bank of 32 bands. To each output a window is applied with variable length, dependent on the spectral contents of the input signal (fast varying input needs short windows for good time resolution, slower variations need longer windows for good frequency resolution). This spectral information is provided for by the system block 'FFT' which is a parallel fast Fourier transform of 1024 samples into frequency points. From this FFT the window lengths are controlled. The windowed filter outputs are transformed into the frequency domain by the block 'MDCT'. All outputs of the MDCT are quantized with individual controlled accuracy, defined by the blocks 'QUANTIZATION CONTROL' and 'MASKING THRESHOLDS' which, in turn, are controlled by the block 'PSYCH.ACOUSTIC MODELING' that contains the human hearing masking properties. In fact, the main contribution to the compression is performed by the quantization control. The

compression is partly improved by the block 'NON UNIFORM QUANTIZATION' which raises the input values to the ¾ power before quantization. This improves the signal-to-noise ratio for small sample values. (The mechanism operates similar to the A-law conversion of fig. 24.4, only the non-linear part of the curve occurs mainly at sample values near zero.) The resulting data are distributed into suitable parts by the block 'DATA BUFFER' and then further compressed by Huffman encoding. The block 'DATA MODELING' combines the encoded sound data with 'metadata' which defines all parameters that can be chosen (the amount of data reduction, the number of channels, the frame lengths used, etc.).

One of the tricks not mentioned yet is making use of the resemblance between the two channels of a stereo sound. Two different systems can be selected. In the '**MS stereo**' mode (Middle/Side) the *sum* of the two channels (the middle) is coded as one signal and the *difference* (the side) is coded as the second signal. The benefit is that often the samples of the side information have low values which can be quantized with less bits. The '**Intensity stereo**' mode makes use of a property of the human hearing: within critical bands in the range higher than about 2000 Hz the perception of the stereo effect depends mainly on the difference between the channel intensities and less of the phase differences. In these parts of the stereo sound the signals of the channels are summed and the individual local intensities are coded only as scale factors so that the individual intensities of the two channels can be reconstructed during the decoding.

The functional diagram of fig. 24.8 suggests the use of many separate 'units' that all perform their own signal processing. In reality, the processing is applied on the data stream of the PCM input and usually occurs in real time (similar to the digital filters in section 18). There is only a delay between input and output. So, all filtering, transforming, quantizing, etc. occurs at the same rate as the input sampling. No need to say that the required computing power of computers that process MP3 coding and decoding is quite high.

Obviously, the subject of signal data compression is very broad and only a very small part is described here. If you are interested in more details then there is an almost unlimited number of books and papers available on this subject.

One final remark seems appropriate: although the quality of the reconstruction of MP3 compressed sound may seem quite high, it still is a *lossy* compression. So, when you want to make sound recordings for research projects (speech analysis, investigating perception of sounds, masking experiments, testing sound equipment, etc.), it may be wise to avoid sound data compression at all and use PCM with sufficient sample frequency and number precision (i.e. at least 44100Hz and 16 bits, respectively). After all, the lossy compression methods all make use of the 'weaknesses of the human hearing and, possibly, the analysis outputs of decoded compressed sounds may display unwanted side effects.

25. Noise suppression

In section 15 is explained that at the ‘ends’ of every electrical conductor a noise voltage is present, due to the random behavior of the free electrons in the conducting material. When a microphone converts the very small air pressure fluctuations of sound into an electrical signal, the achieved voltage fluctuations are also very small. After amplification to a sufficient level for listening, the unwanted noise is amplified too which may result in a poor sound quality w.r.t. the S/N (signal-to-noise) ratio.

When low-intensity sounds must be recorded, the S/N ratio may become unacceptable. In the earlier time of analog recording systems (tapes, vinyl records, compact cassettes) the noise suppression of the recording medium was a big issue as, mostly, this noise had a higher intensity than the electronic amplifier’s noise. Now we have digital recording media (CD’s, DVD’s, hard disks, flash memories, etc.), the noise of the recording equipment can practically be ‘defined’ by the chosen accuracy of the digital conversion electronics, by setting the sampling frequency and number precision. The ‘weak point’ w.r.t. noise, however, is the microphone amplifier which has to contain some ‘analog’ electronics. (You may have heard of the existence of ‘digital microphones‘, suggesting some digital way of conversion of the sound pressure variations into bits but, alas, the word refers to a part of the microphone electronics which produces the digital output, i.e. the analog to digital convertor.) No matter which system is invented in the future, the S/N ratio of the transducer (the microphone ‘element’) is restricted by its thermal noise (as mentioned in section 15).

Thus the ‘bottle neck’ still remaining is the microphone with (pre-) amplifier which means that the suppression of noise after the recording is made is still required sometimes. In addition, there is the need to suppress the noise of old recordings or recover damaged ones.

Although many ingenious noise suppression systems have been developed in the sound processing history, there is no need to explain these any more as most of them are developed to improve the limited S/N ratio of the old recording media. From the remaining general methods to suppress noise from recorded sound, one principle has been already mentioned in section 16 about correlation. A drawback of that method is that the ratio of the spectral components of the wanted sound get changed which generally results in great differences with the original sounds.

One of the best working noise suppression systems is the **spectral subtraction** method. This method is based on the general property of audio signals from ‘natural’ sources (speech, music, animal sounds, etc.): the spectra of these sounds consist of a limited number of narrow peaks of relatively high intensity, whereas the noise spectrum consists of the whole range of frequencies of much weaker intensities. So, between the relatively strong components of the signal, a ‘floor’ of almost all other frequencies of low intensity exists. Now, the idea is, from all frequency components, to subtract a small part of their amplitudes and to convert the modified spectrum back to sound again.

In practice, the spectrum of the noise will not be perfectly flat, so that the size of the part to be subtracted from a frequency component is determined by the corresponding frequency 'bin' of the noise spectrum. Generally, the noise spectrum can be obtained from the Fourier transform of a part of the sound recording where the signal is absent (i.e. in a pause). Of course, some noise spectral components will coincide with those of the signal but subtracting a small part of their amplitudes will hardly influence the contributions of these components to the signal. Only when the signal-to-noise (S/N) ratio is quite low, the distortion of the signal may become unacceptable. Nevertheless,

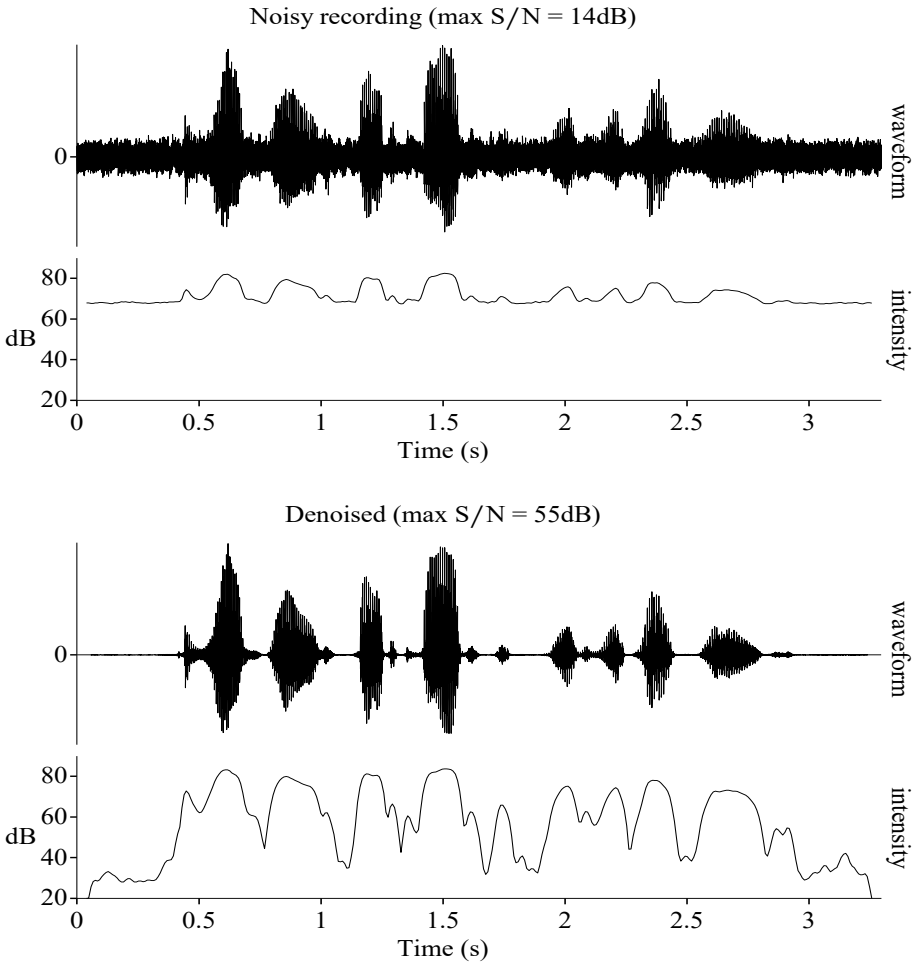


Fig. 25.1. Denoising a recorded sentence by Spectral Subtraction.

the amount of noise suppression that can be obtained will often be quite satisfactory. You can run DEMO 25.1 which plays a spoken sentence with added background noise of relatively high intensity (the S/N ratio is only 6 dB), followed by the same sentence denoised by a simple spectral suppression algorithm. As you can hear, the S/N ratio has been improved substantially (about 40 dB, as you can see in the intensity contours and in fig. 25.1). In the following example, the initial S/N ratio is only 10 dB and the improvement is 38 dB. But now you can hear that the signal quality has deteriorated a lot.

Here again, a similar warning as in the end of section 24 has been formulated, seems appropriate: when research experiments are involved, you should realize that using this spectral suppression in order to improve bad sound recordings will cause differences with the originals. In particular, the ratio of the intensities of voiced parts and unvoiced parts (i.e. consonants) of recorded speech will be changed to some extent (the noisy speech parts may be attenuated a bit). When the noise to be suppressed is not too high in intensity, the difference can be quite acceptable, however.

26. Musical sounds

For the basic sound processing techniques described so far, the speech signal has been mainly used as examples because of the variety of its properties. The only limitation is that the speech signal is **monophonic**, which means that only one sound source is active at the same time (apart from the breezy or noisy voiced speech of the *fricatives*, where the turbulence of air acts as a second sound source besides the vocal folds). When two or more sound sources are active at the same time, the relation of the separate sounds of the sources becomes important. Therefore, a section about the basic structure of musical sounds is included.

Of course, music can be monophonic as well, like solo singing, or playing monophonic instruments (like most wind instruments, one-string instruments, or the *Theremin*).[1] But even monophonic music is not comparable with speech: the frequencies of the musical tones and their sequence are bound to specific relations. In particular, the frequency ratios of sequential musical tones and their rhythm are much more strictly bounded to rules than frequencies and durations of the sequential speech sound parts. In addition, the sound of a certain instrument (say, a clarinet) producing a certain tone can be very different from the same tone (i.e. the same fundamental frequency) from another instrument (say, a harpsichord) as the number and intensities of the harmonics or overtones are different. (Remember: overtone n is the same as harmonic n+1.)

When two tones are played in succession, and the frequency of the second tone is two times the frequency of the first, the interval (which is one *octave*) sounds 'in tune'. When the frequency is doubled again, the perceptual impression of the new interval is the same: the frequency (the *pitch*) sounds one octave higher. Each redoubling of the frequency sounds one octave higher, as the human perception of phenomena in general behaves in a relative way, according to *Weber's law*, which was mentioned already in section 2. But that is only one part of the explanation. The frequency ratio of one octave is equal to 200 %. When the interval is much smaller, say, 1 %, the difference between the two tones can still be perceived very well. In fact, most people can hear frequency differences of about 0.1 % when the frequencies fall in the most frequency-sensitive range of the human hearing (2000 Hz, for example) and the variations are occurring within short time intervals. This frequency difference corresponds to only 1/693 octave![2] You can run DEMO 26.1 to test your own best frequency selectivity. It is a rough test to estimate the **just noticeable difference** (JND) of frequency shifts. You will hear sets of two sinusoidal tones. One of the tones has a steady frequency, the other is frequency modulated by a sine wave of 5 Hz. The sequence of the two tones occurs at random. The frequency deviation (the *modulation depth*) is changed at random, in a range from hardly audible to clearly fluctuating. After each stimulus you can respond by selecting FIRST, SECOND, or DON'T KNOW, to indicate which of

---

1 An electronic 'organ' of which its tone frequency and intensity can be varied by positioning the hands in the neighborhood of an 'antenna'.

2 The ratio at 0.1 % increase is 1.001. Raised to the power of 693 produces 1.999.

the tones was not steady. After 48 stimuli, the number of correct responses for each frequency deviation is displayed in a graph. You can select one of the frequencies 200, 500, 1000 and 2000 Hz at the start of the test. You will discover that your selectivity is the poorest when you selected 200 Hz and, probably, the best when 2000 Hz was selected. (When interpreting the results, one must realize that the displayed frequency modulation percentages are only half of the total shift between maxima and minima.)

Because of this generally high accuracy of frequency shift perception, the tuning of musical instruments and the pitch (fundamental frequency) of singing sounds has to be very precise. This is especially important when two or more tones sound *at the same time*: the sound is **polyphonic**.[1]

As mentioned above, if the frequency ratio is 2, the interval is exactly one octave. Naturally, to create music, some tones must be defined within the range of one octave to divide it into smaller parts. If all parts are to be perceived as 'equal' steps, each tone should have a frequency increment which is a constant *percentage* of the preceding tone frequency. In other words, the frequencies must have a logarithmic relation, just like the octave steps of 2, 4, 8, 16, etc. which are increments of 100 %. For example, if one chooses to make eight steps within an octave, each tone frequency should be the preceding tone frequency multiplied by $2^{(1/8)} = 1.0905$, which means increments of about 9 %. All tone systems with logarithmic relation are called **equal temperament** (ET) systems. For monotonic music any exponential distribution of tones could be adapted.

However, in music, the *relation* of the pitches of tone sequences or tones played at the same time (chords) is extremely important w.r.t. 'harmony' or **consonance**. The more the combination of tones is perceived as 'pleasant', or 'agreeable', the higher the consonance. For two simultaneously single sinusoids the perception of consonance as a function of their frequency difference can be measured, as has been done by Plomp and Levelt in 1965 [10]. The averaged result looks like fig. 26.1 which shows a stylized curve of the consonance as a function of frequency ratio. The vertical scale is arbitrary. The curve starts with zero difference: the consonance is maximum as there is only one sinusoid. When the frequency difference is increased, repeated 'beats' can be heard: one tone with varying amplitude in the rhythm of the frequency difference. When the frequency difference increases further, the beats occur more rapidly, until this beat frequency has become so high that the individual beats cannot be perceived anymore and the sound becomes harsh, unpleasant, rough. When the difference increases further, the sound becomes audible as two separate tones played together. So, the dip in the

[1] This high frequency selectivity of humans has nothing to do with the 'absolute pitch' ability which means that the pitch of a tone can be determined (indicated as a musical note) *without any reference*. This ability is very rare (figures as 1 in 10000 people are reported and even then, fully erasing the memory of earlier given tones in the tests can be very tricky) and it would mean either a tremendous frequency memory of hours of days, or a mechanic 'aberration' like a sharp resonance peak in the mechanics of the ear.

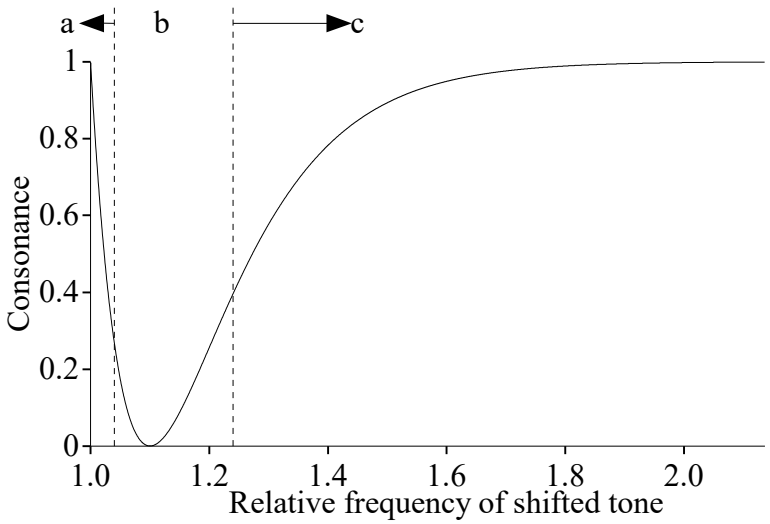


Fig. 26.1. Consonance of two simultaneous sinusoids as function of their frequency difference, ref. Plomp and Levelt. The steady tone is 500 Hz. In the area 'a' the difference is heard as amplitude 'beats'; in the area 'b' the sound is perceived as 'rough' or 'harsh'; in the area 'c' the two components are audible as separate tones.

curve represents the point of maximum **dissonance** (the opposite of consonance). The position of this maximum dissonance is not solely determined by an absolute frequency *difference* (as was stated earlier by Helmholtz) but depends also on the frequency of the tones (although not linearly proportional to them either). The graph of fig. 26.1 is valid for a lower tone of 500 Hz. For higher frequencies, the consonance dip shifts to lower relative values. According to Plomp et al. the position of the maximum dissonance is not completely independent of the frequency but seems to be *a function of the critical bandwidth at the frequencies concerned*. (See section 24 for some remarks about the critical band concept.)

Remarkable is that these curves have no local maxima at positions where the frequency ratios are 'musical' intervals like 3/2 (a 'perfect fifth') or 5/4 (a 'major third'), etc., whereas many people would expect peaks at these 'harmonic' interval positions.[1] This means that, *for simultaneous sinusoidal tones*, the musical frequency ratios *play no part* in the consonance curve. (Plomp and Levelt minimized the possible influence of the listening subjects' musical education on the perception results.)

When produced by musical instruments, however, tone intervals with ratios like 3/2, 5/4, 4/3, etc. generally sound much more harmonious or pleasant than other ratios. So, this must be caused by the *presence of harmonics* of the tones of the musical instruments or voices. For example, the 2nd harmonic of a tone of 630 Hz falls very near the 3rd harmonic of a tone of 425 Hz and may contribute relatively much to the total

[1] This musical numbering for notes stems from the commonly used system of 7 'whole' tones per octave. The terms will be explained later.

dissonance. When the 2nd tone has a frequency of 420 Hz, these harmonics coincide so that their contribution to the total dissonance is zero.

Now, Plomp and Levelt have applied their consonance curves to generated tones with 6 harmonics each. For each shift they added all individual *dissonances* (curves like fig.26.1 upside down) of all combinations of harmonics. Now the resulting dissonance curve showed dips at 6/5, 5/4, 4/3, 3/2, 5/3 and 2/1 times the frequency of the lower tone, in other words, peaks at these positions in the consonance curve! In his publication, William A. Sethares [12] included a program to produce the dissonance curve from a selectable number of harmonics and specified amplitudes of the individual harmonics, based on the consonance curves of Plomp and Levelt. Applying his program to an example of a 'musical instrument tone' with the number of harmonics set to 7 and gradual decaying amplitudes with a factor 0.8, leads to fig. 26.2 which displays the resulting dissonance curve. The curve exposes 9 dips, i.e. 9 tone frequencies at relative low dissonant positions. Although the position of the dip in fig. 26.1 depends on the frequency, the dissonance dips in fig. 26.2 *remain in position* as they are the exact coincidences of harmonics frequencies.[1]

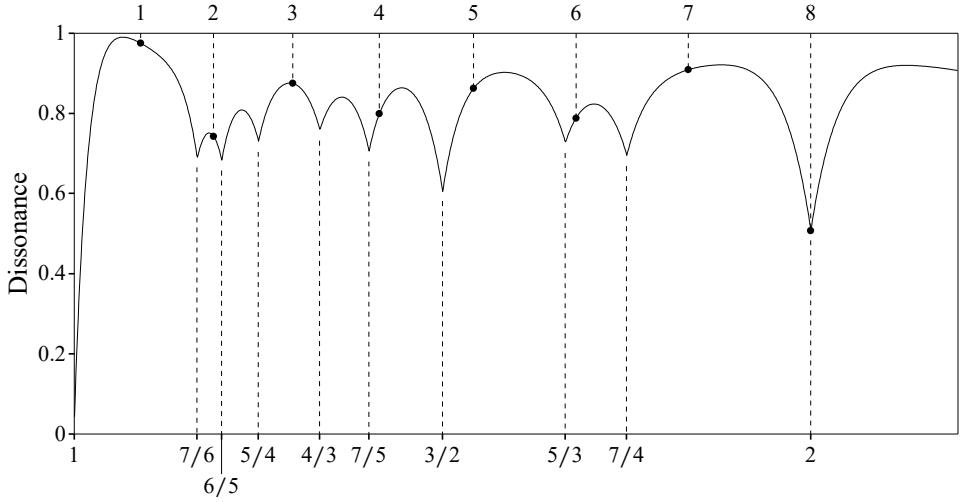


Fig. 26.2. Dissonance of two simultaneous tones with 7 harmonics each, as function of their fundamental frequency difference, calculated from the application of the consonance curve of fig. 26.1 to al harmonics combinations. Based on algorithm from W.A Sethares. The positions of an 8-tone ET example are marked at the top.

The choice of 7 harmonics for the test signal is not very critical; this number is lower than the number of harmonics of most music instruments but the intensities of the

[1] When the tones are produced by two independent instruments, there is no fixed phase relation as in case of the fundamental and the second harmonic of a tone of one instrument. Only when the ratios remain perfectly constant, the phase difference is constant, and no 'beats' can be heard, which is never strictly true in practice. This 'imperfection' when the same note is sung by a group of people is known as 'choir effect'.

higher harmonics of almost all acoustic music instruments decay rapidly and can be ignored for this subject.

When the positions of the 8-tone division of the octave of the example mentioned before are marked in the dissonance curve of fig. 26.2, we can see that many tones of this *ET8 tuning system* fall at unfavorable positions. Then, why not choose the exact positions of the dips of fig. 26.2 for the tones within the octave? There are two reasons why this would work out not very well.

The first one is that the perceptual 'distance' of the frequency steps of these tones is far from constant. Although there may be no serious objections against a slight unevenness of the logarithmic frequency steps, the difference, for example, between the first step

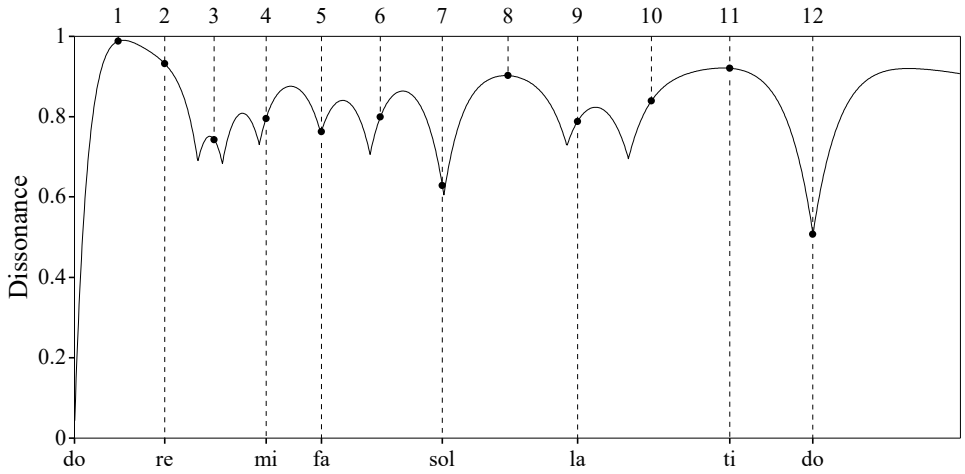


Fig. 26.3. ET12 tones (top) and major scale tones (bottom) placed in dissonance curve. The 7 major scale tones form a more consonant subset of the 12 ET tones.

(7/6 = 17 % step) and the next one ((6/5)/(7/6) = 2.9 % step) is rather unacceptable. Even when the 3rd position is excluded, the step percentages vary from 4.2 % to 17 % which is still very unsatisfying. In addition, the construction of some acoustical music instruments becomes more complicated.

The second reason is of general nature: when a musical piece is played, the base tone (the **key**) must be at the position '1' in this system. In other words: the music instrument, adapted to this tuning system cannot be played at a different key (i.e. **transposed**) because any other selection than '1' will cause different steps or frequency ratios so that the musical melody alters a lot and, what's more, will sound awfully 'out of tune' as all chords will be 'assembled' with different steps and become completely different in character. The only real solution to this problem is a pure logarithmic distribution of tones within the octave: *the equal temperament* tone systems as mentioned above. These are the only systems where the music can be played in any key.

So, as consonance is concerned, the ET tones should be as near as possible to the dips in fig. 26.2. When the positions of the tones of a 12-tone equal temperament system (which is the modern Western 'standard' tuning system) are marked in the curve of fig. 26.2, as is done in fig. 26.3, we can see that some of these tones fall at positions quite near the minimum dissonance points, in particular the 'important' ratios 3/2 and 4/3. In this respect, the 12-tone ET system is a much better choice than the 8-tone example mentioned above. It is possible to calculate the mean consonance per tone for some ET systems with different numbers of tones in the octave, using the dissonance function of fig. 26.2. After calculating this mean for the systems ET6 to ET15, it turns out that the ET12 system has the highest mean consonance (although the differences are small). So, it is not very surprising that the 12-tone equal temperament tuning has become a world standard, in spite of the many alternative tone systems which emerged in the history of the different cultures.

Pythagoras discovered already the importance of the most consonant ratio within the octave: 3/2 (the 'perfect fifth', in musical terms). He calculated the frequency ratios of repetitive increments of one perfect fifth, which means multiplication by 3/2 for each next tone. After the 12th step, the tone is practically the same again, only 7 octaves higher:

$$\left(\frac{3}{2}\right)^{12} = 129.75 \cong 2^7 \qquad (26.1)$$

In this way, each multiplication results in a *different relative position* within next octaves. See fig. 26.4 for a visual explanation. The numbers of the face of a clock serves as markers within the octave, with the '12' as the base tone. When the steps are dimensioned as perceptually equal, each step would be the multiplication of the latest one with $2^{(1/12)} = 1.059$, i.e. the ET12 system. Then a 'fifth' falls at the '7' because $2^{(7/12)}$ is about 3/2. the next increment with a 'fifth' falls about the '2' in the next octave. The next positions on the face, after 'scaling down' to the same octave, are: 9, 4, 11, 6, 1, 8, 3, 10, 5, 12, 7. This famous **circle of fifths** is the basis of the Pythagorean temperament. The small inequality of formula 26.1 is called the "comma of Pythagoras", which is only 1.36 % in the 7th octave (or 0.113 % of the 3/2 ratio itself). Instead of percentages, in music

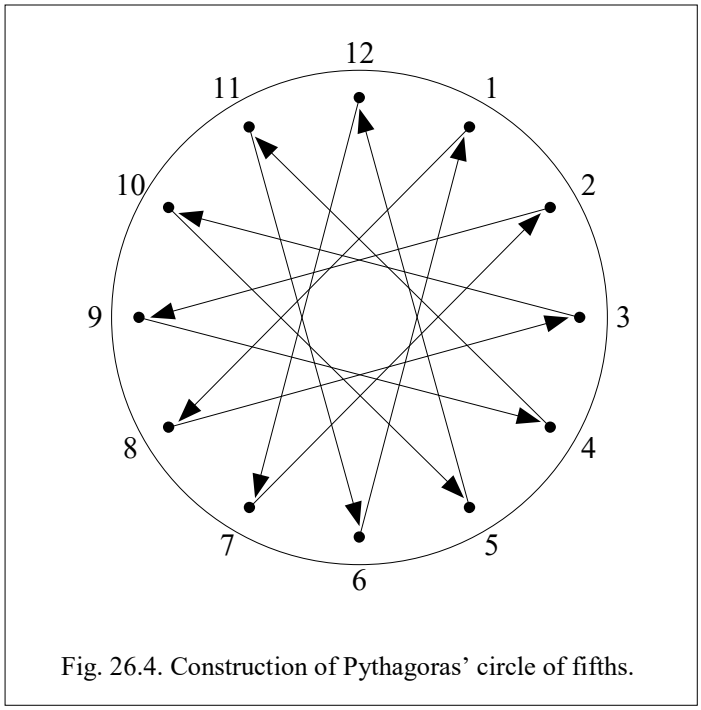


Fig. 26.4. Construction of Pythagoras' circle of fifths.

subjects it is customary to divide each semitone of the octave into 100 parts, called **cents**, which divides the octave logarithmically into 1200 parts. So, this Pythagorean deviation of 0.113 % corresponds with 1.96 cents (1200 * log2(1.00113 ≈ 1.96).

Because of this deviation, the range of this repetitive Pythagorean tuning was limited to one octave (with the 7th step as the center of the 12-tone range): the *12-tone Pythagorean tuning*. The correspondingly ranked tones of the higher octaves were simply multiplications by powers of 2 and the tones of the lower octaves divisions by powers of 2. In the much later applied 'extended Pythagorean tuning' this limitation was cancelled by correction of this comma and the result is exactly the same as the 12 tone Equal Temperament, which was developed as late as 1584!

In the Western 12-tone music tuning system the tone increment is called a **half tone** or **semitone** which implies that there must exist **whole tones** as well. These whole tones form a subset of the 12 half tones. In music notation (where the 'C' is the base note) the whole note range is notated as: C, D, E, F, G, A and B: the *major scale*, mostly learnt in childhood as "do re mi fa sol la ti do". See the white circles in fig. 26.5 for the names of these major scale notes according to the commonly used musical notation. When the positions of these whole tones are placed in the dissonance curve, as has been done in fig. 26.3, we can see that their positions approximate the 'ideal' ones better than the total range of all ET12 tones. In fact, these half tones were often omitted in early music, probably due to their lower consonance. In musical notation, therefore, these 'tones in between' were represented as whole notes with symbols added for modifications: the *sharps* and *flats* (# and b) for half tones higher and half tones lower, respectively. Consequently, a 'C#' is the same note as the 'Db', the 'Eb' is the 'D#', the 'F#' is the 'Gb', etc. (This is only true for the ET12 system: originally, there were small differences, depending on the key in which the music piece was written.) Fig. 26.5 displays the musical names of the places of all 12 tones in the octave. You can see that the interval naming of the 7-tone major scale omits the '0': it starts with '1' for the base and goes on to '1' again for the octave (which is also indicated as '8'). The commonly used historical originated musical notation system with these sharps and flats, together with the use of 5-line bars to accommodate a

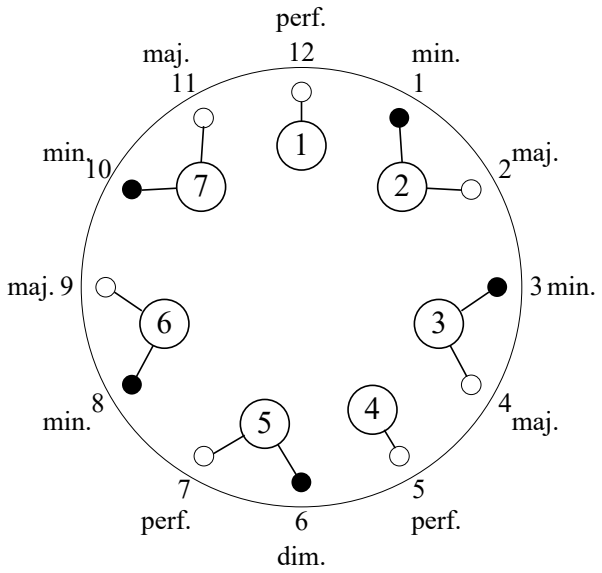


Fig. 26.5. Naming of musical notes. The **maj**or and **perf**ect notes form the 7 notes of the major scale: the white keys on the piano when the base (1) is the C. The **min**or and **dim**inished notes then are the black keys.

subset of only 7 unequal ‘whole tone’ distances, seems very inadequate for a tone system with 12 half notes in the octave. Nevertheless, this notation is used all over the world and educated musicians are able to read it at an amazing speed.

Although being a relatively consonant system, the ET12 contains a lot of compromises, as you can see in figs. 26.2 and 26.3. Even the ‘important’ interval of 3/2 (the perfect fifth) is not exact (the deviation is 0.133 % as mentioned above). When playing cords, which consist of 3 or more tones played simultaneously, these remaining dissonances of the ET12 system can become annoying. Therefore, a lot of alternative tuning systems have been designed in the music history. Of course, any system that is not an ET (equal tempered) system has the problems as described above (the uneven tone distances and the impossibility to transpose). Many musical instruments, however, are able to produce any pitch in their total range, not limited to the discrete steps of the ET12 or other tuning system. For example, with violins, cellos, and trombones (and the singing voice!) any tone within the octave can be produced so that many of the played intervals can be exactly matched with the consonant positions in the octave. Even the fixed tones of some instruments can be varied a small amount by adjusting the manner of blowing or pushing aside the strings. (Naturally, with this type of instruments or the voice, it is possible to produce tones with vibrato: a frequency modulation like you could have heard in the DEMO 26.1. This effect may also be ‘misused’ to mask the disability to produce perfectly tuned tones.)

A description of the numerous alternative tuning systems falls beyond the purpose of this book, only the **just tuning** is mentioned here because it is a tuning system which applies frequency ratios with simple whole numbers like 4/3, 5/4, 3/2 and so on, which are the most consonant ratios. Nevertheless, this just tuning also suffers from dissonants

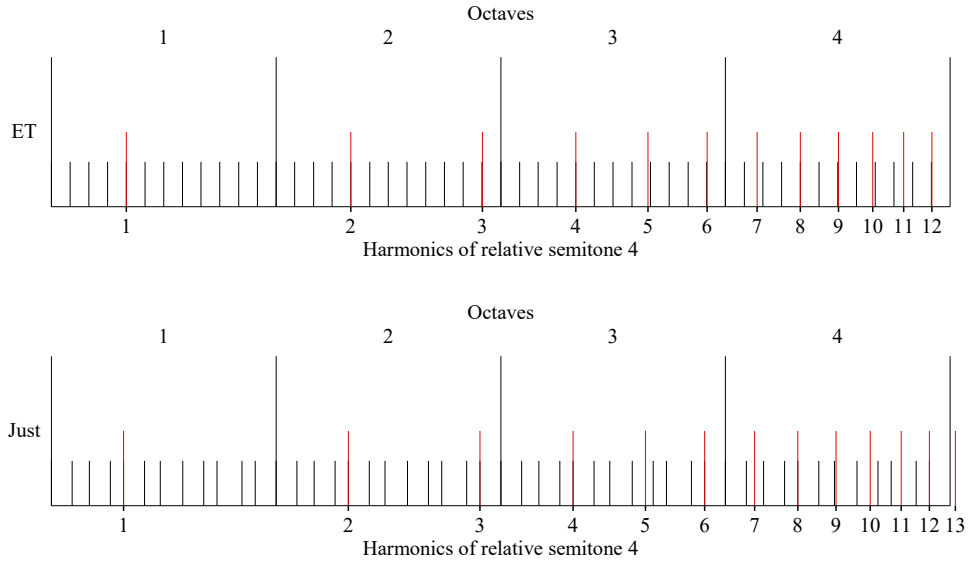


Fig. 26.6. Harmonics of 4th semitone of octave (red lines) mapped on logarithmic frequency scale with marked ET tones (top) and just tuned tones (bottom). Generally, the higher the harmonic, the less it fits with the tone system, which applies for all tones.

formed by harmonics and tones in other octaves, as shown in fig. 26.6, where the positions of harmonics of a tone can fall outside the tone positions in the higher octaves. The figure compares the ET12 system with the just tuning system for the 4th semitone (the major 3rd) as an example. Although the fit of the just tuning and the harmonics in the higher octaves in this example seems poorer than that of the ET, the opposite will be true for some other semitones in the octave. The *mean* dissonance for all tones is somewhat lower for just tuning than for ET. As you can see, there must be always harmonic frequencies which fall somewhere between the nominal tuning tones and thus will cause dissonances. Obviously, this is true for all tuning systems as a linear series (the harmonics) will never fit to a logarithmic series (the frequency perception). Naturally, when a musical instrument produces many and strong overtones (e.g. the harpsichord), the probability of dissonances to occur when chords are played is much higher than when the overtone amplitudes at higher frequencies weaken out very fast (e.g. in case of the oboe).

A great number of music instrument types have basic sound sources of which, globally, the harmonics weaken in intensity as their frequencies increase, e.g. strings and 'reeds'. The resonance box or sound board or other resonating part of the instrument amplifies certain harmonics and attenuates others, in analogy with the human voice: the vocal tract filters the harmonics of the source: the vibration of the vocal folds, see section 19. Usually, the 'peaks' of the filter function encompass several spectral lines (harmonics), as is the case with pianos, violins, cellos, acoustic guitars, harps, etc. The *timbre* (i.e. the total envelope formed by the spectral lines) is characteristic for each instrument. The analogy with the human voice applies very well for these instruments, although the spectral maxima (or *formants*) of these instruments generally cannot be varied while the vocal tract spectral peaks can be varied in a great range.

Other types of instruments function differently: usually, the resonant part together with the source functions as a kind of oscillator by feedback of the resonating part to the source (most wind instruments). Here, the dimensions of the resonant part determine the exact frequency of the tone produced (i.e. an organ pipe, of which its resonance principle is mentioned in the subject of the vocal tract in section 19). To generate different tones, either the effective pipe length is altered (flutes, trombones, clarinets, etc.), or there are separate pipes for each tone (pipe organ, pan flute). Thus, these instruments work not at all like the human voice and the dimensions are crucial for the tuning, while these are not for instruments of the former category.

Occasionally, however, a filter 'peak' of the resonance box of a violin or cello can be quite narrow so that the energy at the resonance frequency can be relatively high when the fundamental or a higher harmonic of the source comes in the neighborhood of the resonance frequency. Due to the acoustic coupling of the string and the resonance box (a *Helmholtz resonator*) there is some mutual influence between the oscillations of the string and those of the resonance box. A small difference between their frequencies may result in strong 'beats': the sound can be almost zero in the minima. When the repeated beats occur in a particular rate, the result does people think about the howling

of a wolf: the **wolf tones**. (Similar small differences of tones and harmonics in tuning systems as mentioned above are sometimes called ‘wolf intervals’.) There are some methods to avoid these wolf tones, which could easily occur even to high quality instruments, mainly based on damping the string energy at these wolf frequencies.

As you may know from section 19 about tubes which are open at one end (almost all pipe instruments), is that the resonances occur at odd multiples of ¼ wavelengths which means that only odd-numbered harmonics emerge. To create tones with different characters with the pipe organ, more pipes are ‘switched on’ which are tuned to higher harmonics to assemble a certain spectral envelope. In this way, the ‘timbre’ of the sound can be greatly altered. This principle of “Fourier synthesis” is also applied in Hammond organs and, of course, electronic synthesizers.

27. Miscellaneous practical subjects

27.1. Praat's Spectrum

As mentioned already in section 15 about noise, the number of different spectral components is infinite, theoretically. For a component to have any sense, it must have some magnitude. Because the power of the signal cannot be infinitely high, the components are not regarded as individual sinusoidal waves but seen as spectral *density* values, defined in Pa/Hz or dB/Hz units. Obviously, the same applies to continuous spectra. This is the reason that Praat's spectra (and those of some other sound analysis programs) are *density spectra*.[1]

This implies that, in case of pure sinusoids in a sound, the magnitudes of the spectral components *depend on the time length* of the sound. This may seem counter-intuitive: when the sound contains a sinusoidal wave of, say, an amplitude of 1Pa, you may expect a spectral component with a magnitude of 1 Pa (or 91 dB on a log scale, see section 22.2 about intensity). However, this expectation assumes that the components are independent of the length of the sound, in other words, *mean values*. Recalling the computation of the Fourier components as presented in section 6, the components are cross-correlation factors of the sound with a cosine and a sine of 1 Pa amplitude. When the means of the results are computed, the spectrum is an amplitude spectrum. When the total sum *or integral* is taken (which obviously is the mean multiplied with the duration, see the box SIGNAL COMPARISON in section 6), the spectrum is a density spectrum. So, only when the sound duration is 1 s, the spectral density magnitude is equal to the amplitude. Otherwise the spectral values of the cosine and sine components should be divided by the duration in seconds to convert the density spectrum to an equivalent amplitude spectrum. (As you may know from section 5 the combined amplitude is equal to the square root of the sum of the squares of the cosine and sine amplitudes.)

Why so many words about such a simple matter? As a conclusion, the display of a spectrum of a sine wave with an amplitude of 1/2 Pa and a duration of 1 second should be equal to a spectrum of a sine wave with an amplitude of 1 Pa and a duration of 1/2 second, as their densities are the same, right? Alas! It is not true. See fig. 27.1.1 for an example with a sinusoidal wave with a whole number of periods in both cases so that there exists only one spectral component. Although the densities are the same, the difference between the two displayed spectrum values is 3 dB! The cause of this is that the dB values *refer to power*, as already explained in section 2. They do not represent power but refer to the power which is generated as a *result* of the amplitude.

---

[1] In the display the discrete spectral values of the bins are connected with straight line segments and no 'spectral lines' to the horizontal axis are drawn, resulting in a continuous curve.

Consequently, for the spectral density the amplitude part is valued differently from that of the time part. So, the factor 2 increase of the amplitude part means 6 dB and the factor 2 decrease of the time part means -3 dB (which is -10*log(2). The result is an increase of 3 dB. The spectral magnitudes of a density spectrum *in dBs* are proportional to the duration of the time signal but also proportional to the *square* of the component amplitudes.

This is a consequence of the convention to express amplitudes (or SPL's) in power dB's. It was adopted by telephone engineers in earlier days and everybody is accustomed to this habit. In my opinion, seen from a fundamentally viewpoint, this was not a very good idea. I would prefer pure logarithmic values of ratios of amplitude, SPL, velocity, etc., independent of possible *effects* in other areas (power, energy). This preference may be understood a bit from this description about the spectral density concept. Anyway, there is no choice: we have to use the spectral representation in power dBs for adaption to the rest of the world.

So, if you want to derive the amplitude spectrum from the display of the density spectrum in Praat, the number of dBs to subtract is equal to 10*log(duration).)

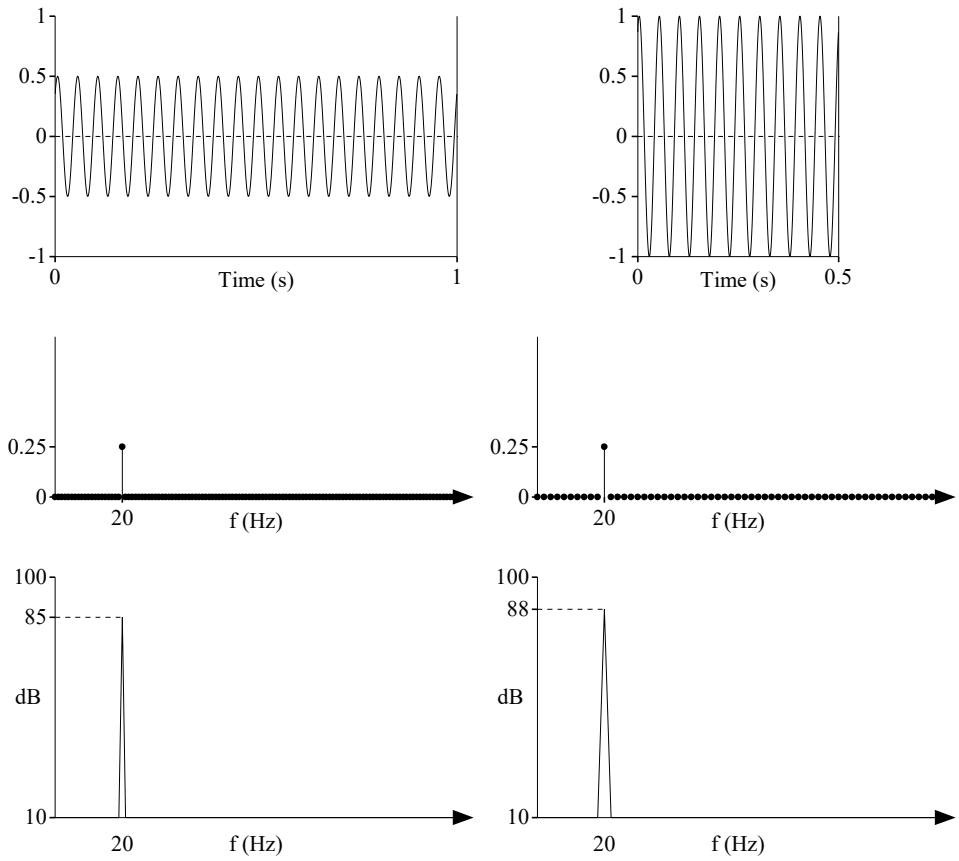


Fig. 27.1.1. Top: waveform parts with integer number of periods. Center: spectral densities are equal. Bottom: display of spectra in dB's differ 3 dB.

Then the linear value is 10^(dBnum/20)*20 µPa as explained in section 2. This linear value is the *rms (root mean square) value*, as the reference of 20 µPa is also defined as its rms value. To get the amplitude, the linear value must be multiplied by √2 as already explained in section 5. To put it all in one formula we get after some manipulations:

$$A = 2 \cdot \sqrt{\frac{2}{T}} \cdot 10^{\left(\frac{\#dB}{20} - 5\right)} \qquad (27.1.1)$$

where A is the amplitude of the spectral component, T the signal duration in s, and *#dB* the value to read from Praat's spectrum display.

In Praat, the underlying spectral data are contained in the Spectrum *object*, listed in the window "Praat objects". The Spectrum object values are *complex* (see appendix II.3 for the complex representation of Fourier components) which means that it comprises the phase information. The Spectrum object consists of a matrix with two rows and columns for each frequency bin. Basically, the first row contains the cross-correlation factors of the signal with *cosines* of amplitude 1 and the second row the cc factors with *sines* of amplitude 1. (Because of the conventions when dealing with complex representations, the numbers in the sines row are opposite in sign. Therefore, fig. 11.5 in section 11 can only represent the Praat Spectrum object when the red graphs are inverted.)

The spectral magnitude which emerges when the component frequency equals the signal frequency is not equal to but half the amplitude of the component in the signal. This stems from the fact that for the cc factors the time functions are multiplied, and multiplication of two sine waves with the same frequency and phase will produce a sine wave (with the frequency doubled) with amplitude ½ A_1A_2 where A_1 and A_2 are the individual amplitudes. One of them (the component with which the time signal is cross-correlated) is always 1 which causes the result always to be half the 'real' spectral value of the signal. A reverse Fourier transform by reconstruction of the signal from the frequency components should render the same amplitudes as in the original so that the cc components should be multiplied by 2. Thus, multiplication of both the cosine and sine components by 2 doubles the magnitude and would reconstruct the original signal exactly when all were summed. In Praat, this scaling is not done for the Spectrum object (as it is complex, containing also negative frequencies, see appendix II.3) but the calculation of the dB's in the display compensates for this.

As mentioned in part A, the spectrum in Praat can be produced either by using the DFT (Discrete Fourier Transform) or the FFT (Fast Fourier Transform). The DFT is the 'standard' Fourier transform as mentioned in section 6 and all spectral properties described so far are valid for the DFT. For the FFT, which performs much faster when the sounds durations are longer than some tens of seconds or so, it is required that the sound to transform consists of a number of samples that is a whole power of 2. Because, in general, this is not the case, some samples should be added before the

start or after the end of the sound. (Selection of a part of the sound such that the contained number of samples is a power of 2 would also do but is seldom used for reasons of selection flexibility. For sounds of which their parameters remain quite constant so that the length of the selection is not critical, however, this procedure would offer some advantages, as may become clear from the following.) Because there is no data to fill-in, zero value samples are added (*zero padding*).

A disadvantage of the zero padding to accommodate the FFT is that the consequences for the spectrum can be serious: when a 500 Hz sine sound, for example, has a duration of 0.5 s, the DFT shows a perfect single point (which you would expect because of the number of periods being a whole number), whereas the FFT shows strong *side lobes* over a very wide area, the nearest only about 15 dB or so lower than the center, see fig. 27.1.2. The same occurs at all currently used nominal sample frequencies. It will be no surprise after reading section 13 about windows and time-insertion: the addition of time causes to emerge spectral 'samples' from the underlying sinc spectrum of the rectangular 'window' with length 0.5 s. Of course, this is a theoretical example: a 'clean' DFT spectrum occurs only when an integer number of exactly the same periods is selected. In practice this is highly improbable. So, windowing the sound prior to the transform should be the 'standard' procedure[1] and for FFT it is absolutely necessary.

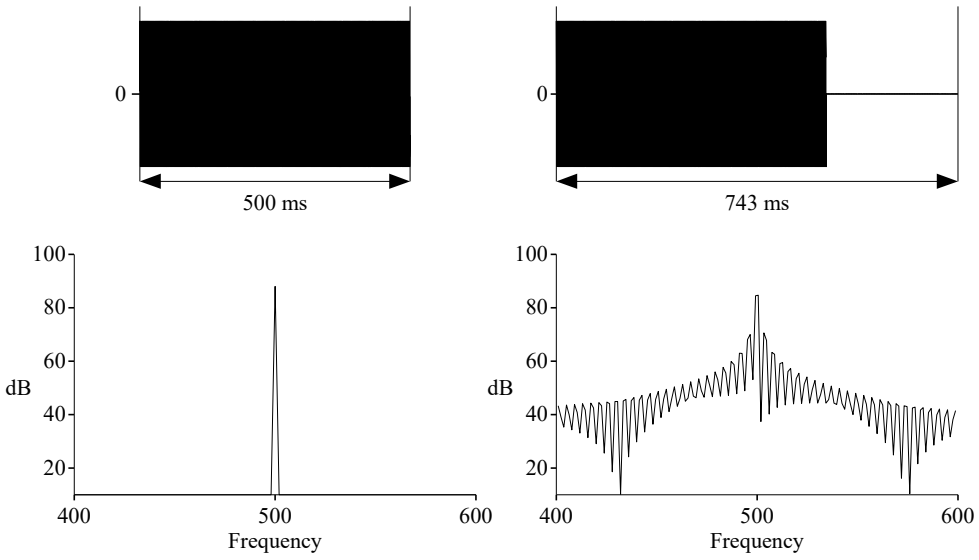


Fig. 27.1.2. Bottom: zoomed-in part of spectra of an integer number of periods of a sine wave of 500 Hz. Left: DFT. Right: FFT. Top: the corresponding waveforms of the reverse Fourier transforms.

[1] The main reason to window the signal in advance, as mentioned in section 17 about sampling, is that windowing greatly suppresses the *spectral leakage* that occurs when the time interval is not equal to an integer number of signal periods, which in general is the case.

However, even if signals are windowed before taking the FFT, the zero padding can still ‘distort’ the spectrum severely as the next example shows in fig. 27.1.3. In the left column the time function is 0.64 seconds of a sine wave of 200 Hz, sampled at

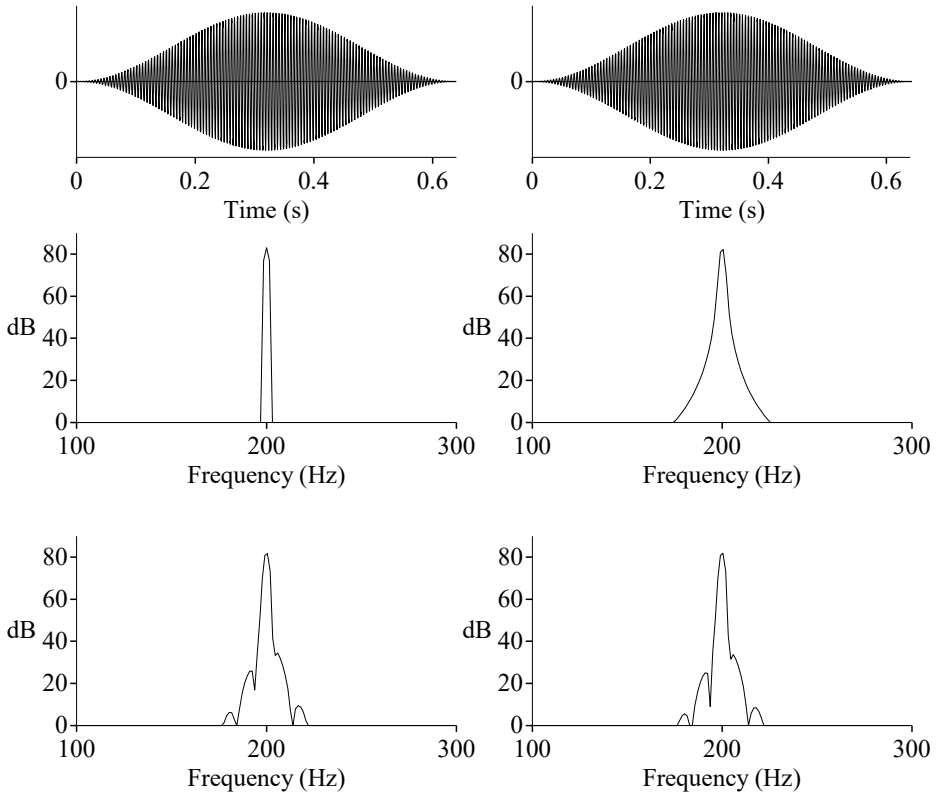


Fig. 27.1.3. DFT (center) and FFT (bottom) of a Hann-windowed sine wave of 200 Hz. Left: integer number of periods; Right: not an integer number of periods.

44100 Hz. The sound is windowed by a Hann window, which has a moderate side lobe suppression. The ‘zoomed-in’ spectrum part 100…300 Hz shows the zero-padding effect.

Even when the number of signal periods is not an integer, as displayed in the right column, the spectral distortion of the DFT is more acceptable than that of the FFT spectrum. (Although this is not always the case for several other types of windows with moderate side lobe suppression.) Of course, when a better window is used, like the ‘Gaussian2’ in Praat, the DFT and FFT spectra are practically the same.

An additional disadvantage of the FFT spectrum with zero padding is that, in general, the inverse FFT produces back a sound longer than the original, which usually is not desired. (See the example in fig. 27.1.2.) In practice, the amount of added time is unknown: it can vary from 0 (pure luck!) to the length of the signal minus one sample time.

Finally, the time insertion of the FFT causes a level inaccuracy. Because the spectral density is inverse proportional to the duration of the time signal, which can vary from the original T to almost $2T$, the difference between the spectral dB's due to the varying amount of added time runs from 0 to -3 dB. See fig. 27.1.4 for an example of these two extrema which occur when the number of samples crosses a power-of-two boundary. (Sine of 200Hz, sampled with 44.1kHz, Gaussian2-windowed). As a consequence, the relation between dB's in the density spectrum and the linear amplitudes according to formula 27.1.1 is only valid for the DFT and not exactly for the FFT. (Although it is possible to correct the dB values automatically, this is not applied in Praat.)

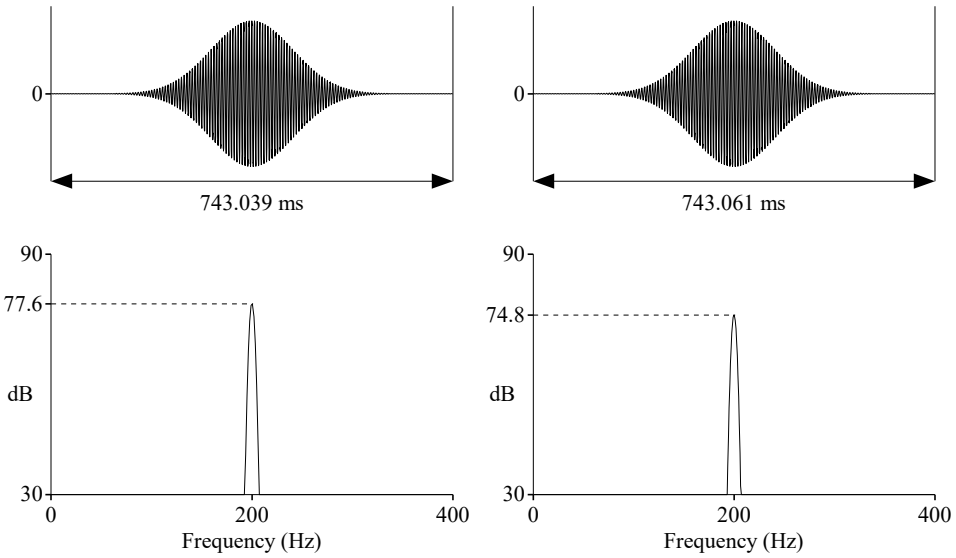


Fig. 27.1.4. When the number of samples exceeds the boundary of 2^n the time insertion of the FFT jumps from 0 to almost the length of the time interval: here the number of samples increases from 32768 (left) to 32769 (right). The spectral value drops almost 3 dB.

The FFT was developed in earlier times when the huge number of multiplications in the computer for performing a DFT took a lot of time. Now you should better avoid the FFT for all the reasons mentioned above, as the speed of the modern pc's is sufficiently high for fast computation of "short time" spectra.

It may be clear now that the selection of the time interval duration so that the remaining number is a power of 2 instead of inserting zeros, as mentioned above, would eliminate all disadvantages of the zero padding. Naturally, this can only be applied when the spectral properties to detect are all contained in the selected piece of signal, and when there are no other restrictions about time selection (as, for example, selection of an integer number of signal F_0 periods). In addition, the exact time intervals to select should then always be calculated from the sampling frequency.

Finally, to bear in mind the calibration remark in section 3 about the SPL (sound pressure level) in Praat: although the dB axis suggests a calibration of SPL with 0 dB referencing to the hearing threshold, there exists no calibration because the spectral values are derived from the sound, which is not calibrated in the first place (which is also explained in the Praat manual).

27.2. Frequency range

From section 17 about sampling we know that, in practice, the sampling frequency should be somewhat higher than twice the highest frequency that has to be processed. As, in many cases, the number of bits to represent the signal digitally has to be limited due to quality restrictions of the equipment or speed of transport via networks, the question arises: what frequency range is adequately? And how is it defined? Because there are no abrupt boundaries of frequency ranges of microphones, filters, amplifiers, etc., the frequency range is usually specified within an amplitude variation of +3 dB and –3 dB.

About the desired frequency range of audio equipment, a lot has been said and written. Especially, the 'audio purists' sometimes claim that a maximum frequency like 25 kHz or even more is 'absolutely necessary', although only young children sometimes have an upper hearing threshold of 18 or 20 kHz. The majority of people have a much lower frequency threshold like, say, 12 or 15 kHz. Nevertheless, the 'high end' sector of audio equipment often boasts about specifications of 96 kHz sampling frequencies. Apart from the idiocy of this (we are no dolphins or bats to perceive a range like 45 kHz or so), the 48 kHz or 44.1 kHz sampling applied in most equipment offers a sufficiently high upper boundary frequency which is well above the range of the human hearing.

Besides, a very high frequency range means a greater chance that a sensitive input of a (pre)amplifier picks up some high frequency harmonics from *switching power supplies*: present in many types of electronic devices, or light dimming units. The result can often be heard as disturbing humming or cracking noises, especially when the input leads of the audio equipment are poorly *shielded*.[1]

In addition, the range of most microphones, the transducers of nearly all sounds to electrical voltages, is also limited to about 20 kHz as well and most high-quality microphone pre-amplifiers (built-in as part of the devices or as a separate unit) have a low-pass filter to avoid very high frequency components to reach the rest of the audio electronics chain. Finally, the transducers which transform the voltages back to sound again, the loudspeakers or headphones, all have their own limits concerning the frequency range which rarely go beyond 20 kHz.

As for the *low frequency* boundary, a similar exaggeration can be found among audio

[1] In every electric conductor (wire) many very weak voltages can emerge by the 'antenna' effect or capacitive coupling with the leads of the electricity net. This effect depends on the frequencies: the higher the frequency, the stronger these 'spurious' voltages. *Shielding* means embedding the sensitive leads in a metal enclosure which is connected to a constant zero level in the device which works similar to the *Faraday cage*. It is an effective way to suppress the spurious voltages.

purists. Technically, it would be possible to set the lower threshold at 0 Hz when precautions are made in the (analog) electronics to stabilize the DC component (the zero level). For normal audio processing, however, it makes no sense to process frequencies lower than about 20 Hz as this is the normalized lower threshold of humans. When very low frequencies could be produced at all (by special types of loudspeakers only) these components, with sufficient intensity, could only be perceived as some vibration of your belly or fluttering of the legs of your pants. Also, musical instruments have fundamental frequencies seldom lower than 32 Hz.[1]

But, why *limit* the frequency range at the low end? If it is technically no problem to design amplifiers and microphones which function from 0 Hz on, why not make it a standard feature? There are several reasons not to do that:

1. All electronic amplifiers produce a small DC level at the output, even when the input signal is zero (the *zero offset*). The audio chain as a whole consists of a cascading of a number of amplifying units (stages) which means that in case of a range from 0 Hz, the small DC error voltage at the input stage can be amplified substantially. The result is that a certain 'DC error' at the output cannot be avoided and, obviously, it limits the dynamic amplitude range. When outputs like this are coupled with an input of, for example, a power amplifier with loudspeakers, the DC error causes the speaker *cone* to shift a certain distance from its neutral position. Because of this asymmetry, the maximum power which can be produced in the air will also be limited. So, blocking this DC component is often applied to minimize and stabilize the levels of the audio devices at zero input. (In the program Praat, the DC error level at the input of the sound card or sound unit of the computer can be eliminated very simple by choosing the option: "Subtract mean" which, as it says, computes the mean of the whole signal and subtracts this value from all sample values. This is always advisable to do prior to any signal analysis procedure as some DC level can disturb measurements as intensity, power, rms, etc.)

2. The human hearing sensitivity to very low frequencies is quite low compared to the midrange around, say, 1000 Hz. Microphone recordings might contain high intensities of these low frequency components while, when listening to the recorded sound, these are not audible at all. For example, thumping sounds from footsteps on wooden floors can be so strong that the 'zero line' in the waveform 'jumps' from zero up and down over large parts of the complete amplitude range. Even when recordings are carried out in a special 'sound proof' booth, these very low frequency noises are almost as strong as outside the booth because the sound isolation of the booth cannot be made very effective for these frequencies. When analyzing these sounds, the undesired influence of this 'zero line' movements on the analysis outputs can be

[1] Low fundamental frequencies can still be perceived while the electronic equipment (i.e. the loudspeakers or phones) cannot produce them. This is caused by the presence of harmonics. The repetition rate of the F_0 period does not mean that the F_0 component itself needs to be present (the term *missing fundamental* is sometimes used). This was mentioned also in section 4 about Fourier series.

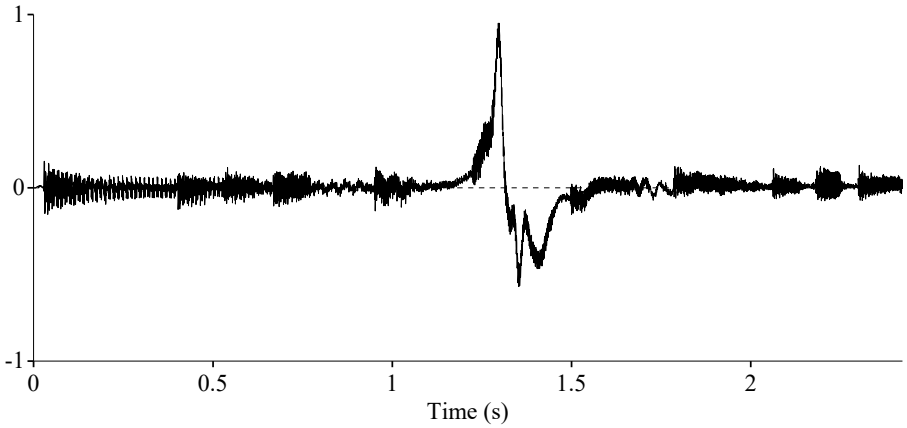


Fig. 27.2.1. A few seconds of a music recording which contains a low frequency 'thump' sound. The noise is not audible, despite its high amplitude (run DEMO 27.2).

substantial. You can run DEMO 27.2 for an example of a waveform of a recording of a music segment with this kind of low frequency noise (see also fig. 27.2.1). The thumping noise is hardly audible. The actual acoustical intensity of the thump must have been even higher because of the attenuation of these low frequencies by the microphone and the processing audio equipment used. The amount of shift of the 'zero line' can sometimes even be so great that the amplitude peaks of the wanted signal are limited to the maximum possible amplitude of the audio device ('*clipped*'). Another example: the *popping* noises that emerge when one is speaking too close to the microphone can be so strong that the wanted sound is 'pushed away' completely. The next section described the effects of this clipping more detailed. Obviously, the very low frequencies should be attenuated sufficiently in the first stage of the microphone amplifier to prevent distortion in the next stages. In practice, however, this is often not done, as you may have experienced sometimes when the voice of someone addressing an audience in a room is amplified by the *public-address system* used.

3. Low frequency components of machine noise, like air conditioning or fan noise, are often not heard, even while they can be relatively strong compared with speech or music sounds intensities, because people are accustomed to these constant intensity sounds in such a way that they are not aware of them anymore. Afterwards, when listening to the recording, this noise is noticed and can be very annoying. This noise does not disturb the 'zero line' substantially but the quality of the recordings might be too poor for analysis or presentation.

In this light it seems wise to suggest sound signal researchers to attenuate the very low frequencies *during the recording* of the sound material. If that is not possible (any more) then they should have their recorded sounds high-pass filtered, prior to whatever analysis is to be carried out. A cross-over frequency of, say, 80 Hz would be a practical value, for speech but *also for music*. The order of the low-pass filter should be at least

two, otherwise the attenuation of the noise components could be insufficient. An order higher than 8 or so, should be avoided because of the long impulse response of these filters. Many researchers hate the idea of attenuating some frequency areas but maybe they should not. A well-known fact is that the intelligibility of speech, especially when the listening conditions are poor, improves when the low frequencies are attenuated. The influence on the results of the commonly applied analysis types is negligible. But, it’s hard to fight conventions...

27.3. Signal clipping

The audio signal that is input to an audio device, like an amplifier or audio recorder, must be suitable to its input requirements. In the list of specifications of these devices, the *input sensitivity* can be found which defines the *minimum* input voltage to reach the maximum output of the device (in cases of a power amplifier) or to reach the maximum recording level (in case of a recorder or a computer). Larger input voltages can be attenuated by the input volume control of the signal receiving device so that the amplitude peaks of the signal will not be limited (clipped) by the receiving device to its absolute maximum level. Usually, however, there is a limitation to the *maximum* input level as well, even when the input volume can be adjusted to attenuate the signal such that the output level remains below its maximum. The reason is that this clipping of the signal can occur in the electronics stage(s) before the signal reaches the volume control electronics. The signal peaks are cut-off abruptly: the input is **overloaded**. Unfortunately, the *maximum input voltage* is almost never mentioned in specifications of audio equipment. Therefore, people may painstakingly adjust the recording level of an *already clipped* signal.[1]

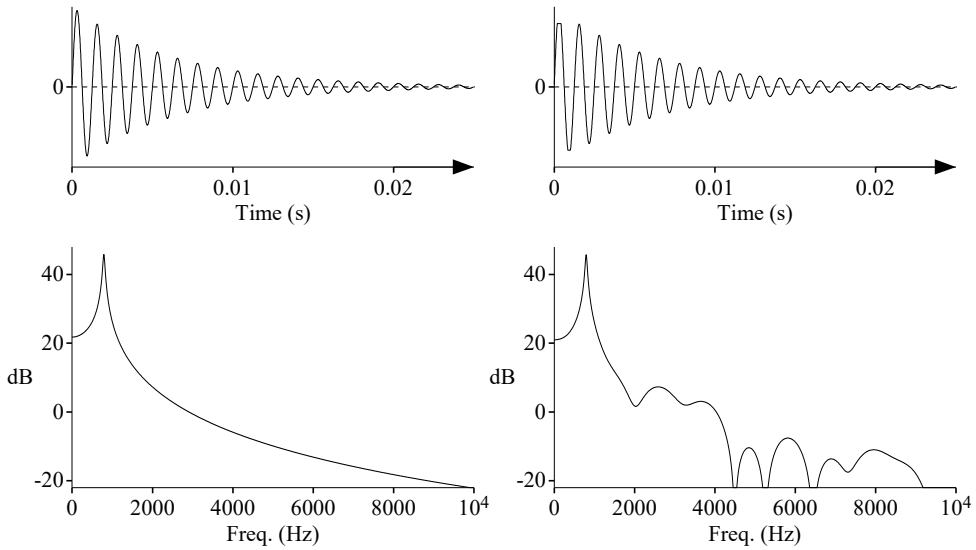


Fig. 27.3.1. Left column: undistorted damped sine and its spectrum. Right column: clipping only a small part of the first amplitude peak in the time signal causes great differences in its spectrum.

To demonstrate the effect of clipping on the signal properties, see fig. 27.3.1 for an example. The left column shows the first 25 ms of a one second damped sine of 800 Hz and its spectrum. The right column displays the same, except for a slight flattening of the first peak of the time function. This minor modification of the time function still

[1] Audio equipment should be designed in such a way that the input signal can be volume-controlled before it passes any active circuitry (amplifying stages). So, do not count on the digital push button volume controls or software controls in this respect!

produces a very different spectrum, as you can see. Especially for speech analysis such as formant measurements, signal clipping can corrupt the analyses severely as shown in fig. 27.3.2. Here the sound is an artificial 'vowel' with only two formants. The clipping even being rather weak, the spectral result shows lots of 'formants'.

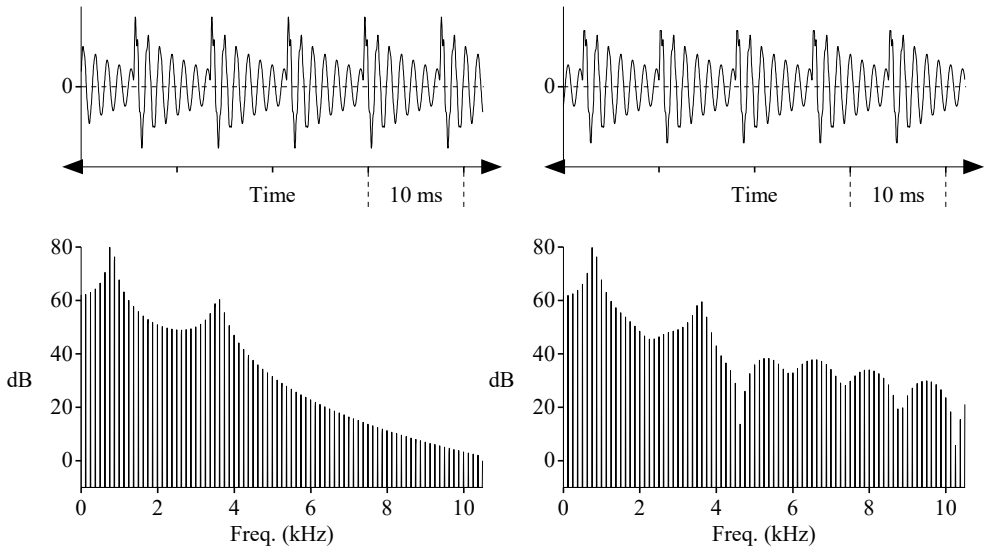


Fig. 27.3.2. Left column: artificial vowel sound with two 'formants' and its spectrum. Right column: clipping a small part of the amplitude peaks in the time signal causes many 'false formants' in its spectrum.

Additionally, clipping of signals cannot be corrected afterwards: there is no way to reconstruct the waveform above the clipping level. (Theoretically, in the examples given, the waveforms could be reconstructed from the knowledge that the signal contains only one or two damped sines with known frequencies; in practice, however, the waveform is the result of a vast and unknown number of frequency components with unknown frequency, amplitude and phase so that the cut-off parts are highly unpredictable.) There exist some methods to process the clipped signal in such a way that the undesired spectral effects will be suppressed somewhat (based on limitation of the intensity of its first derivative) but the results are mainly unsatisfying.

So, avoiding clipping is of paramount importance. How to avoid it? There are several positions in the audio chain where the signal volume could be too high, causing it to become clipped in the following electronics. See fig. 27.3.3 for a functional diagram of a typical audio chain. The maximum interconnection signal level of sound equipment like players, recorders, radio tuners, etc. has been standardized to **line level** so that all outputs of devices can be coupled to inputs of other devices. Originated in the telephone technology, this line level was defined as 1 mW (milliwatt) power into a 600 ohm 'line impedance', which corresponds with a rms voltage of 774.6 mV (millivolt) across the line. On the dB scale, this reference is indicated as 0 dBm. Although this 600-ohm line impedance does not apply to the audio interconnections, this voltage is maintained as

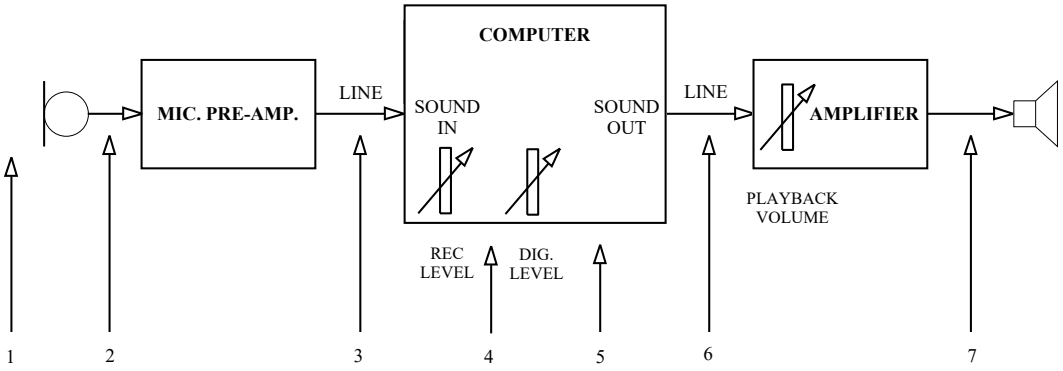


Fig. 27.3.3. Positions in an audio chain for possible overloading inputs of next parts.

the normalized level, and expressed on the dB scale as 0 dBu, where the 'u' stems from 'unloaded', i.e. not loaded with 600 ohms or any other resistance. (Also, dBv is used.) Unfortunately, this level standard is not unique. The dBV, for example, refers to a 0 dB value of 1 volt rms. Things are complicated further because the 'professional' audio equipment uses a line level of 4 dBu (of which 0 dB refers to 774.6 mV), representing 1.228 V, and the 'consumer' audio uses -10 dBV (of which 0 dB refers to 1 V), representing 316 mV. All voltages are rms. Devices which cannot meet the line level standard are microphones, loudspeakers and headphones. The signal from the microphone is mostly less than 20 mV so that a pre-amplifier is needed to raise the mic output to line level. The loudspeakers and headphones need power, which means that they require current, whereas the line outputs cannot produce currents of these sizes. A standard line output cannot 'drive' loudspeakers or headphones. Therefore, speakers and headphones need special 'power amplifiers'.

With reference to fig. 27.3.3 we can now sum-up the sequential positions in one channel of the audio chain for possible signal overloading as follows:

1. Too high acoustical volume that the microphone picks up. As most types of microphones have some electronics built-in, there is a maximum sound pressure level to process without excessive distortion. That level is specified by the manufacturers of the majority of microphone types. The only way to prevent overloading here is to create a greater distance between sound source and microphone.
2. Too strong signal at the input of the pre-amplifier. Obviously, here is the solution also increasing the distance from source to microphone. (Some pre-amp types have a switch which can decrease the amplification in one or two 10 dB steps to accommodate high acoustical volumes.)
3. Too high level at the input of the sound system of the computer or sound recorder. As most microphone pre-amplifiers do not have an output volume control, the first possibility to adjust the signal level is the recording volume control of the computer or recorder. As already mentioned above, the signal can be clipped in the electronics which the signal passes before it reaches the recording volume control. In addition, overloading the audio inputs of most sound recorders or computers will cause a sudden

serious **cross talk** between the two audio channels: the signal from the left channel emerges also in the right channel and vice versa.
4. In cases of audio recorders, now the signal can be controlled so that the maximum recording level is not exceeded. All audio recorders have a recording level control and a level display. For computers, similar possibilities for checking and controlling the recording level are present in the recording software. Therefore, avoiding clipping (or *overmodulation*) here is easy to carry out.[1]
5. After manipulating the sound in the computer the result is usually played back to listen to it. The peak amplitude of the result might have become too high for proper DAC (digital to analog conversion) so that the analog signal will be clipped. The obvious solution is sufficient attenuation of all sample values with the same factor. (This position does not apply to sound recorders.)
6. The analog (line) output signal of the computer or recorder can be too high for the input of the power amplifier. Usually, power amplifiers have their own volume controls but here the same danger exists as described above for signal clipping in the electronics before the controls.
7. The last clipping possibility in the audio chain is the signal limitation by the speakers or phones themselves or clipping in the last power amplifier stage by overloading. In general, this is directly audible but many people are 'too tolerant' regarding signal distortion so that they are not aware of a substantial amount of it. To avoid side effects of this distortion in the perception (experiments) of the sound material it is necessary to adjust the volume control to make sure that the maximum power limit is not exceeded, for speaker/phones as well as for the power amplifier itself.

Whether or not clipping occurs in the electronics before the recording volume control can be tested by connecting the (high-volume) sound source with the recorder's or computer's input, setting the recording volume low enough to prevent overloading the recorder or computer by checking its recording level display, and test the waveform during playback. Some maximum input voltages of the line inputs of a few devices found in this way:
sound card in pc1: 1.45 V
sound input pc2: 1.0 V
sound card in pc3: 2.5 V
recorder Zoom H2: 600 mV
recorder Edirol R09: 4.9 V
recorder Edirol R1: unlimited (input is directly connected with level control)

You can see that these clipping levels can vary a lot and that some maximum input values are lower than the 'professional' line output voltages. In general, connection of a 'professional' device having a line output of 4 dBu, with a 'consumer' device

[1] Very short sound segments ('bursts' or 'transients') can be too short for proper indication on the older (slow) types of VU meters. One turns-up the recording level until the display shows a sufficient value, causing almost certainly clipping of the signal. Modern electronic VU displays are very fast and hold the peak levels a bit longer so that proper adjustment is possible.

having a line input of -10 dBV may easily cause clipping in the receiving device.

The **headroom** is the amplitude range between the maximum recording level and the clipping level. So, only 'consumer' devices with sufficiently headroom can be connected with 'professional' devices without clipping problems.

Obviously, the recording level control should be adjusted such that the highest peak of the input signal should not exceed the maximum ADC input voltage. On the other hand, the input signal should not be attenuated too much because it would 'use' only a small part of the possible number of amplitude steps and the number of bits would be limited, raising the *sample noise*. Because the intensity fluctuations of the input sound are huge (60 dB means a factor 1000 in amplitude, for example), the recording level should preferably be displayed on a log scale. The level indication (commonly still named 'VU meter' after Volume Units) on the audio recorders or in the computers usually approximates a logarithmic response, whereby the maximum level is defined as 0 dB. To avoid overloading, there is often arranged for a safe margin of about 3 dB. In practice, it seems that the majority of the 'sound engineers' tends to set the recording level too high. The peaks in the signal occur not very frequently and the average level is much lower so that the recording level seems quite low. To 'see the signal' clearly, the level is raised and the chance that peaks are clipped is higher. (See also the next segment about signal-to-noise ratio in this respect.)

To simplify adjustment of the recording level, some recorders have an 'automatic volume control' feature. If a sudden peak in the signal intensity occurs, the amplification factor is decreased very fast, such that the maximum recording level is not exceeded. After that, the amplification is increased very slowly, until a next signal intensity peak causes sufficient decreasing again of the amplification. In this way, the amplification is matched to the peaks in the signal intensity and will not change substantially. Of course, the sudden decrease of amplification will distort the waveform of the start of sudden signal bursts 'transients'. Together with the property of fluctuating amplification, this feature is not quite suitable for sound analysis. Fortunately, in most recorders this feature can be switched off.

Another means of avoiding clipping is the 'signal limiter'. This device provides for a gradual decrease of the amplification as the (absolute value of the) amplitude increases. This manipulation works 'direct' on the waveform and, therefore, lacks the distortion of transients as in the case of the automatic volume control. However, the signal waveform is altered already at much lower levels than the absolute maximum which means that distortion occurs already from medium levels on, to the maximum. (See also the next section about distortion.)

A final note about built-in microphone pre-amps:
The use of the microphone input of a standard sound card, or the mic input of a laptop would make the use of a mic pre-amplifier superfluous. However, these inputs can be overloaded quite easily: the clipping threshold is about 50 mV_{eff} which is reached with

a commonly used condenser mic at approximately 110 dB SPL. Especially when the distance between mouth and mic is short (as is advisable in connection with the suppression of background noise and chamber resonances) this upper SPL limit can be too low. This means that it is important to avoid using the mic inputs of pc's and laptops. Using the line inputs instead (or, depending on the software, setting the mic input at line level), together with a proper microphone pre-amp, is a much better solution. (Obviously, the mic pre-amp should not produce a too high output voltage for the line input of the computer or recorder.)

27.4. Signal to noise ratio

The signal to noise (S/N) ratio has been mentioned many times in the book already. It is a convenient measure to define an important aspect of quality of a (recorded) sound. Formally, the S/N ratio is the power of a 'clean' maximum audio signal, usually one sinusoidal component of 1000 Hz, divided by the power of the noise, expressed in dB's. From section 2 you may know that, for amplitudes (or rms values) the ratio in dB's is $20*\log(v_S/v_N)$, where v_S is the signal amplitude and v_N the noise amplitude, represented as voltages. For practical reasons, the level of the signal is measured inclusive the noise. So, strictly speaking, the power formula should be $10*\log[(P_{S+}P_N)/P_N)]$. In practice, this error is ignored because, usually, the signal level is much higher than the noise level. As you know from section 15 about noise, its average power depends on the frequency range of its spectrum. Therefore, the S/N ratios are specified for a 'bandwidth' (frequency range) from 20 Hz to 20 kHz.

Especially, in earlier times of analog audio processing, all commercial audio equipment companies competed with each other to specify low S/N ratios for their products. Figures like 65 or 70 dB were seldom exceeded, mainly because of the limitations of the recording media (vinyl records, magnetic tape). Now, in this digital era of audio recording and processing, the noise of the digital 'medium' itself consists only of 'sample noise' which can be chosen almost freely by the number of bits of the sample values. This 'number precision' then is an economic matter (see section 24 about signal compression). The remaining crucial parts w.r.t. noise are the ADC (analog digital converter) and microphone pre-amplifier electronics. In this light one can question the contribution to the audio quality of the 24-bit precision of the 'high end' recording equipment. As you may know from section 17, the theoretical S/N of 16-bit precision as applied in the standard CD and common sound cards is about 98 dB. When peaks of the acoustical volume are allowed to a level of, say, 110 dB_{SPL}, the acoustical noise would amount to 12 dB_{SPL} only. This level is far below the residual noise of even a special built sound isolated room. Therefore, the intensity range of 16-bit audio is more than enough. The theoretical intensity range of 24-bit audio is 146 dB, which can be labeled as 'ridiculous'. In practice, this S/N figure cannot be approximated at all, neither by the ADC, nor by the electronic amplifiers. In addition, the thermal noise of the microphone itself limits it theoretical S/N ratio to less than 65 dB (see section 26.6 about microphone types). Even more, the unwanted acoustical background noise usually produces a much higher voltage in the microphone than the voltage of its noise floor.

The earlier analog registration systems of sound had a big disadvantage compared to the digital recordings: copying recordings multiple times decreased the S/N ratio each time about 3 dB (when the recording levels were always adjusted optimally). So, it was advisable to 'start' with a S/N as high as possible. The digital recordings, however, can

be copied as many times as one wants, without any degradation of quality. There is no need any more to raise the S/N in this respect too.

As may be clear, the specifications of frequency range (see section 27.2) and S/N ratio of a device are not independent of each other. As the bandwidth of the noise is increased, the noise power increases as well, causing the S/N ratio to decrease. Also, the allowed distortion has some influence on the other two specifications, as a higher distortion factor allows increasing the maximum signal level. Therefore, it makes not much sense to compare different brands of equipment by the figures of a single feature.

A commonly used alternative method to measure the S/N ratio is via spectral 'correction' by **A-weighting**, which takes into account the different sensitivities to

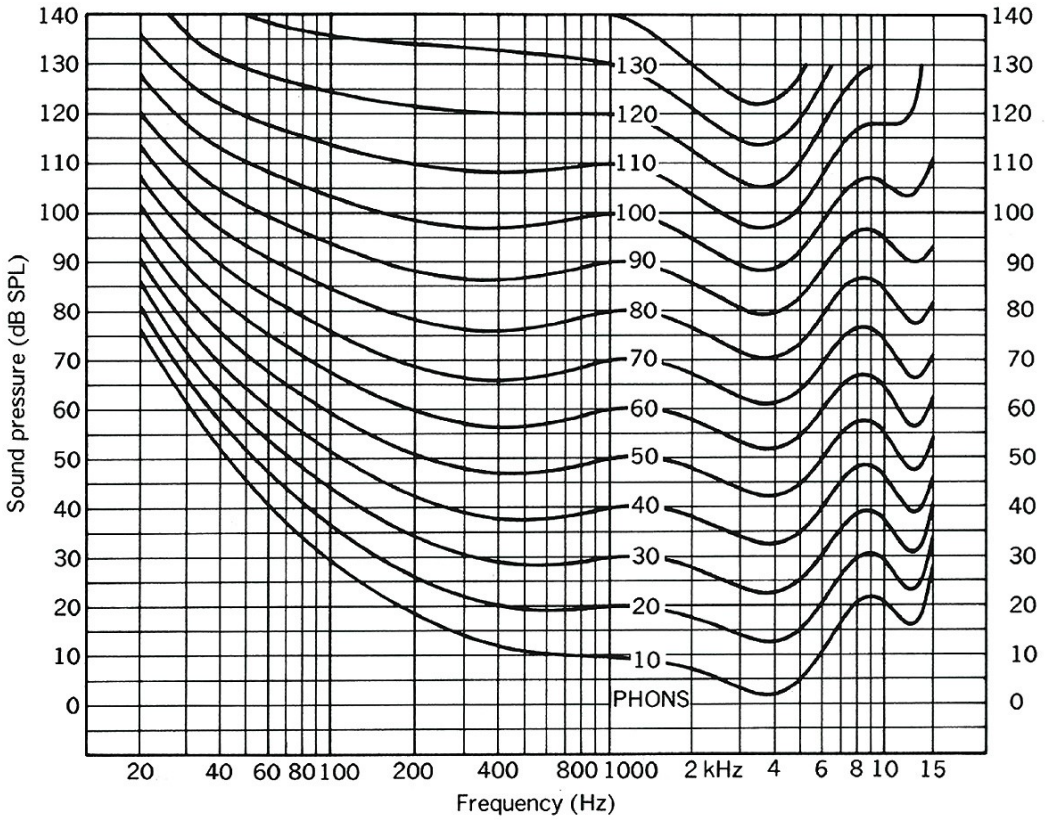


Fig. 27.4.1. Equal loudness contours of the human hearing.

different frequencies of the human hearing. In fig. 27.4.1 the 'equal loudness contours' of the human hearing are displayed on log scales. The graphs are modified versions of the original curves found by Fletcher and Munson in 1933. As you can see, the sensitivity for very low frequencies can be roughly 50 dB lower than the sensitivity at 1000 Hz, depending on the intensity. The reference frequency for the perceived intensities is 1000 Hz, measured in **phons**. The 'standard' A-weighting curve has been derived from the 40-phon curve and is a simplified reverse version of it, as displayed in fig. 27.4.2. This curve represents a commonly agreed normalized 'correction' of the

human perception of sound intensity as function of frequency. Unfortunately, the A-weighting curve is only valid for one intensity level (the *quite low* loudness value of 40 phons). There are other weighting norms for higher intensities, like the B and C curves, but they are seldom applied. In sound intensity meters ('dB meters'), for example, there are usually two measure possibilities only: linear, specified as dB_{SPL} and

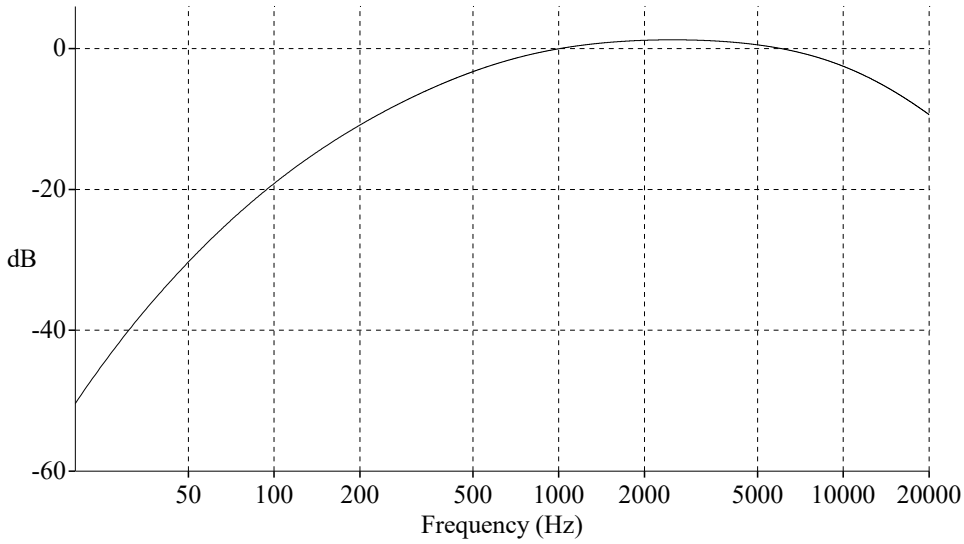


Fig. 27.4.2. A-weighting curve (ANSI standard).

A-weighted, specified as dBA.
When S/N measurements of audio devices are A-weighted, the spectrum of the noise is multiplied by this function, causing attenuation of low and high frequencies, w.r.t. the center frequency around 1000 Hz. Obviously, the A-weighted S/N ratios will be more favorable than the 'linear' S/N ratios. That may explain why many companies define the S/N ratios of their equipment as A-weighted ratios...

As already mentioned in the preceding section, the sample noise is the lowest when the whole range of the ADC is covered by the signal amplitude. Therefore, people tend to turn-up the recording level. The levels of the signal peaks, however, are quite unpredictable so that the danger of clipping increases. When the input level is halved, the S/N ratio is decreased with 6 dB only. On the total range of 98 dB this is of minor importance, while the clipping danger has practically vanished. Although many people think that a low recording level means a bad S/N ratio, the truth is that, with exception of extremely low recording levels, the overall S/N ratio highly depends on the input electronics of the microphone preamp. As, generally, this electronic part is working on the signal *before* its recording level is controlled, this means that the position of this control is of minor importance w.r.t. the S/N ratio.

So, the effects of overloading being severe, the best compromise then is to give the avoidance of clipping much higher priority than lowering the null noise from the equipment. This is even more true when you bear in mind that the acoustical background noise's intensity is much higher.

27.5. Distortion

When the relation between the output signal and the input signal of an amplifier (the *gain function*) is presented in a graphic way, this would result in a straight line of which the slope represents the amplification factor which is equal to the tangent of the angle with the horizontal axis. A waveform at the input will produce the *same form* of that waveform at the output, only with a different scaled amplitude. When the output/input relation is a curved line, the amplification is not fixed but depends on the slope at the position on the graph: the form of the output waveform differs from that of the input and the signal has been **distorted**. See fig. 27.5.1 for an example of this distortion of a sine wave.

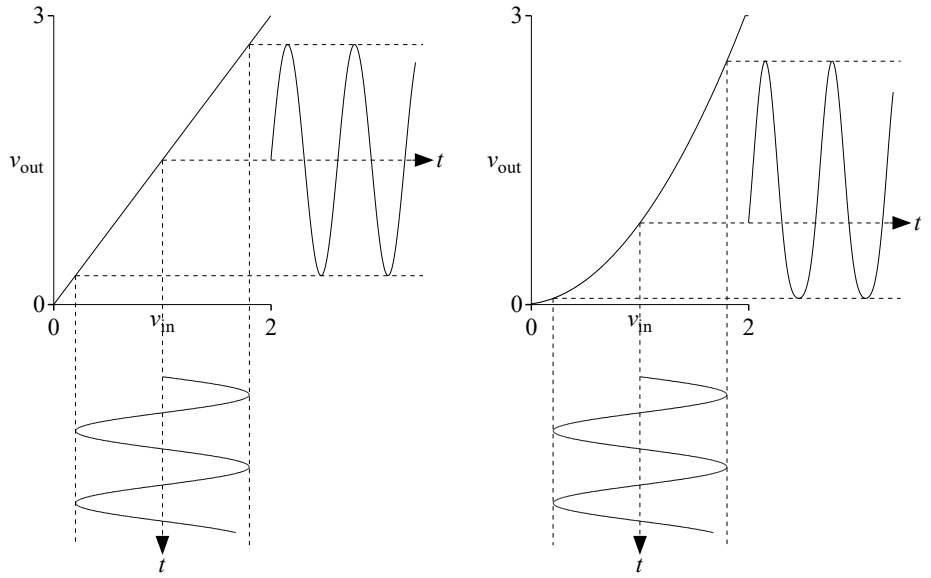


Fig. 27.5.1. Left: undistorted amplification of sine wave. Right: amplification with distortion of wave form by curved gain function.

This type of distortion is commonly called 'non-linear distortion' to distinguish it from 'linear distortion' which refers to the unwanted *filtering* of the signal, like unwanted limitation of spectral range or deviation from the spectral target. Unlike this linear distortion, having the possibility to correct it afterwards by reverse filtering, the non-linear distortion *cannot be corrected* afterwards. (An extreme example of non-linear distortion is the *clipping* of the signal, as described in section 27.3.)

So, in practice, amplifiers or other analog devices are designed in such a way that their gain functions are as straight as possible to minimize the waveform distortion. Many electronic amplifying components function in such a way that their gain curve has a 2nd order form, producing 2nd order or *quadratic distortion*. (In fact, the curved gain function in fig. 27.5.1 is a quadratic one.)

The spectral effects of the quadratic distortion of signals are demonstrated in fig. 27.5.2, where the spectral changes are shown for one single sinusoidal wave (upper part) and the spectral changes for a signal containing two sine components (lower part). In case of a single frequency component, one extra component emerges with double frequency (and a DC component). In case of two frequency components, the extra emerging components are the two double frequencies, their sum, and their difference (and a DC component). So, the extra components that emerge are all possible double, sum, and difference frequencies. You can imagine that, in practice, the number of 'spurious' components can become very large for sounds which have many spectral components. Especially, the *sum* and *difference* frequencies are disturbing because, in general, they

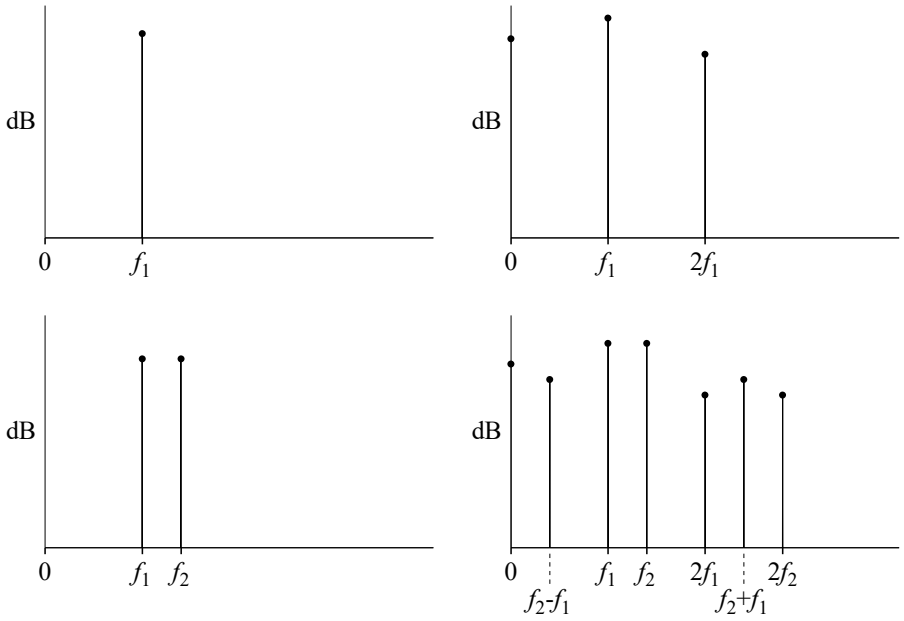


Fig. 27.5.2. Left: spectra of undistorted sine wave(s). Right: the spectral effects after quadratic distortion of the waveforms.

bear no harmonic relation to the original components in the signal. The distortion which consists of sum and difference frequencies is also called *intermodulation distortion*. As you can read in section 10 these sum and difference frequencies result from amplitude modulation which occurs by multiplication of frequency components. In fact, distortion components occur by multiplication of the signal components by themselves. The number of these multiples depends on the order of the output/input function.

For clarity, the amount of distortion in the examples of the figs. 27.5.1 and 27.5.2 is very great as the used part of the quadratic curve is chosen to be large. Fortunately, in the electronic circuitry of amplifiers the amount of distortion can be greatly decreased by *negative feedback* which limits the part of the curve used for the signal amplitude range. (Explanation of this negative feedback principle falls beyond the scope of this book.) Distortion figures like 0.01% or less are quite common for practical amplifiers. The distortion of loudspeakers, however, is much greater, depending on the produced

volume, and may easily reach values like 5 % or so.

The tendency among some audio electronics users to apply *electronic tube* amplifiers instead of transistor amplifiers to 'improve' the audio quality of the sound can be at least labeled as 'astonishing': the gain function at the input of a tube amplifier is the quadratic one in fig. 27.5.1. The emerging sum and difference frequencies of all combinations of two of the spectral components of the input sound will produce a great number of non-harmonic components which have nothing to do with the original sound. In addition, the common opinion among this group of 'audiophiles' about negative feedback is that it must not be applied *at all* which means that the signal uses a substantial part of the curve and will yield a high distortion percentage. One peculiar phenomenon of some tube amplifiers is that the sound of an amplitude modulated (AM) radio station can be heard at low volume! How is that possible, because an amplifier is not a radio? The carrier frequency of an AM radio station is too high to pass through any audio amplifier (even if it is a tube amplifier without negative feedback). The quadratic function 'multiplies' each sinusoid component by itself so that the difference component is a DC component when there is no modulation, i.e. when there is a silence in the radio program. When there is modulation, however, the difference contains all audio frequency components. (See section 10 about amplitude modulation and demodulation.) So, when you hear music coming out of the speakers or phones without any music source coupled with the equipment then you have a badly curved input characteristic, and a poorly shielded input circuit. Ten to one it's a vacuum tube amplifier...

How is the percentage of distortion defined? The distortion produces a number of extra frequency components added to the components of the audio signal. Because the distortion for each signal frequency component may be different, the distortion is defined for one 'standard' signal frequency only, usually 400 Hz or 1000 Hz. When the distortion figure is given, the applied frequency should be mentioned as well. The distortion of one signal component can contain many components, as shown above. Therefore, the adopted definition of the distortion figure is: divide the total energy of all distortion components by the energy of the clean signal component. This energy ratio is then 'converted' to a voltage ratio and expressed as a percentage. This means that the total rms value of all components of the distortion together is expressed as a percentage of the rms value of the clean signal component. (Sometimes the distortion is expressed in dB's, which thus refers directly to the power ratio or energy ratio.) To get the value of all distortion components, the total output is *notch-filtered* which almost completely suppresses a narrow part of the spectrum which contains the signal component frequency; then the total rms value of the output is measured with an AC voltmeter (AC means alternate current which is the opponent of DC, as mentioned in section 6). The clean signal component is usually not achieved by band filtering but is approximated by measuring the total output, being the signal plus distortion. In practice, the error is negligible when the distortion is low.

Most analog devices have the lowest distortion at mid-values of the frequency. At very

low or very high frequencies, the distortion figure can be substantially higher. It might explain why the audio equipment manufacturers define their distortion figures usually at 1000 Hz.

Some audio recorders are equipped with a feature to avoid clipping of the signal by a **limiter** or **compressor** (as mentioned in section 24, the meaning of the word *compressor* here differs from the data compression as described in that section). The output/input relation of a limiter has a strong curvature for high amplitude levels and a straighter line for low levels. Fig. 27.5.3 displays one example of the many different limiter functions used. This function is defined by the formula:

$$v_o = \frac{1}{1+e^{-5v_i}} - \frac{1}{2} \qquad (27.5.1)$$

As you can see, the distortion for a sine wave of which its maximum input amplitude reaches the strong curvature area of the function is quite high and consists of a great number of components. (As explained in section 6 the symmetry of the positive and negative part means that the spectrum contains only odd harmonics.) At this input amplitude the limitation factor is about 40 % (the amplification near zero is maximum

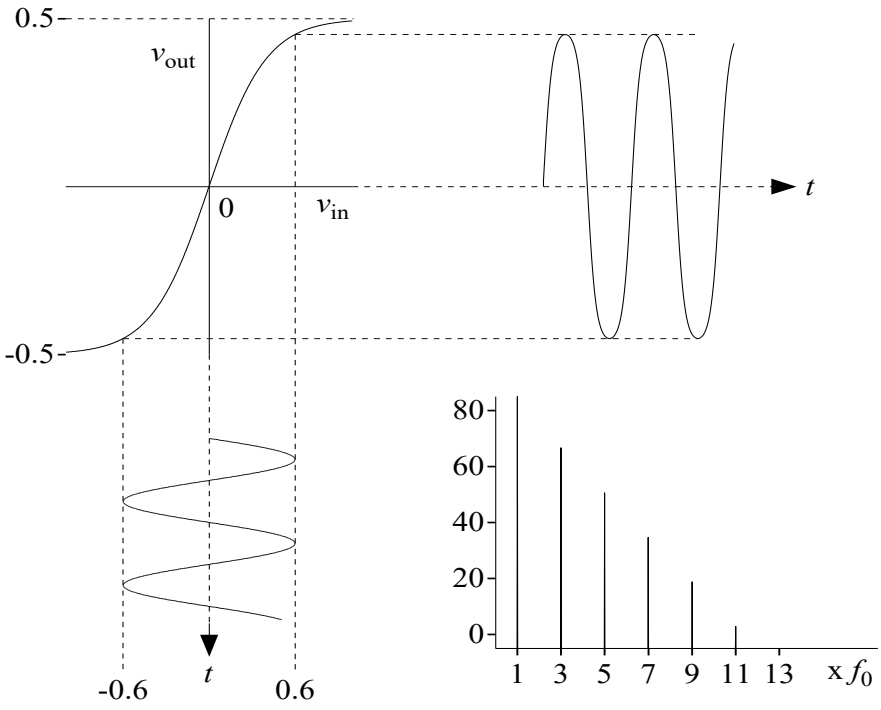


Fig. 27.5.3. Example of an amplitude limiter. The spectrum shows the distortion components as odd multiples of the input signal frequency, here causing 8 % distortion.

and is equal to 1.25; the unlimited output value would be 0.6 x 1.25 = 0.75; the value from the formula is 0.4526 which is 39.7 % of 0.75). Naturally, such a strong limitation

causes a high distortion factor. In this example the estimated distortion figure is about 8 %. When the input amplitude is halved to 0.3, the limitation factor is about 18 % and the distortion in that case still is about 4 % which is definitely unacceptable for 'hi-fidelity' audio or signal analysis purposes. Of course, there are numerous limiter systems which produce less distortion by applying a straighter line at lower amplitudes but, consequently, they need stronger curvature at higher amplitudes. In all systems, the actual amplitude limitation needs curvature of the output/input relation so that the distortion then is too high for many purposes.

Better, therefore, is to avoid using limiters at all and care for a proper control of the input amplitudes of all audio devices instead.

27.6. Microphones, acoustics

The microphone acts as a *transducer* which converts the air pressure fluctuations of the sound to proportional electric voltage. The subject of microphones is very broad which is reflected by the huge number of books on this subject. Instead to try to cover this field completely, here the subject will be limited to a description of the working principles of a few of the most commonly used types and their properties.

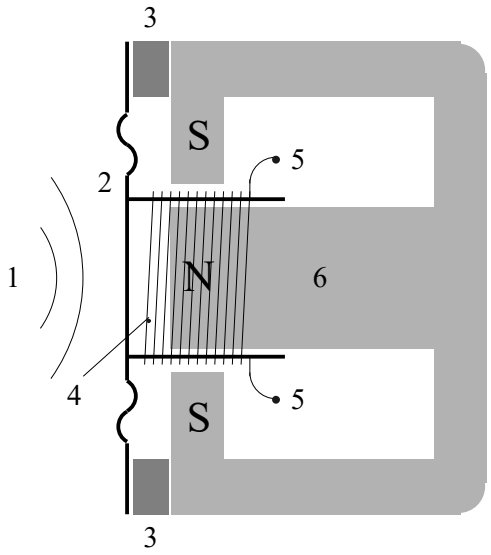


Fig. 27.6.1. Dynamic microphone.
1: Soundwaves in air. 2: Diaphragm. 3: Spacer.
4: Coil. 5: Electrical connections. 6: Magnet.

One of the microphone types which has been used from the beginning of the audio electronics and is still used is the **dynamic microphone**. Its working is based on the electromagnetic induction principle of Faraday: when electric current flows through a wire, a magnetic field occurs around the wire. When this wire is placed in a static magnetic field, a force is applied to the wire, the direction depending on the polarity of the current and the magnetic poles. The reverse is also true: when the wire of the electric circuit is moved within a (static) magnetic field, this movement will cause a current to flow through the wire. The faster the movement, the higher the current. In practice, the effect is enhanced by winding the wire into a coil which highly concentrates its magnetic field as all the windings add to the total magnetism. Using this principle, a microphone can be constructed, see fig. 27.6.1. The diaphragm is moved back and forth by the sound waves in the air. The coil with all windings is moving with it, so that alternating current flows in the electrical circuit of which the coil wire is a part. The voltage alterations across this coil are an electric representation of the sound pressure and can be amplified by the microphone pre-amplifier. For obvious reasons this type of microphone is also named as '*moving coil* microphone'. Because the movements of the diaphragm are extremely small (i.e. maximally around 0.5 µm/Pa which is comparative to the movements of the eardrum), the power of this electrical *source* is also very low. The power is independent of the number of windings, as a relative high voltage by applying many windings causes a reversely proportional low current and, vice versa, a small number of windings causes a low voltage and a relative high current. (Remember that power is equal to voltage times current.) In the first case, the microphone is of the high-impedance type, in the second case it is a low-impedance type. The properties of the pre-amplifier used has to be matched to the impedance of the microphone. As for frequency range and flatness

of the frequency response curve, the low-impedance type is preferred. Its low output voltage is a minor problem as the modern electronics makes it possible to construct low-noise pre-amps that have sufficient amplification for the low output voltage (about 1.5 ... 2 mV/Pa) of the low-impedance types. However, even when the pre-amplifier would add no noise at all, the S/N ratio of the microphone with pre-amp will be limited by the thermal noise which emerges in all electrical conductors, like the coil of the microphone. For example, when the impedance is 200 Ohms (which is a common value for low-impedance dynamic microphones), the thermal noise voltage according to formula 15.2 in section 15 equals 0.257 μV (microvolt) for a frequency range of 20 kHz. When the microphone sensitivity is 1.8 mV/Pa, which is also a common value, the S/N ratio is: 20 log (1800/0.257) = 76.9 dB. (Which is substantially lower than the S/N ratio of a 16-bit AD convertor, while the sound level of 1 Pa = 94 dB_{SPL} is quite loud.)

Of course, the output power of the microphone can be improved by applying a stronger magnet and a higher-surface diaphragm. The latter, however, limits the maximum frequency which can be processed without significant attenuation and deviations from a straight frequency response curve.

A useful way to express the noise properties of microphones is the 'self-noise'. In the example above, the self-noise is equivalent with 94 – 76.9 dB_{SPL} = 17.1 dB_{SPL} which can be seen as a noise sound source of 17.1 dB which is picked up by the microphone, regarded as noiseless. (In practice, this self-noise figure example indicates a fairly high-quality mic w.r.t. its noise properties.)

A special type of dynamic microphone is the **ribbon microphone**. The 'coil' here is no real coil but a flexible corrugated metal ribbon which is positioned between the poles of a strong magnet. The sound in air moves the ribbon in a direction perpendicular to the magnetic field direction so that (low) voltages are generated between the ends of the ribbon.

A different type of microphone which has become most widely used now, is the condenser microphone. Its working principle can be best clarified by first explaining the condenser (or capacitor), see fig. 27.6.2. The condenser can be 'charged' or 'loaded' by connection of a voltage source. Between the plates, an *electric field* arises due to the loads of opposite polarity. When the voltage source is removed, the capacitor remains its charge so that the voltage which had been applied remains present across the capacitor and the electric field continues to exist. The amount of charge (i.e. the number of electrons or positive ions) for a given voltage is proportional to the *surface* of the plates and reversely proportional to the distance between the plates. Also, the choice of isolation material between the plates (the *dielectric*) effects this amount of charge. This amount of charge per voltage is the **capacity** of the condenser: $C = Q/V$, where Q is the load in coulombs and V is the voltage. (Actually, Q represents an absolute amount of electron loads, equivalent to the number of electrons that flow through an electric wire by a current of 1 ampere during 1 second, which is about 6.3×10^{18}.) The capacity of a

condenser is expressed in farads (indeed, from Faraday) which is a very large unit in practice. Therefore, the capacity of condensers is usually expressed in µF (microfarad, which is 10^{-6} farad), or even pF (picofarad, which is 10^{-12} farad). If, instead of a battery,

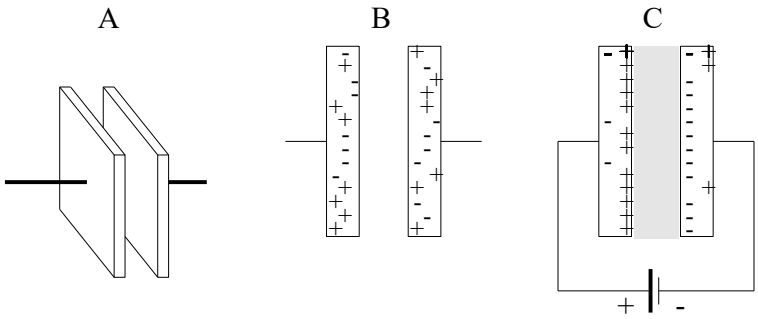


Fig. 27.6.2. Capacitor principle. A: basic model with two electrically isolated conductive plates. B: positive ions and negative electrons are balanced. C: free electrons of left plate move to the positive battery pole; electrons from the negative battery pole move into the right plate and recombine with the positive ions. Between the plates an electric 'field' emerges through which no current flows (grey area).

an AC (alternating current) source is connected, the capacitor will be loaded, unloaded, reversely loaded, and reversely unloaded, etc. ad infinitum. It *seems* as if alternating current flows through the capacitor. In reality, the current flows through the wires and the current source but not through the capacitor. For the 'outside world' it seems that the capacitor is a kind of conductor for AC.

The disconnected charged condenser does not hold its load forever: the isolation is not absolute and some stray electrons will move through it. In practice, the condenser's load will vanish after a few seconds only. For short times, however, the load present in the condenser can be regarded as being constant. When the capacity is changed within this short time (for example by alteration of the plates distance) the voltage will change because Q is constant. An increase of the plates distance, for example, will decrease the capacity and, therefore, *in*crease the voltage across the condenser, and vice versa. This is the principle of the condenser microphone, see fig. 27.6.3. The two 'plates' of the capacitor are created by the diaphragm (or membrane) and the backplate. The capacity of the system varies according to the sound pressure variations so that the voltage across the condenser varies reversely proportional to the sound pressure. In this type of condenser microphone (the **electret** microphone) the necessary constant load to the capacitor is built-in in a layer on the diaphragm material or on the backplate. The layer consists of a special kind of semiconductor which received its load at its fabrication. The load particles are locked within the semiconductor material which can hold its charge during hundreds of years. (Older types of condenser mics used an external high voltage of up to 200 V applied to the backplate to provide for the constant electrical load.) The capacity of the condenser mic is only about 15 to 80 pF. This implies that the impedance of the 'sound source' of this condenser mic is very high which means

that it cannot feed its signal through a cable to the pre-amplifier. (The capacity of the cable itself would attenuate the microphone signal almost completely.) Therefore, the transistor is needed to function as an impedance convertor. A transistor needs some power to be able to function. The power voltage is supplied by the pre- amplifier used, via the microphone cable. (A commonly used method to realize this is the **phantom powering**, which is explained at the end of this section.)

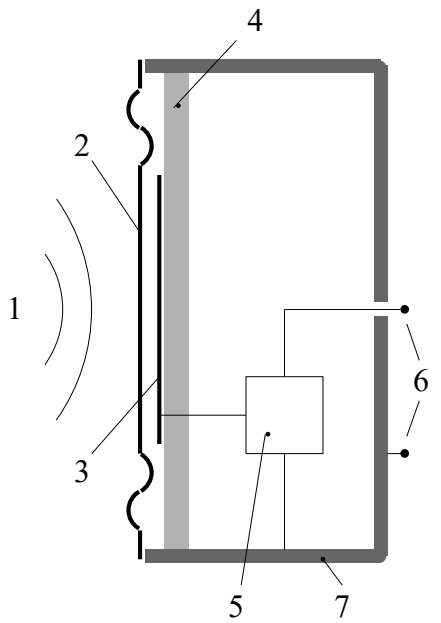


Fig. 27.6.3. Condenser microphone.
1: Soundwaves in air. 2: Diaphragm with electret layer. 3: Backplate. 4: Isolator. 5: Transistor. 6: Electrical connections. 7: Metal housing.

The signal voltage from a typical condenser type of microphone with a diaphragm diameter of ½ inch is, roughly, 10 times of that of a comparable dynamic type. The necessity to a high input impedance of the connected electronics to (i.e. the transistor in the microphone case) implies that the thermal noise level is higher than that of the dynamic microphone. Still, the S/N ratio can be acceptable because of the higher signal voltage of the condenser microphone. In addition, the output voltage of condenser mics can be improved by applying a very narrow space between diaphragm and backplate. Also, the surface area of the diaphragm and backplate can be increased, however, the larger the surface, the more the mechanical resonance frequencies will corrupt the linearity and level of the frequency response for high frequencies.

A different type of condenser microphone is the HF (high frequency) system. It does not work as a loaded or pre-loaded condenser but as a condenser as frequency defining element in a high-frequency oscillator (generator). A nominal frequency of, say, 8 MHz (megahertz) is frequency-modulated by the variations of the capacity and then demodulated to extract the audio frequency components. The gain of this more complicated design is the possibility to create a much lower impedance of the source of the mic: just about 1500 ohm instead of the millions of ohms of the charged capacitor. The much lower impedance means a much lower self-noise. This type offers the best S/N ratio of all dynamic and common condenser microphones (electret or external charge voltage).

Apart from dynamic, (coil or ribbon) and condenser microphones, there exist other types, like *piezo, carbon, ceramic* and even *laser beam* ones. The principles of the two

types described above, however, cover the majority of types used and, therefore, explaining all different types will be omitted in this book.

The choice of the microphone is not critical when the distance between sound source and mic is lower than, say, 30 centimeters or so: even the most ludicrously cheap electret types produce a reasonable sound quality. For *head-mounted* types, however, the mic is positioned so near the mouth that the danger of overmodulation, and thus clipping, emerges. Especially, the cheap kind of electret microphones use a power supply voltage of only a few volts (often only 1.5 V provided via the microphone input of a computer or laptop). Even if the signal is not clipped, the distortion for high sound levels can be quite high in most cases. Usually much better are the 'USB microphones' which convert the 5 volts of the USB connector to a higher value as, for example, 12 volts or higher. A word of caution here: some USB microphones do not sufficiently stabilize the power voltage: the USB power source of a computer is mostly far from 'clean' and components of many different frequencies are present as fluctuations of the nominal value of 5 volts. When the power voltage is not thoroughly stabilized, these frequency components may occur partly within the range where the ear is most sensitive so that they can become audible as soft, ever changing tones at the background. Of course, this decreases the S/N ratio. However, the specifications of the microphones are usually measured by the manufacturer using a 'clean' supply voltage.

As the distance between sound source and mic increases, the quality of the microphone becomes more important, as the output voltage decreases and, with it, the S/N ratio. In addition, the sound quality decreases w.r.t. background noise and room resonances. Whereas the noise of a *dynamic* microphone is mostly determined by the thermal noise of the coil (at least if the noise of the pre-amplifier used is negligible), the S/N ratio will not vary much for most types on the market: generally, it will fall within the range 65...78 dB. The noise of a *condenser* microphone, however, depends mainly on the capacity, the surface area of the diaphragm, the electrical charge (whether or not built-in), and the transistor used. Indeed, the self-noise of condenser microphones on the market differ greatly and, unfortunately, this specification is often not mentioned at all by the manufacturers, especially for the cheap products. The self-noise of cheap condenser mics will be much higher than that of the dynamic ones. The self-noise of the best 'low noise' condenser mics (except the HF types), still is some dB's *higher* than the self-noise of the dynamic types with equal frequency range. Only with the HF condenser types it is possible to realize lower self-noise and thus a higher S/N ratio than with dynamic mics.

Generally, the factory specifications of S/N ratios and self-noise are better than the figures 65...78 dB and 17.1 dB, respectively, as mentioned before. This is caused by the fact that the mic manufacturers all give these data "A-weighted" which means that the noise output has been filtered with the standardized "A-weighting curve" (see section 27.4: from fig. 27.4.2 you can see that the frequencies in the lower area are attenuated considerably). Over the entire frequency range (20 kHz), filtering or not makes only a difference of 2.4 dB if the intensity values were evenly distributed along

this range but... in the case of a condenser mic the noise increases as the frequencies decrease: the noise is not 'white' but 'brown' or -6 dB/oct. So, the relatively high noise intensity at low frequencies is attenuated considerably and the overall S/N ratio and the self-noise specifications are much favorable when the sound is "A-weighted"! One could argue that the ear is less sensitive for this lower frequency area noise (that is the whole idea of this A-filter after all). When the sound is to be analyzed, however, then a high level of this low-frequency noise can cause inaccuracies or even errors.

For recording low intensity sounds at high distances between mic and sound source, the self-noise becomes very important. In the opposite case, for high intensity sounds and microphones placed very near the sound sources, the maximum sound pressure that the microphone can process without too much distortion is of paramount importance, together with the maximum input that the pre-amplifier can accept. There exists no single microphone/pre-amp combination which is suitable for all sound levels: the dynamic range of the sound intensity simply is too large.

When the self-noise is not specified, the S/N ratio may be mentioned instead. In that case, the self-noise can be easily determined by subtraction of the S/N ratio from 94 dB_{SPL}, being the SPL of 1 Pa.

An important feature of microphones is the *directional* sensitivity. Microphones with an equal sensitivity for sounds from all directions, the *omni directional* types, are *pressure transducers*: they react on air pressure. In these types the space at the rear side of the diaphragm is enclosed so that the air pressure variations only affect the front side of the diaphragm. Thus, the position of the diaphragm plane w.r.t. the sound source has no influence on the sensitivity. Microphones can be constructed differently so that the rear side is open to the sound waves as well. In that case, they are *pressure gradient transducers*. They react to the pressure *difference* between front and back of the diaphragm. The ribbon microphone is a pressure gradient transducer.

To specify the relative variation of sensitivity in all directions, a direction diagram is used. Although the sensitivity depends on the direction in three dimensions, it is customary to present directional diagrams or **polar patterns** of microphones in a two-dimensional form as 'seen from above', see fig. 27.6.4. For omni-directional types, the polar diagram is a circle. When the microphone is a pure pressure gradient type like the ribbon type, the polar pattern is a figure-of-eight. A combination of a circular and a figure-of-eight diagram yields a *cardioid* diagram. (The figure-of-eight and the cardioid patterns differ from the 'pure' forms as the distances are presented on a log scale which is the usual presentation form.)

The 0-dB reference of the dB scale is defined at a fixed position in the diagram. Of course, the sensitivities of the different directional patterns at this position are not the same: in fact, the sensitivity of a cardioid mic in this position is about 4.3 dB higher than that of an omnidirectional mic with all other properties equal. In other words: the **forward gain** is 4.3 dB. Different combinations of pressure and pressure gradient

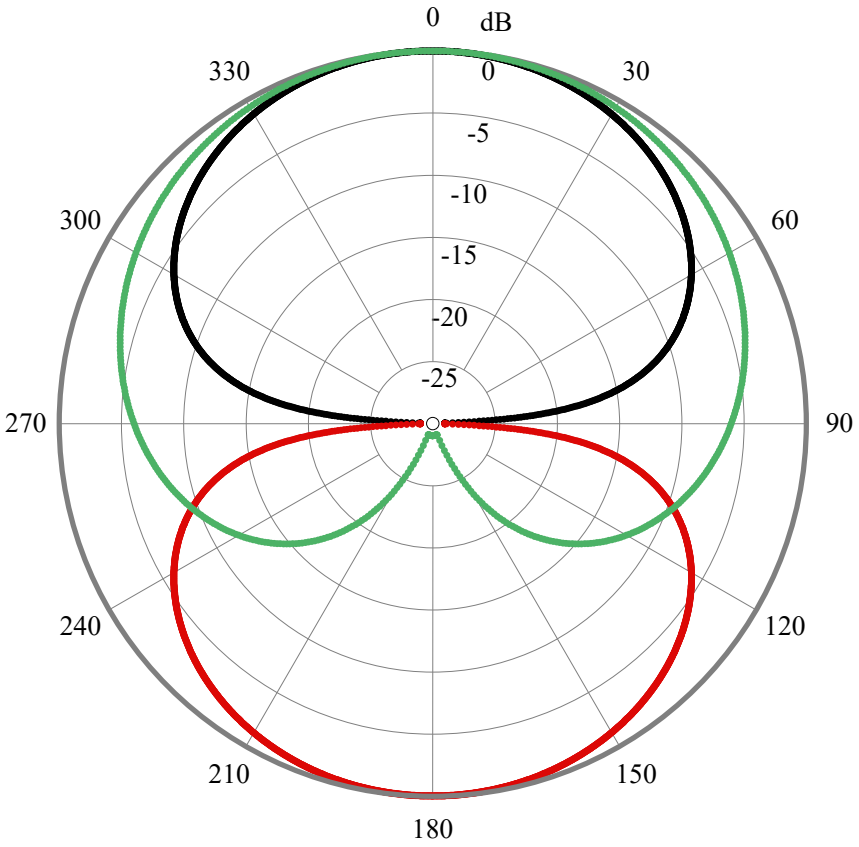


Fig. 27.6.4. Polar patterns of microphones. The cardioid diagram (green) emerges by combination of the omnidirectional (grey) and the bidirectional (black and red) diagrams. The phase of the red half of the bidirectional pattern is opposite to the other patterns phase.

systems produce various kinds of cardioid diagrams like *super cardioid* and *hyper cardioid*. The ultimate directional mic with a very tight polar pattern is the *shotgun microphone*; it has the highest forward gain. It contains an array of several microphone elements in a line and its working is based on the phase differences caused by the distances between the elements. It is used for situations where the sound sources are relatively far away from the mic, i.e. recordings of animal sounds in nature. The frequency response, however, is not very linear.

The general polar patterns specified by the microphone manufacturers are valid only for 1000 Hz. The higher the frequency, the more irregular the polar pattern, partly due to sound reflections by the mic housing. See fig, 27.6.5 for an example of cardioid microphone polar patterns for different frequencies (Sennheiser E 835). As the polar patterns are always symmetrical, only one half is presented; the low range up to 1000 Hz at the left side, the high range from 2000 Hz on, at the right side. You can see that for frequencies below about 500 Hz the pattern is almost circular whereas the patterns for very high frequencies become very irregular.

Which type should be used, omni-directional or *uni-directional*? Not surprisingly, it depends on the purpose of the sound processing action. Basically, the pressure transducers have a constant sensitivity for the whole frequency range, whereas the pressure gradient transducers, which in fact react on velocity of air particles, have a sensitivity proportional to the frequency. To approximate a straighter frequency response curve, the mechanical construction applies a combination of (overlapping) resonances. Alternatively, an electrical filter low-pass filter is used. Nevertheless, for recordings of very low frequencies or for accurate sound measurements, the uni-directional types should better be omitted. Another typical property of uni-directional mics is the **proximity effect**. When the distance between sound source and microphone becomes very short (about 20 cm or less) the sensitivity for low frequencies is increased w.r.t. mid and high frequencies. Although this is generally undesired, it is used sometimes to emphasize the low-frequency sound intensity of solo singers. A third property of directional mics is their susceptibility to wind noise, which is much stronger than that of the omni-directional types. The most important property of uni-directional microphones is their ability to improve the S/N ratio for sounds coming from the front direction by attenuation sounds from other directions and emphasizing the sound from the front by the forward gain. However, the best way to improve the ratio of the desired sound and the background noise is to position the mics very near the sound source (because the SPL of a sound source decreases with the same ratio of which the distance increases, see formula 1.2 in section 1), but if this is not possible, the use of a uni-directional microphone may be the only solution.

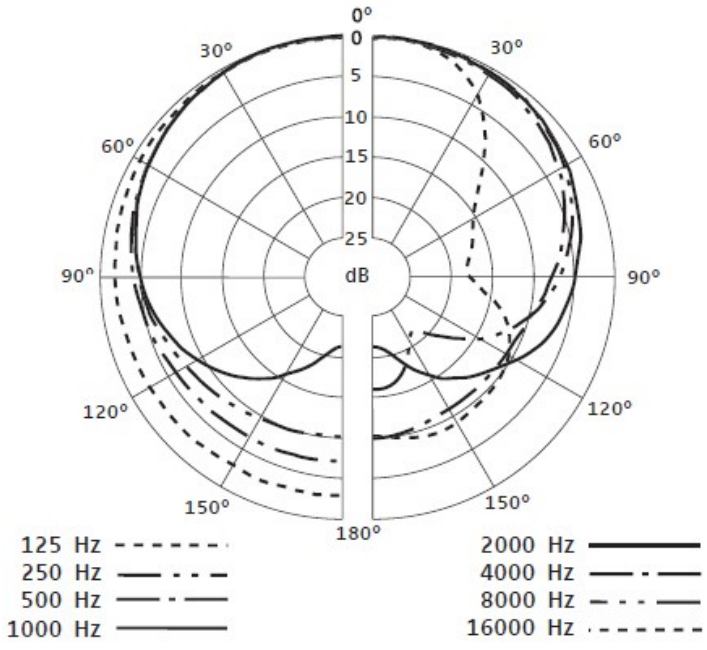


Fig. 27.6.5. Polar patterns of cardioid microphone for different frequencies (Sennheiser type E835).

For recordings, made in *normal rooms* (as opposed to concert halls, sound isolated booths, or quiet outdoor places), the self-noise of the microphone(s) is of minor importance, *even at long distances*, as the usual background noise (from air conditioning systems, traffic noise, computer fans, etc.) will mostly have an intensity above 35 dB, which is more than four times the equivalent self-noise (say, 22 dB) of a medium quality microphone. So, for recordings of sounds in 'normal rooms', the

conclusion then is: forget all lyrical remarks about so-called “high end” microphones: that is a waste of money when used for these recordings. For recording of speech: try to get a “head mounted” microphone which, if possible, is unidirectional and preferably must be powered by 9V or more.

And which type is better, dynamic or condenser? Again, this depends on the application. The condenser microphone has a more linear frequency response than the dynamic type, as the coil of a dynamic microphone has more mass, and thus lower resonance frequency effects compared with the very thin diaphragm of the condenser mic. For high accuracy sound level measurements, only a small diaphragm omni-directional condenser mic is appropriate. (A large diameter diaphragm offers a higher sensitivity and a higher S/N ratio but the high frequencies produce resonances in the membrane which cause peaks and throughs in the frequency response curve.) The processing of very high sound intensities is best done by dynamic types. Also, their resistance against moisture and shocks is great, in general. They do not need power but a state-of-the-art microphone pre-amp is important w.r.t. thermal noise.

‘Phantom powered’ condenser microphones have the advantage that the mic cable can be quite long without interference signals which compromise the sound quality, like humming and cracking, (i.e. 15 m or so is no problem at all). This is not caused by the phantom powering, which is only a method to provide for power to the electronics in the microphone, but by the low impedance created by the electronics used and the symmetrical coupling of the microphone with pre-amp. Naturally the recorder or pre-amplifier then should have this phantom powered input facility. The principle of this phantom powered connection between microphone and amplifier is clarified by fig. 27.6.6. Both leads of the microphone cable receive the power voltage from the pre-amp via resistors (R). The microphone electronics receives its power from the signal leads w.r.t. the shield of the cable. (The shield is a flexible metal wrapping around the signal leads to protect them from picking-up electrical noise signals from the surroundings.)

The microphone electronics provides for two equal signal outputs with opposite phase (**balanced outputs**). When, despite the cable shield, spurious disturbing voltages are picked-up by the cable, these voltages will be equal for both leads in the cable (because the leads are practically in exactly the same position). The pre-amp is a difference amplifier: it only reacts on the voltage difference between the + and – inputs. Thus, the spurious noise voltages cancel each other completely by this **differential connection**.

For dynamic microphones, which usually do not have electronics inside (i.e. the **passive** types), the symmetry of the lead voltages is created by avoiding any connection between coil contacts and shield (the coil is electrically *floating*); the input circuitry of the pre-amplifier cares for a zero voltage (‘grounded’) virtual mid-point so that its + and – input voltages have opposite phases. That is why a passive dynamic microphone still can be connected with a phantom powered pre-amp. It simply does not use the

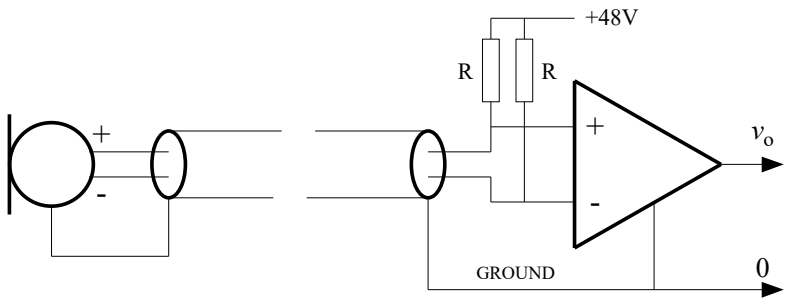


Fig. 27.6.6. Phantom powering of microphone with differential connection.

phantom power (preferably it is switched-off if possible) but the advantage of the differential connection remains.

All information about the *directions* where the sounds come from are contained in two channels, as you can conclude for yourself with your two ears (see also the remark in this respect in section 1 about sound). Therefore, to preserve the direction information of sounds in audio recordings, **stereo** recording is necessary. Two separate sound channels should be used, with two microphones, similar to our two-ear perception of sound.

Often however, more than two microphones are used to be able to position them very near the individual sound sources, i.e. the musical instruments, to improve the S/N ratio. Afterwards, in the mixing stage, the ratio of the left and right part of each individual channel is adjusted to 'place' each sound source somewhere between the utmost left and right positions ('panning'). Although this creates some spatial effect, the directional phase information is gone. Only stereo with two separate channels is able to approximate the real ability of our human two-ear spatial perception. For each microphone, the sound arrives at slightly different times due to the differences in lengths of the *sound paths*. The direction information is contained in the phase differences between the two signals, which are unique for each frequency. As already explained in section 3, the relation between propagation velocity of sound, frequency and wavelength is:

$$\lambda = \frac{c}{f} \tag{27.6.1}$$

where λ is the wavelength, c is the propagation speed of sound and f is the frequency. During one period of a sound wave of, say, 400 Hz, the sound wave travels 340/400 m in space, so its wavelength = 0.85 m. When the difference between the path lengths of the sound source to the two microphones is, for example, 60 cm, the phase difference between the times of arrival comprises 60/85 wavelengths which causes a phase difference of 0.6/0.85*2π = 4.435 radians. When the frequency is much higher, say, 10000 Hz, the wave length is 340/10000 = 3.4 cm. The phase difference then is

60/3.4*2π = 110.98 radians. (The phase angle in each bin of a spectrum cannot be greater than 2π radians; it is equal to the remainder after the division by 2π. The time delays which are greater than one period result from the whole set of angles of all frequency bins of the spectrum object. See section 27.1 about the spectrum representation in Praat.)

There are a few different ways to position the microphones for stereo recording/processing. The most obvious way is the **AB** configuration: two omni-directional or unidirectional microphones are placed in the room, both facing the sound source(s) at some distance. They are usually placed about 50 cm from each other, one at the left, the other at the right. This distance is chosen because it should be at least ¼ of a wavelength of the lowest frequency for containing directional information. Now, our ears (with our brains) cannot perceive directional information below, say, 150 Hz. The quarter wavelength in air of this frequency is about 50 cm.

A problem arises when the stereo channels are summed to get one mono channel, which sometimes may be desired. When the two signals arrived via different paths they might cancel each other by opposite phases. The result is an unwanted *comb filtering* which causes multiple zeros in the spectrum. (When our ears receive sounds with equal intensity but opposite phases, we still perceive the sound: there is no cancelling as we do not add the two sounds in our brains.) To overcome the problem of this cancelling, the two microphones are placed as closely as possible: *coincident* stereo. The **XY** configuration is such a method: the two directional microphones are placed facing each other, usually with an angle somewhere between 90 and 135 degrees (see fig. 27.6.7). Now the phase cancellation is negligible (only at very high frequencies some comb filtering may be possible as the two diaphragms cannot be in exactly the same place). A different coincident method is the **MS** (middle/side) configuration, also displayed in the figure. Here, the center of the sound field is covered by a cardioid microphone facing the sound source, while the side fields are captured by a bi-directional microphone (the grey rectangle), its figure-of-eight polar pattern facing to the left and right. The signals from the two mics cannot be applied directly to the left and right channels but have to be pre-processed by a matrix which converts the MS signals into left and right signals. Therefore, the bi-directional mic signal is duplicated and phase-inversed (output = - input). The left signal is formed by the sum of the center signal and the original side signal; the right signal is formed by the sum of the center signal and the *inversed* side signal. A big advantage of this system is that it is possible to adjust the 'amount' of stereo effect by controlling the volume of the center signal w.r.t. the side signal volumes. This can even be done afterwards with an existing recording of M and S signals.

The disadvantage of coincident stereo is that, in principle, there is no phase difference between the two channels as both microphone diaphragms are (almost) in the same position. The spatial information can only be represented by intensity differences (that is why the term *intensity stereo* is also used). With this configuration, therefore, real spatial stereo cannot be achieved. This problem, however, sounds worse than it is: the

different delay times of sounds, arriving along different paths, remain preserved, of course, so that the sounds from different directions have their own arriving times, giving a strong sense of space. The limitation is that the experienced *directions* while replaying the sound cover only the range from left to right; all other real directions are 'projected' onto this range.
A compromise is realized by the 'near coincident' configuration, like the system used by the NOS (the Dutch Broadcasting Foundation) as also shown in fig. 27.6.7. The

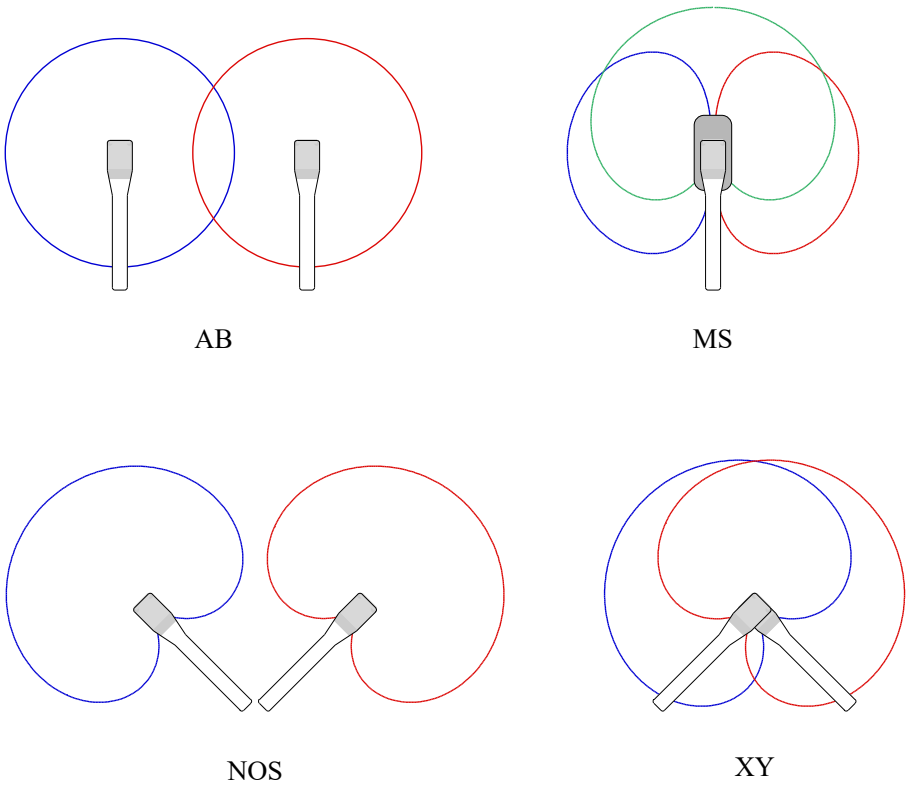


Fig. 27.6.7. Various microphone positions for stereo recording/processing of sound in rooms or concert halls. Blue patterns are processed as left signals, red patterns as right signals, green is the M part of the MS system (see text).

distance between the diaphragms of the unidirectional microphones is 30 cm. Some alternatives exist, like the popular French ORTF system, which uses an angle of 110 degrees and a distance of 17 cm.

The *artificial head stereophony*, as mentioned already in section 1, mimics the human listening situation as accurate as possible. Two omnidirectional microphones are placed in a synthetic model of the human head at the positions of the human ear drums. The two signals are simply processed separately. (Therefore, the name **binaural audio** is also used.) When the (recorded) signals are played back via headphones, the spatial impression is almost ideal. When the signals are played back by two loudspeakers,

however, the result is found dissatisfying in general. This is not very surprising as the listening distance and the listening room acoustics disturb the phase relations between left and right signals. In addition, people in their rooms at home expect a much shorter distance from the listening position to the sound sources compared to the listening distances of the audience in the concert hall, for example.

To avoid the cumbersome realization of artificial head recording, a more practical approximation is possible by using a **Jecklin disc** (named after a Swiss sound engineer). This disk, with a diameter of 30 cm, is placed between two omnidirectional microphones which are positioned 16.5 cm from each other. In this way, a better separation of the two acoustic signals is achieved. The disk is covered with acoustical absorption material to avoid unwanted reflections. A modification which Jecklin applied later is more suitable to playing through loudspeakers: then the distance between the mics is 36 cm while the disk diameter is 35 cm.

A final remark about placing microphones for stereo recording: even if the signals of the left and right channels would resemble very accurately the SPL variations at a normal listening position in a concert hall (say, in the middle of the audience space), the result when listening to the recording via the loudspeakers at home would be disappointing. The main reason is that the listener at home expects a much more direct sound, as if the distance between sound sources and listener were much shorter. The general rule, therefore, is to make sure that the distance from sound sources to microphones will be below about 50 cm (for solo singing or playing) or only a few meters, depending on the size of the orchestra or choir. (For binaural audio it should be more appropriate to position the artificial head just in front of the musical orchestra instead of a position in the audience space.)

Appendix I: Fourier series

The cross-correlation *factor* of two signals of equal duration T can be expressed by:

$$r_0 = \int_{-T/2}^{T/2} x_1(t) \cdot x_2(t) dt \tag{I.1}$$

which simply means the value of the definite integral of their multiplication. As a direct application of this correlation factor we can use it to find the values of the components of the Fourier series. We know that the Fourier components only exist at multiples of $1/T$ and that the phase of a sinusoidal wave component can be accomplished by summing a sine and a cosine component having a proper amplitude ratio. So, if we correlate each sine and cosine wave with the signal over the interval T we get the 'best fit' of all individual components.

Strictly, this can only be done when the component values are independent of each other. The 'optimal' amplitude of one component should not have any influence on the optimal value of any other. This requirement is fulfilled by sines and cosines: they are **orthogonal**. It means that multiplication of only sine and cosine components with different frequencies should result in a *zero* mean. In addition, the optimal sine amplitude and corresponding cosine amplitude are independent of each other. In fig. I.1 two sinusoidal signals with different amplitudes and frequencies together with their products are displayed.

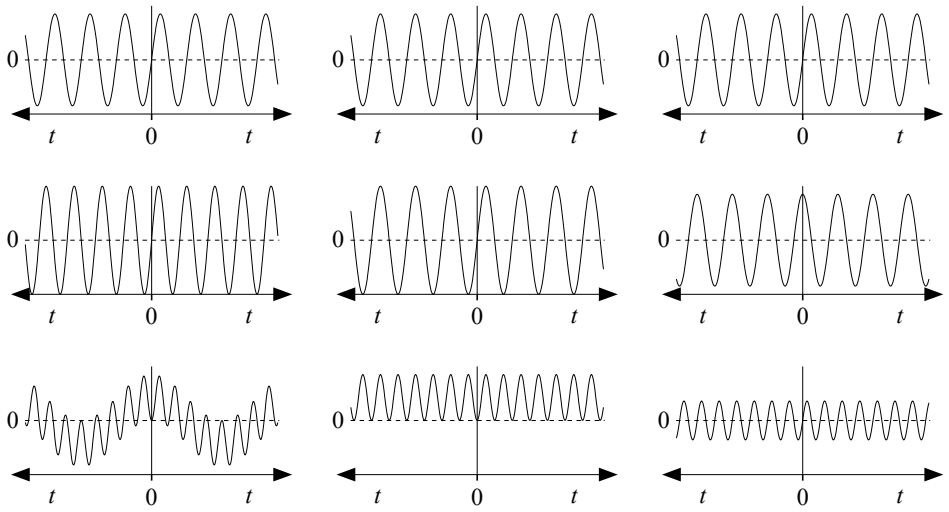


Fig. I.1. Orthogonality shown by multiplication of frequency components. Left column: different frequencies produce zero mean. Mid column: equal frequencies and equal phase produce non-zero mean. Right column: sine and cosine wave with equal frequencies produce zero mean. Other phase differences than π/2 result in non-zero mean values.

The multiplication of sinusoidal signals with *different* frequencies always results in a waveform of which the areas above and below the x axis are equal so that the mean is zero. When the frequencies are *equal* the mean value then depends on the phase difference. A phase difference of $\pi/2$ (i.e. a sine and a cosine) results in a zero mean. Any other phase difference results in a non-zero value.

The orthogonality implies that each component correlation with the signal does not contribute to the other component correlations. Now we can write, for example, the formula for all Fourier cosine components:

$$c_k = \int_{-T/2}^{T/2} x(t)\cos(2\pi \cdot kf_0 \cdot t)dt \tag{I.2}$$

where k is any integer from 0 to infinity. (Because of the fact that the Fourier series always use an integer number of periods of the sine or cosine components, their contribution to each period of the signal is the same. Therefore, the integral interval can be limited to the period boundaries, according to formula I.1.) Likewise, for the sine components:

$$s_k = \int_{-T/2}^{T/2} x(t)\sin(2\pi \cdot kf_0 \cdot t)dt \tag{I.3}$$

These component values being correlations become higher when T gets longer. For *amplitude* values we must divide by the interval time T so that the values are independent of the interval length and the spectrum becomes an *amplitude spectrum*. Furthermore, we must multiply by two because the multiplication of two equal sinusoidal waves of amplitude 1 gives a mean of 0.5. Then, the final formulas for the Fourier *amplitude* components become:

$$a_k = \frac{2}{T}\int_{-T/2}^{T/2} x(t)\cos(2\pi \cdot kf_0 \cdot t)dt \tag{I.4}$$

$$b_k = \frac{2}{T}\int_{-T/2}^{T/2} x(t)\sin(2\pi \cdot kf_0 \cdot t)dt \tag{I.5}$$

(In Praat, this division by T is not applied which implies that Praat's spectra are *density spectra*.)

What if $k = 0$, or in other words, what is the component value at zero frequency? It follows from the formulas that $b_0 = 0$ as sin (0) = 0 and a_0 is the mean of the signal as cos (0) = 1. This a_0 is the 'DC component' of the signal (DC stands for 'direct current'). For most practical audio signals this DC value is zero or very near zero and in general it can be neglected. See part B for more about DC levels.

So, writing any periodical signal as the sum of sine and cosine components leads us to the formula for the Fourier series of a periodical signal:

$$x(t) = a_0 + \sum_{k=1}^{\infty} a_k \cos(k\omega_0 t) + \sum_{k=1}^{\infty} b_k \sin(k\omega_0 t) \tag{I.6}$$

Appendix II: Complex representation of sinusoids

Using complex numbers is not essential for some understanding of the signal analysis within the scope of this book. My hope is, however, that you'll be willing to follow the explanations in this appendix and gradually see the beauty of this way of handling sinusoidal waves. First of all, the reward for thinking in this way is the great power of the mathematics used. And possibly it might be satisfying to discover how *imagination* can lead to a powerful tool for calculations with *real* signals.

II.1 Complex plane

Let's think about a coordinate system with a 'normal' *x*-axis and an 'imaginary' y-axis (see fig. II.1). Any position on this plane can be defined by two coordinates: *a* on the horizontal axis and *b* on the vertical axis. The horizontal axis represents real 'existing' numbers and the vertical axis represents *imaginary numbers*. This stems from the expression $\sqrt{-1}$ which we learned as having no meaning. There is a symbol for this function-without-meaning: the symbol *j*. (In mathematics books the symbol *i* is mostly used, but in the field of engineering this *i* is already used to represent electrical current, hence the *j* to prevent confusion.)

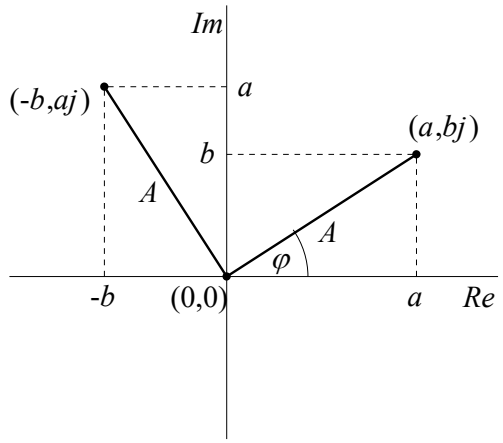


Fig. II.1. Complex plane. Multiplication of a complex vector by *j* results in an angle rotation of π/2 radians.

Now, when we represent a number on this plane we can define it by (*a*, *bj)* for example, which simply implies that we go a distance *a* from the origin in the direction of the horizontal axis and a distance *b* in the direction of the vertical axis. This is the **Cartesian** notation of the number. Any number in the plane can be represented in the form of $z = a + bj$ which is called a *complex number*. A line from the origin to the defined position in the plane is called a **vector**. It has a *magnitude* (the length of the vector) and a *phase* (the angle of the vector with the horizontal axis). If we express the position in the plane by these values we use the **polar** notation of the complex number.

Whether or not the symbol j has meaning, we know that $j^2 = -1$. Then, if we multiply a complex number $z = a + bj$ by j we get the number $-b + aj$. That turns the vector $\pi/2$ radians (= 90 degrees) counterclockwise. Multiplying again with j yields $-a - bj$ which causes another turn of $\pi/2$ radians counterclockwise and which obviously results into $-z$. We see that each multiplication with j is equivalent to increasing the angle with $\pi/2$ radians.

But what happens when we multiply the vector by $\sqrt{j}$? We can solve this problem by considering what happens when we *repeat that action*: then we will have multiplied by j which turns the vector over an angle $\pi/2$. So, multiplication by $\sqrt{j}$ (or $j^{1/2}$) apparently turns the vector half the angle, or $\pi/4$. Likewise, we can apply this trick for any fraction of the power of j. The amount of rotation can be set to any value, positive or negative. We see that the power of j defines the angle.

In polar notation this vector z can be written as:

$$z(\varphi) = A[\cos(\varphi) + j\sin\varphi] \tag{II.1}$$

where A represents the magnitude and φ the angle with the horizontal axis. If $A = 1$ then the positions for all values of z lie on the **unit circle** having a radius of 1.

We can now think of a rotating vector if we vary the angle as a function of time, thus creating a sinusoidal wave:

$$z(t) = A[\cos(\omega t) + j\sin(\omega t)] \tag{II.2}$$

This $z(t)$ is no real sinusoidal wave: we have created a **complex sinusoidal wave**. The great advantage of manipulation with complex sinusoidal waves will be clarified some more in the next appendices.

If we want to create a *real* sinusoidal wave on the complex plane there is a trick: we use *two* complex rotating vectors. For example, if we write

$$z_1(t) = \frac{A}{2}[\cos(\omega t) + j\sin(\omega t)] + \frac{A}{2}[\cos(\omega t) - j\sin(\omega t)] \tag{II.3}$$

the result is the real cosine $A\cos(\omega t)$. As $\sin x = -\sin(-x)$ and $\cos x = \cos(-x)$ we may write:

$$z_1(t) = A\cos(\omega t) = \frac{A}{2}[\cos(\omega t) + j\sin(\omega t)] + \frac{A}{2}[\cos(-\omega t) + j\sin(-\omega t)] \tag{II.4}$$

Fig. II.2 shows the graphical representation: we have one vector rotating in the 'normal' counterclockwise direction and a vector rotating *in the opposite direction* which is the **complex conjugate** of the first vector. Each vector has half the amplitude. The b values on the vertical axis always cancel each other out. The result lies always on the

horizontal, real axis and is equal to A cos (ωt). In any position of the vectors they are each other's complex conjugates.

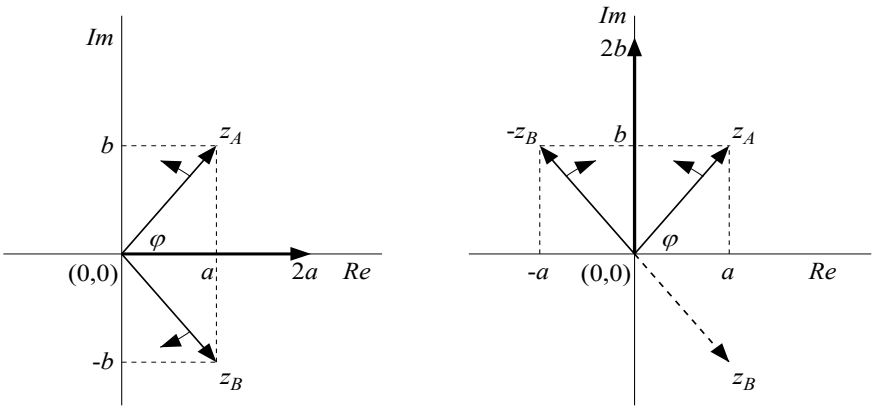


Fig. II.2. Real cosine and sine wave construction by adding two complex vectors, rotating in opposite directions.

In the same way the real *sine* wave $A\sin(\omega t)$ can be made by changing the sign of the second vector. Now the a values on the real axis cancel each other and the result of the sum of these vectors lies always on the *imaginary* axis, which means that we have to divide by j:

$$z_2(t) = A\sin(\omega t) = \frac{A}{2j}\left[\cos(\omega t) + j\sin(\omega t)\right] - \frac{A}{2j}\left[\cos(-\omega t) + j\sin(-\omega t)\right] \qquad \text{(II.5)}$$

Thus, for the construction of real sinusoidal waves using complex waves we need their complex conjugates, and therefore *negative frequencies*.

If we use complex sinusoidal waves let's see what happens when we multiply two *different vectors*:

$$z_M = \left[\cos(\alpha) + j\sin(\alpha)\right] \cdot \left[\cos(\beta) + j\sin(\beta)\right] \qquad \text{(II.6)}$$

This leads to:

$$z_M = \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta) + j\sin(\alpha)\cos(\beta) + j\cos(\alpha)\sin(\beta) \qquad \text{(II.7)}$$

According to our secondary school knowledge (see also the box called **MULTIPLICATION OF TWO SINE WAVES** in section 10) the first half of the formula equals cos ($\alpha + \beta$) and the second half j sin ($\alpha + \beta$):

$$z_M = \cos(\alpha + \beta) + j\sin(\alpha + \beta) \qquad \text{(II.8)}$$

Applying to complex sinusoidal *waves*:

$$z_M(t) = \cos[(\omega_1 + \omega_2)t] + j\sin[(\omega_1 + \omega_2)t] \tag{II.9}$$

which shows that multiplication of two complex sinusoidal waves produces one complex sinusoidal wave with *only the sum* of the individual frequencies.

If the two vectors are the same ($\omega_1 = \omega_2 = \omega$) then we can write:

$$z_M = [\cos(\omega t) + j\sin(\omega t)]^2 = \cos(2\omega t) + j\sin(2\omega t) \tag{II.10}$$

If we multiply the result *once more* with the original cos (ωt) + j sin (ωt) we get the sum of 2ω and ω which is 3ω and so on. In general:

$$[\cos(\omega t) + j\sin(\omega t)]^n = \cos(n\omega t) + j\sin(n\omega t) \tag{II.11}$$

This is **de Moivre's formula**. As a consequence, multiplying a complex sinusoidal waveform with itself a few times shifts its frequency ω radians higher each time.

The fact that multiplication of *complex* sinusoidal waves produces only *sums* of frequencies, not differences, is also true for negative frequencies which we can find out by multiplication of two different vectors rotating in the opposite direction:

$$z_{m-} = [\cos(\omega_1 t) - j\sin(\omega_1 t)] \cdot [\cos(\omega_2 t) - j\sin(\omega_2 t)] \tag{II.12}$$

Substituting α for $\omega_1 t$ and β for $\omega_2 t$ yields:

$$z_{m-} = \cos\alpha \cdot \cos\beta - \sin\alpha \cdot \sin\beta - j\sin\alpha \cdot \cos\beta - j\cos\alpha \cdot \sin\beta \tag{II.13}$$

which simplifies to:

$$z_{m-} = \cos(\alpha + \beta) - j\sin(\alpha + \beta) \tag{II.14}$$

Applied to sinusoidal waves:

$$z_{M-}(t) = \cos[(\omega_1 + \omega_2)t] - j\sin[(\omega_1 + \omega_2)t] \tag{II.15}$$

which is a vector rotating in the opposite direction at the *sum* of the frequencies.

As a consequence, when a vector with a positive rotation and another vector with a negative rotation are multiplied, the result is a vector rotating at the difference of the frequencies.

II.2 Complex exponentials

The great Swiss mathematician **Leonhard Euler** found an alternative way to express complex numbers in polar notation using *complex exponentials*, known as **Euler's relation**:

$$e^{j\varphi} = \cos(\varphi) + j\sin(\varphi) \tag{II.16}$$

The proof of this relation normally is done by using Taylor power series for sine and cosine:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + ... \tag{II.17}$$

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + ... \tag{II.18}$$

and the power series for the *e* exponential:

$$e^z = 1 + z + \frac{z^2}{2!} + \frac{z^3}{3!} + \frac{z^4}{4!} + ... \tag{II.19}$$

As $j^2 = -1$ substituting $j\varphi$ for z in this formula gives us:

$$e^{j\varphi} = 1 + j\varphi - \frac{\varphi^2}{2!} - \frac{j\varphi^3}{3!} + \frac{\varphi^4}{4!} + \frac{j\varphi^5}{5!} - \frac{\varphi^6}{6!} - \frac{j\varphi^7}{7!} + \frac{\varphi^8}{8!} + ... \tag{II.20}$$

If we arrange the terms so that the even and odd powers each are grouped separately, we will get:

$$e^{j\varphi} = 1 - \frac{\varphi^2}{2!} + \frac{\varphi^4}{4!} - \frac{\varphi^6}{6!} + \frac{\varphi^8}{8!} - ... + j\varphi - \frac{j\varphi^3}{3!} + \frac{j\varphi^5}{5!} - \frac{j\varphi^7}{7!} + \frac{j\varphi^9}{9!} - .... \tag{II.21}$$

Substitution by using formulas (II.17) and (II.18) leads directly to Euler's relation (II.16).

The mathematical proof of the Taylor exponential series being rather complicated, we will have to take the explanation for granted at this point. Many books on general mathematical analysis deal with these problems. To gain some insight in the approximation mechanism of the sine and cosine of the Taylor series, we can run DEMO.II.2.1.script and DEMO.II.2.2.script. These show us the result after each addition of a term. Adding terms improves the approximation over a wider and wider area from the center (zero) and only a couple of terms (say, 5 or 6) are sufficient to approximate one sine or cosine ‘period’ quite accurately.

The Taylor series for sine and cosine also teach us also something about the *radian*. If we run our DEMOs II.2.1 and II.2.2 we can define a scale factor in the starting window. If we choose the default value (1) we see that the 'period' of the sine or cosine, i.e. the angle of a complete circle, is equal to 2π! So, the reason for defining a complete circle with an angle of 2π is a very practical one: this factor emerges from the relation between the path of the moving vector (which represents a *part of the circle circumference* of the rotating vector) and the angle of the sine or cosine, given by the Taylor series. (When using radians, the sine of an infinitesimal angle equals the angle itself, on which property the Taylor series is based.) As you know, this radian is not necessarily the only way to define angles (see the box called **THE RADIAN** in section 5). It is perfectly valid to define the angle of the complete circle as 1 so that the angular velocity is the same as the frequency. However, to make use of the convenient Euler relation, we must accept this factor 2π, no matter how we define the angles. If you like 360 degrees in the circle, for example, you have to fill in a factor $2\pi/360$ in the demo starting window. Then a 'period' of the sine or cosine runs from -180 to 180 as you see. As a consequence, the new angular frequency ω' then simply would be $360f$ instead of $2\pi f$.

The reason this radian story is emphasized somewhat is because many people seem to think that the radian has fundamental properties and that the only way to calculate things in the field of frequency analysis is to use the relation $\omega=2\pi f$, which definitely is not true. It has only one practical reason: using the Euler formula greatly simplifies the math but, at a small price, it needs the factor 2π!

If you do not like the use of Taylor series to prove the Euler relation there is another way that uses direct and simple calculus (for example by Freeman [4]):

To prove:

$$e^{j\varphi} = \cos(\varphi) + j\sin(\varphi) \tag{II.22}$$

we can regard the right side of the formula as some complex function:

$$y(\varphi) = \cos(\varphi) + j\sin(\varphi) \tag{II.23}$$

Take the 1st derivative of both sides:

$$\frac{dy(\varphi)}{d\varphi} = -\sin(\varphi) + j\cos(\varphi) \tag{II.24}$$

As $j^2 = -1$ we may write:

$$-\sin(\varphi) + j\cos(\varphi) = j\left[j\sin(\varphi) + \cos(\varphi)\right] \tag{II.25}$$

which means that:

$$\frac{dy(\varphi)}{d\varphi} = j \cdot y(\varphi) \tag{II.26}$$

This important relation shows us that in order to get the 1st derivative of a complex sine wave we simply have to multiply it with *j*! Now, we multiply both sides with $d\varphi / y(\varphi)$:

$$\frac{dy(\varphi)}{y(\varphi)} = j \cdot d\varphi \tag{II.27}$$

and integrate both sides:

$$\ln[y(\varphi)] + c_1 = j(\varphi + c_2) \tag{II.28}$$

Combining the constants yields:

$$\ln[y(\varphi)] = j\varphi + c \tag{II.29}$$

where *c* is a complex constant. To calculate this constant, we substitute 0 for φ:

$$\ln[y(0)] = c \text{ or: } \ln[\cos(0) + j\sin(0)] = c \tag{II.30}$$

That implies that $c = \ln(1)$, so $c = 0$ which simplifies our formula to:

$$\ln[y(\varphi)] = j\varphi \tag{II.31}$$

Write both sides as a power of *e*:

$$y(\varphi) = e^{j\varphi} \tag{II.32}$$

which brings us Euler's formula:

$$\cos(\varphi) + j\sin(\varphi) = e^{j\varphi} \tag{II.33}$$

As we know now that Euler's relation must be true, we can use it to express the sine and cosine waves from formulas (II.4) and (II.5) with their complex exponential versions:

$$A\cos(\omega t) = \frac{1}{2} A\left(e^{j\omega t} + e^{-j\omega t}\right) \tag{II.34}$$

$$A\sin(\omega t) = \frac{1}{2j} A\left(e^{j\omega t} - e^{-j\omega t}\right) \tag{II.35}$$

Compared to formulas (II.4) and (II.5) these forms look much simpler. What's more, the mathematical manipulations with these exponentials are a great deal easier to perform. As an example, let us multiply two complex sine waves again:

$$z(t) = A_1 e^{j\omega_1 t} \cdot A_2 e^{j\omega_2 t} \tag{II.36}$$

This leads straight to:

$$z(t) = A_1 A_2 e^{j(\omega_1+\omega_2)t} \tag{II.37}$$

which shows directly that we get the sum of the frequencies, as in formula (II.9).

Multiplying a sine wave with itself *n* times:

$$\left(A \cdot e^{j\omega t}\right)^n = A^n \cdot e^{jn\omega t} \tag{II.38}$$

produces the exponential form of De Moivre's formula (II.11) ready-made.

When dividing complex sine waves, we simply subtract the one frequency from the other, instead of adding the frequencies, which is quite trivial here. And, again, if we want a *real* sinusoid, we add a second vector that rotates in the other direction, according to formulas (II.34) and (II.35).

II.3. Complex Fourier transform

The previous section showed us how to define a complex sinusoidal wave in exponential form: $z(t) = Ae^{j\omega t}$. If we apply this to the Fourier series formula I.6 of appendix I we get a complex Fourier series formula:

$$x(t) = \sum_{n=-\infty}^{\infty} r_n e^{jn\omega_0 t} \tag{II.39}$$

where r_n must have some relation to the a_n and b_n coefficients. In fact, this equation represents a series of complex vectors rotating in positive direction for positive n and in negative direction for negative n. As we have seen, complex vectors can be converted into real ones by adding equal vectors rotating in the opposite direction. This explains the expansion of the range for n with negative values: here we need the negative frequencies, whereas in the case of the Fourier *series* the index n is limited to positive values only. If we make sure that part of r_n consists of $a_k/2$, the positive and negative rotating vectors of that part of the function then add up so that we get real cosines with amplitude a_n. Next, if we care for the rest of r_n to contain the value $-jb_n/2$ for positive n and $jb_n/2$ for negative n we get real sine components. Concluding, we can write for r_n:

$$r_n = \frac{1}{2}(a_n - jb_n) \text{ for } n > 0,\ r_n = \frac{1}{2}(a_n + jb_n) \text{ for } n < 0 \tag{II.40}$$

If $n = 0$ we get the mean of the signal so that $r_0 = a_0$. So, in general, the exponential coefficients of the Fourier series are complex. Only when a coefficient pair r_n and r_{-n} both are real, it is a cosine. When both members of the pair are imaginary it is a sine. All this should not be very surprising, considering the explanations in appendix II.1 about the complex representation of sinusoids.

Using formula II.40 we can now replace the value r_n in formula II.39 by the values for a_k and b_k of the formulas I.4 and I.5 from appendix I. If n is positive we get:

$$r_n = \frac{1}{2} \cdot \frac{2}{T}\left[\int_{-T/2}^{T/2} x(t)\cos(n\omega_0 t)dt - j\int_{-T/2}^{T/2} x(t)\sin(n\omega_0 t)dt\right] \tag{II.41}$$

Using Euler's formula, we may write:

$$r_n = \frac{1}{T}\int_{-T/2}^{T/2} x(t)e^{-jn\omega_0 t}dt \tag{II.42}$$

If n is negative we can use the identities $\cos(-x) = \cos(x)$ and $\sin(-x) = -\sin(x)$. We can see that for negative n the result is the same: formula II.42 is valid for all values of n

(the value 0 results in the average of the signal's period). The formula II.42, therefore, is the complex equivalent of the Fourier components of a periodic signal (formula. I.6).

In section 9 the concept of continuous spectra was introduced by way of the time insert experiment. We let the period length increase to infinity. At that point, the distance between the spectral components at the frequency axis will have been decreased to zero. The number of Fourier components is then infinite for any frequency interval. While time T in formula II.42 increases, ω_0 tends to zero. The value r_n tends to zero as well. The product r_nT, however, remains a measure for the frequency component values. If we represent that product as X and define it as a function of ω we are now able to change formula II.42 into its continuous form:

$$X(\omega) = \int_{-\infty}^{\infty} x(t) \cdot e^{-j\omega t} dt \qquad \text{(II.43)}$$

which transforms a once-occurring time signal into its continuous spectrum.

When we substitute $X(\omega)/T$ for r_n in formula II.39 we may write:
$x(t) = \sum_{n=-\infty}^{\infty} \frac{X(\omega)}{T} \cdot e^{jn\omega_0 t} = \sum_{n=-\infty}^{\infty} X(\omega) \cdot \frac{\omega_0}{2\pi} \cdot e^{jn\omega_0 t}$. To convert this discrete function into the continuous version we replace $n\omega_0$ by ω and, as ω_0 becomes infinitesimally small, ω_0 by dω:

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\omega) \cdot e^{j\omega t} d\omega \qquad \text{(II.44)}$$

which is the reverse transform: it transforms a continuous spectrum into its time-limited once-occurring time function.

The functions in formulas II.43 and II.44 are known as the *Fourier integrals*.

Using formula II.43 to calculate the Fourier transform of the single rectangular pulse we may write:

$$X(\omega) = \int_{-T/2}^{T/2} 1 \cdot e^{-j\omega t} dt = -\frac{1}{j\omega} \left[e^{-j\omega t} \right]_{-T/2}^{T/2} = -\frac{1}{j\omega} \left(e^{-j\omega T/2} - e^{j\omega T/2} \right) \qquad \text{(II.45)}$$

Applying Euler's formula this can be written as:

$$X(\omega) = \frac{2\sin(\omega T/2)}{\omega} = T \frac{\sin(\omega T/2)}{\omega T/2} \text{ or } X(f) = T \frac{\sin(\pi f T)}{\pi f T} \qquad \text{(II.46)}$$

which represents the well-known sinc function.

There is a striking symmetry in the Fourier integral formulas II.43 and II.44, as you see. (The factor 2π stems from the use of the radian instead of the hertz for the frequency.) When the time function is even, the time-frequency symmetry is ideal, which can be shown as follows. Reversing the time by substituting $-t_R$ for t in formula II.43 reverses the integration boundaries so that we can write:

$$X(\omega) = \int_{\infty}^{-\infty} x(-t_R) \cdot e^{j\omega \cdot t_R} \cdot (-dt_R) = \int_{-\infty}^{\infty} x(-t_R) \cdot e^{j\omega \cdot t_R} \cdot dt_R \qquad \text{(II.47)}$$

Because the time function is even $x(-t_R) = x(t_R)$ so that:

$$X(\omega) = \int_{-\infty}^{\infty} x(t_R) \cdot e^{j\omega \cdot t_R} \cdot dt_R \qquad \text{(II.48)}$$

This is exactly equal to expression II.44 with interchanged functions $x(t)$ and $X(\omega)$. As an example, the digital windowed sinc low-pass filter from section 18 uses this time-frequency symmetry: the spectrum's pass band is almost perfectly limited according to a rectangular function by applying a sinc function approximation as the impulse response in the time domain.

All even time functions with their origin in the center have this ideal Fourier symmetry: they form a *Fourier transform pair*. Obviously, all symmetrical window functions (see section 13) and their spectra form Fourier transform pairs.

The ultimate symmetry can be found (among others) in the Gauss function. Its time function, scaled to 1 at $t = 0$, is represented by $x(t) = e^{-\alpha^2 t^2}$. Using formula II.43 the Fourier transform can be written as:

$$X(\omega) = \int_{-\infty}^{\infty} e^{-\alpha^2 t^2} e^{-j\omega t} dt \qquad \text{(II.49)}$$

To evaluate this integral we reduce it to the special case when $\alpha^2 = \pi$:

$$X(f) = \int_{-\infty}^{\infty} e^{-\pi t^2} \cdot e^{-j2\pi ft} dt = \int_{-\infty}^{\infty} e^{-\pi\left(t^2 + j2ft\right)} dt \qquad \text{(II.50)}$$

As we integrate in t there is no objection to multiply the integrand with $e^{\pi \cdot f^2}$ and correct this by multiplying the function with $e^{-\pi \cdot f^2}$:

$$X(f) = e^{-\pi \cdot f^2} \int_{-\infty}^{\infty} e^{-\pi\left(t^2 + j2f \cdot t - f^2\right)} = e^{-\pi \cdot f^2} \int_{-\infty}^{\infty} e^{-\pi(t + j \cdot f)^2} dt \qquad \text{(II.51)}$$

Substituting u for $t + j{\cdot}f$ gives:

$$X(f) = e^{-\pi \cdot f^2} \int_{-\infty}^{\infty} e^{-\pi \cdot u^2} du \tag{II.52}$$

The function now contains the *Gaussian integral* which equals 1. The conclusion is that the Fourier transform of a Gaussian is also a Gaussian. To apply this to the general Gaussian $e^{-\alpha^2 t^2}$ we have to scale the time with the factor $\alpha / \sqrt{\pi}$. The effect of time scaling on the Fourier transform we can evaluate as follows:

If we define a positive scaling factor p then $u = pt$, $t = \frac{u}{p}$ and $dt = \frac{du}{p}$. We write:

$$X(\omega) = \int_{-\infty}^{\infty} x(pt) e^{-j\omega t} dt = \int_{-\infty}^{\infty} x(u) e^{-j\omega \cdot u / p} \frac{du}{p} \tag{II.53}$$

This leads to the equation:

$$x(pt) \Leftrightarrow \frac{1}{p} X\left(\frac{\omega}{p}\right) \quad (p > 0) \tag{II.54}$$

which expresses the time scaling property of the Fourier transform.

When we apply the scaling factor $\alpha / \sqrt{\pi}$ to our originally defined Gaussian function we get its transform:

$$e^{-\alpha^2 t^2} \Leftrightarrow \frac{\sqrt{\pi}}{\alpha} e^{-(\pi \cdot f / \alpha)^2} \tag{II.55}$$

The spectral bandwidth B can be found in a straightforward manner. The -3 dB points are given by:

$$e^{-(\pi \cdot f / \alpha)^2} = \frac{1}{\sqrt{2}} \text{ or } (\pi \cdot f / \alpha)^2 = \ln\left(\sqrt{2}\right) \text{ or } f^2 = \frac{\alpha^2}{\pi^2} \frac{1}{2} \ln(2) \tag{II.56}$$

Therefore: $f = \pm \frac{\alpha \sqrt{\ln(2)}}{\pi \sqrt{2}} = \frac{\alpha \sqrt{2 \ln(2)}}{2\pi}$. The difference of these values equals the bandwidth:

$$B = \frac{\alpha}{\pi} \sqrt{2 \ln(2)} \text{ hertz} \tag{II.57}$$

A final example of a function with its Fourier transform is the damped sine wave:

$$x(t) = e^{-\alpha t} \sin(\omega_R t) \quad (t \geq 0) \tag{II.58}$$

Using the Euler formula for the sine its Fourier transform becomes:

$$X(\omega) = \int_0^\infty e^{-\alpha t} \frac{1}{2j} \left[e^{j\omega_R \cdot t} - e^{-j\omega_R \cdot t} \right] \cdot e^{-j\omega t} dt$$

which evaluates as:

$$X(\omega) = \frac{1}{2j} \left[\int_0^\infty e^{(-\alpha + j\omega_R - j\omega)t} dt - \int_0^\infty e^{(-\alpha - j\omega_R - j\omega)t} dt \right]$$

$$= \frac{1}{2j} \left[\frac{e^{(-\alpha + j\omega_R - j\omega)t}}{-\alpha + j\omega_R - j\omega} - \frac{e^{(-\alpha - j\omega_R - j\omega)t}}{-\alpha - j\omega_R - j\omega} \right]_0^\infty$$

A complex vector like $e^{-j\omega t}$ remains between -1 and +1 when t increases to infinity while the multiplier $e^{-\alpha t}$ becomes 0 so we can evaluate the equation into:

$$X(\omega) = \frac{1}{2j} \left[\frac{1}{\alpha - j\omega_R + j\omega} - \frac{1}{\alpha + j\omega_R + j\omega} \right]$$

This leads to the complex spectrum:

$$X(\omega) = \frac{\omega_R}{(\alpha + j\omega)^2 + \omega_R^2} \tag{II.59}$$

The fact that the spectrum is complex means that it contains sines and cosines which is understandable as the damped sine wave is not an even function but a complex function. A graph of the real and imaginary parts of the spectrum is shown in fig. II.3.

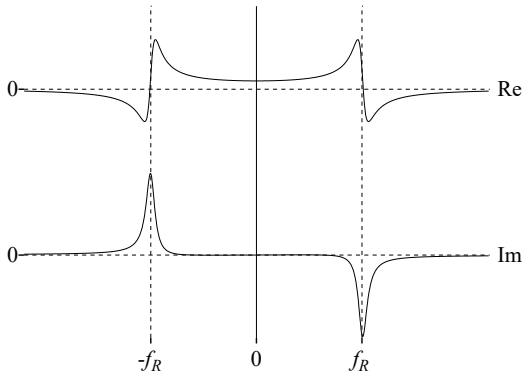


Fig. II.3. Real (top) and imaginary (bottom) parts of the complex spectrum of a damped sine wave.

At $f = 0$ the imaginary part is 0 and the real value is equal to the average of the time function (the *DC component*). At $f = f_R$ the real part is not exactly, but very near, zero if $\alpha^2 << \omega_R^2$. In the figure, this difference is too small to be visible. It means that the spectral peak frequency of the damped sine spectrum is slightly lower than the "original" sine frequency

ω_R. In the box called **SPECTRAL PEAK POSITION OF DAMPED SINE** the exact peak frequency is calculated. This resonance shift is often negligible: when, for example, every damped sine period amplitude is half of its preceding period's amplitude, the shift is less than 0.6%.

To apply the complex Fourier transform to *sampled* signals we must bear in mind that the sampled signal consists of N samples so that, for t running from 0 to T, the index n runs from 0 to N. The time variable changes into $t = n/N$. Applying this to formula II.42 turns it into its discrete version:

$$X_k = \frac{1}{N}\sum_{n=0}^{N-1} x_n \cdot e^{-j\cdot 2\pi\cdot k\cdot n/N} \quad (k = 0\ldots N\text{-}1) \tag{II.60}$$

where k represents the steps in the frequency domain and n the samples in the time domain. Because negative indexes of the output array are usually avoided, k runs from

SPECTRAL PEAK POSITION OF DAMPED SINE

The damped sine complex spectrum is (formula II.59): $X(\omega) = \dfrac{\omega_R}{(\alpha + j\omega)^2 + \omega_R^2}$

Multiplying the denominator with its complex conjugate produces the square of its magnitude:

$D^2 = (\alpha^2 - \omega^2 + \omega_R^2 + 2\alpha j\omega)(\alpha^2 - \omega^2 + \omega_R^2 - 2\alpha j\omega) = (\alpha^2 - \omega^2 + \omega_R^2)^2 + 4\alpha^2\omega^2$

$= \alpha^4 + \omega^4 + \omega_R^4 + 2\alpha^2\omega^2 - 2\omega_R^2\omega^2 + 2\alpha^2\omega_R^2$.

To calculate the position of its minimum, i.e. the exact frequency of the peak of the amplitude spectrum, we take the 1st derivative and set it to 0:

$\dfrac{dD^2}{d\omega} = 4\omega^3 + 4\alpha^2\omega - 4\omega_R^2\omega = 0$. One root lies at ω=0 and is of no concern here. Dividing by 4ω gives: $\omega^2 = \omega_R^2 - \alpha^2$. So, the peak is positioned at $\omega_P = \sqrt{\omega_R^2 - \alpha^2}$.

0 to N-1. Due to the cyclic property of the spectrum of sampled signals it is mirrored at the Nyquist frequency so that the 'negative' frequencies are positioned beyond the Nyquist, i.e. in the range of $N/2\ldots N$-1. The complete spectrum for positive frequencies, therefore, is defined by the complex numbers in the range $k = 0 \ldots N/2$.

The *inverse* complex Fourier transform is the sum of the complex Fourier components:

$$x_n = \sum_{n=-\infty}^{\infty} X_k \cdot e^{j2\pi\cdot k\cdot n/N} \tag{II.61}$$

Appendix III: Laplace transform

The responses of practical filters start from the moments of excitation ($t = 0$) and vanish some time after the excitation ends. When the Fourier transform is used to describe the output signals the once-occurring response is thought to consist of continuous sinusoidal waves that go on forever and always existed in the past. It seems, therefore, that the Fourier transform will not necessarily be the most suitable tool to define filter impulse responses. In fact, the behavior of systems in nature can be described by *differential equations* more adequately than by steady state functions. (Additionally, there exist some functions for which the Fourier integral cannot be calculated at all as their integral is not convergent.)

When a transform uses *damped* sines instead of continuous ones this (complex) transform can be written like:

$$X(\sigma,\omega)=\int_0^\infty x(t)\cdot e^{-\sigma t}e^{-j\omega t}\,dt \tag{III.1}$$

Compared to the Fourier transform this function runs from $t = 0$ instead of $t = -\infty$ and the spectral components are *complex exponentials*. The exponents of e can be combined in one complex variable s:

$$X(s)=\int_0^\infty x(t)\cdot e^{-st}\,dt \tag{III.2}$$

where $s = \sigma + j\omega$. This formula represents the **Laplace transform**. For the Laplace spectrum of the resonator, which has only one damped sine as impulse response, only one complex exponential suffices. In that case σ is equal to $-\alpha$, the damping factor of the damped sine wave. When $\sigma = 0$ the Laplace transform changes into the Fourier transform.

When we look at the Fourier transform of the resonator (see formula II.59) we can write its denominator $(\alpha + j\omega)^2 + \omega_R^2$ as $(j\omega)^2 + 2\alpha(j\omega) + \alpha^2 + \omega_R^2$. As this is a 2nd order function we can express it by $(j\omega - p_1)\cdot(j\omega - p_2)$ where $p1$ and $p2$ are the *roots* of the function. We can evaluate these roots as

$$(j\omega)_{1,2}=\frac{-2\alpha\pm\sqrt{4\alpha^2-4\left(\alpha^2+\omega_R^2\right)}}{2}=-\alpha\pm j\omega_R \tag{III.3}$$

Because the denominator is 0 when $j\omega$ is equal to a root, the function becomes infinite. Therefore, these positions are called **poles**. The poles of the resonator filter are complex conjugates: they form a *complex pole pair*. Because any function can be expressed by

a denominator function of some order n and a numerator of some order m (= n or lower) we can write this as:

$$H(s) = \frac{(s-d_1)\cdot(s-d_2)\ldots(s-d_m)}{(s-p_1)\cdot(s-p_2)\ldots(s-p_n)} \qquad \text{(III.4)}$$

For obvious reasons the roots of the numerator are called **zeros** (in the formula called d for *dips*, so as to avoid confusion with the *z-transform* which is described in appendix IV). Now the poles and zeros of a Laplace transform can be displayed in a complex plane, the **s-plane**, where σ is represented on the real axis and ω on the imaginary axis. (Note that this representation has nothing to do with the rotating complex vectors of sinusoidal waves as covered in appendix II.1) In fig. III.1 the pole pair of the resonator filter is displayed in this way. The magnitude of $H(s)$ cannot be seen in these **pole-zero plots**: they contain only the positions of the poles and zeros in the complex plane. To add the magnitude of $H(s)$ to the figure an extra dimension is needed. However, at the poles the magnitude is infinite which cannot be represented. Sometimes 3D pictures are published where the function at the poles is displayed with strong peaks (see fig. III.2).

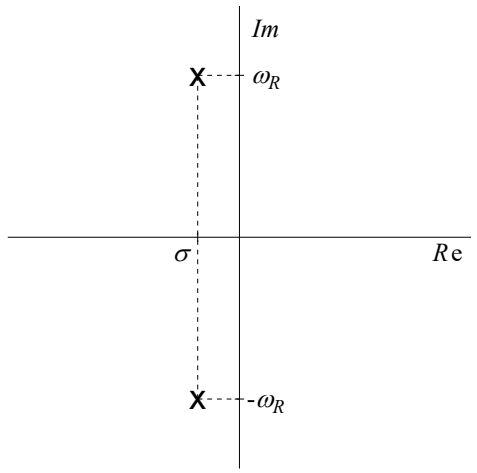


Fig. III.1. Complex pole-pair of the resonator filter spectrum in the complex s-plane. There are no zeros in this function. (Zeros normally are marked with 0.)

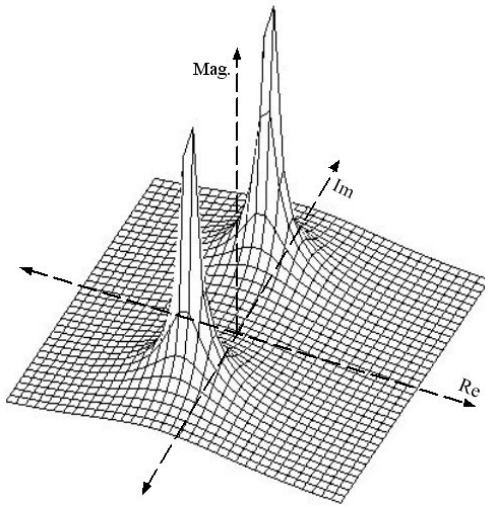


Fig. III.2. Complex pole-pair in 3D.

Any filter function can be defined by poles and zeros. For an example, look at fig. III.3 for a pole-zero plot of a Butterworth 10th order broad band-pass filter consisting of five complex conjugate pole pairs and five zeros at the origin. In all pole-zero diagrams the poles are positioned in the left half of the plane, which should be understandable: only when σ is negative the exponential sine wave components associated with a pole pair would decrease in amplitude. Pole pairs in the right half of the plane would result in unlimited increase of the amplitude and therefore in an unstable filter.

The benefit of the Laplace transform lies in the possibility of defining most types of (causal) filters by a limited number of poles and zeros which simplifies the complex equations. Another useful property of the Laplace transform is that using it for the design of analog electronic filters, the calculation of them can be greatly simplified. A crucial electronic device to build a filter is the **capacitor** (also called **condenser**). A capacitor can collect and hold electrical charge when a current is flowing through it. The longer the duration of the current, the higher the charge becomes. It is a *current integrator*: the voltage of the capacitor is proportional to the integral of the current:

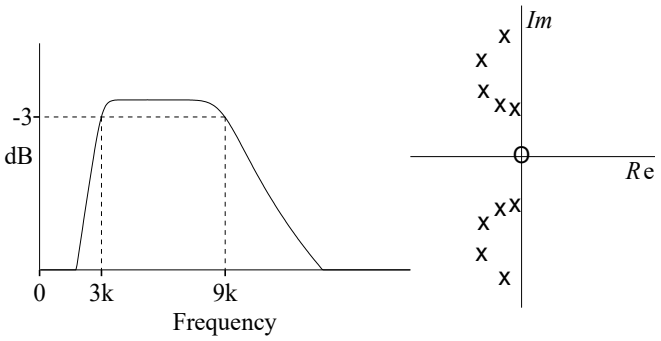


Fig. III.3. Complex pole-pairs of a Butterworth band-pass filter in the complex plane. There are five zeros coinciding at the origin.

$$v(t) = \frac{1}{C}\int_{-\infty}^{\infty} i(t)dt \quad \text{or, which is equivalent:} \quad i(t) = C\frac{dv(t)}{dt} \tag{III.5}$$

Here *v* represents the voltage across the capacitor, *i* the current through it and *C* a constant (the *capacitance*) which is dependent of the amount of electrical charge (proportional to the number of electrons) of the device for a specific voltage.

The opposite is possible with an **inductor**. The voltage across the device is proportional to the speed of current changes. It is a current differentiator:

$$v(t) = L\frac{di}{dt} \tag{III.6}$$

where *L* is a constant that represents the *inductance* of the device. To define a filter function, therefore, the functions of formulas III.5 and III.6 are needed within the equation which turns it into a *differential equation*. It describes the filter completely. However, the math to calculate a filter of some complexity can become quite difficult. The behavior of the filter as regards *complex sinusoidal waves* can greatly simplify the calculation. If we write $v(t)$ as a complex sinusoidal wave, according to appendix II.2, we get: $v(t) = A \cdot e^{j\omega t}$. Its derivative is then:

$$\frac{dv(t)}{dt} = A \cdot j\omega \cdot e^{j\omega t} \tag{III.7}$$

By combining formula III.7 with the right part of formula III.5 we can write for $i(t)$:

$$i(t) = C \cdot A \cdot j\omega \cdot e^{j\omega t} \tag{III.8}$$

Now, just like the *resistance* of a resistor equals the voltage divided by the current through it (see the box **OHM'S LAW** in section 1), the same Ohm's law can be applied to the capacitor:

$$Z_C = \frac{v(t)}{i(t)} = \frac{A \cdot e^{j\omega t}}{C \cdot A \cdot j\omega \cdot e^{j\omega t}} = \frac{1}{j\omega C} \tag{III.9}$$

We do not speak of resistance now but of *impedance*. The reason is that the impedance of a capacitor is no simple quantity but a *vector*: the phase of a sinusoidal voltage across the capacitor is $\pi/2$ radians *behind* the phase of the current through the capacitor. In a similar way we can prove that the impedance of an inductor equals:

$$Z_L = j\omega L \tag{III.10}$$

which means that the voltage across an inductor is $\pi/2$ radians *ahead* of the current through it. These impedances are special cases: in general, the phase angle of an impedance can have any value. When phase angles of $\pi/2$ radians are involved, the term **reactance** is used as in the case of capacitors and of inductors. Commonly, the character X is used for reactance and Z for impedance. For the resistance the symbol R is used (which is no vector but a quantity as it is frequency-independent and has no phase difference between current and voltage).

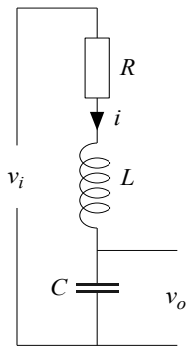


Fig. III.4. Second order band filter realized by using electronic components.

To illustrate the flexibility of working with complex impedances let's explore a simple case (see fig. III.4). The current through all three components equals $i = v_i / z$ where z simply is the sum of the individual impedances: $z = R + j\omega L + 1/(j\omega C)$. The output voltage is taken from the capacitor. Its voltage is $v_o = i \cdot X_C = i/(j\omega C)$. Substituting v_i / z for i leads to:

$$\frac{v_o}{v_i} = G(j\omega) = \frac{1}{j\omega C\left(R + j\omega L + \dfrac{1}{j\omega C}\right)} = \frac{1}{LC(j\omega)^2 + CR(j\omega) + 1} \tag{III.11}$$

which shows the filter function directly as a 2nd order band-pass with one complex pole pair and no zero. The resonance frequency ω_R approximates $1/\sqrt{LC}$ and the damping α is $R/(2L)$ (see the box called **RLC RESONATOR** for comparison to the "standard" one-pole-pair filter of equation II.54).

Any analog filter basically consists of a combination of several capacitors, resistors and/or inductors. When combined with *operational amplifiers* (semiconductor chips that are integrated amplifiers which have almost ideal properties so that in practice the behavior of the complete filter system can be completely defined by its formula) the filter design possibilities are almost unlimited. The combination of filter components with operational amplifiers is called **active filters**. It is not even necessary to apply inductors: they can be simulated perfectly by combinations of capacitors and operational amplifiers. The stability, which is always an important issue in analog filter design, depends almost solely on the external (*passive*) components, which can be made very stable. In the practical part of the book an example of an active filter is described.

The Laplace transform together with the pole-zero concept provides for a suitable tool

RLC RESONATOR (fig. III.4)

Dividing numerator and denominator of equation III.11 by *LC* yields:

$$G(j\omega) = \frac{1/(LC)}{(j\omega)^2 + R/L \cdot (j\omega) + 1/(LC)}$$

This resonator transfer function is a second order function with one pole pair and no zeros so that the impulse response is a damped sine. To compute the poles, we take the complex roots of the denominator:

$$(j\omega)_{1,2} = \frac{-R}{2L} \pm \frac{1}{2}\sqrt{\frac{R^2}{L^2} - \frac{4}{LC}} = \frac{-R}{2L} \pm j\sqrt{\frac{1}{LC} - \frac{R^2}{4L^2}}$$

The real part of the poles $\sigma = -R/(2L)$ so that the damping α of the damped sine is equal to $R/(2L)$. Substituting α for $R/(2L)$ yields: $p_{1,2} = \alpha \pm j\sqrt{\frac{1}{LC} - \alpha^2}$. If we compare this with the spectrum of the damped sine (formula II.59) we see that the 'resonance frequency' ω_R of the damped sine is independent of the damping (it is defined by the zero crossing distances) whereas the equivalent resonance frequency of this resonator is slightly lowered by the damping.
For estimation of the peak position we multiply the denominator of formula III.1 with its complex conjugate:
$(1 - \omega^2 LC + j\omega CR)(1 - \omega^2 LC - j\omega CR) = 1 + \omega^4 L^2 C^2 - 2\omega^2 LC + \omega^2 C^2 R^2$. Setting its 1st derivative to 0 gives: $4L^2C^2\omega^3 - 4LC\omega + 2C^2R^2\omega = 0$ Dividing by 2ω results in:
$\omega^2 = \frac{2LC - C^2R^2}{2L^2C^2} = \frac{1}{LC} - \frac{R^2}{2L^2}$. So, the position of its peak is: $\omega_P = \sqrt{\frac{1}{LC} - 2\alpha^2}$.
Another difference with the formula II.59 is the ω_R^2 instead of ω_R in the numerator. This is only a scaling factor. The resonator's peak level equals ω_R times the level of the peak of equation II.59 which is typical for practical resonators: the magnitude at resonance can be boosted to high levels when the losses in the inductor and capacitor (that work as resistance and are contained within *R*) are low.

for calculation of analog filters. *Sampled* versions of signals, however, open the gate to *digital filtering*. The proper transform for digital signals and filters is the **z-transform**

which is described in appendix IV. The stability and properties of digital filters greatly outrank the analog filtering's possibilities and for this reason the analog filter design (and with it the Laplace transform) have become less popular.

Appendix IV. Z transform

The Laplace transform (see appendix III) of a Dirac pulse centered at $t = 0$ is:

$$X(s) = \int_{-\infty}^{\infty} \delta(t) \cdot e^{-st}\, dt \tag{IV.1}$$

As the area of the Dirac pulse is 1 by definition, the Laplace transform of a Dirac pulse at $t = 0$ equals 1. When a delayed Dirac pulse occurs at $t = T$ each frequency component is delayed T seconds:

$$X(s) = \int_{-\infty}^{\infty} \delta(t - T) \cdot e^{-st}\, dt = e^{-sT} \tag{IV.2}$$

A sampled signal can be seen as a series of stepwise delayed Dirac pulses, each one weighted by the value of the sample at that time:

$$x(t) = x_0 \cdot \delta(t) + x_1 \cdot \delta(t - T) + x_2 \cdot \delta(t - 2T) + \cdots \tag{IV.3}$$

where T is the sampling period. Its Laplace transform then will be:

$$X(s) = x_0 + x_1 \cdot e^{-sT} + x_2 \cdot e^{-s2T} + \cdots \tag{IV.4}$$

where $s = \sigma + j\omega$. Defining a complex variable as $z = e^{sT}$ we can write:

$$X(z) = x_0 + x_1 z^{-1} + x_2 z^{-2} + \cdots \tag{IV.5}$$

which represents the **z-transform** of $x(t)$. Because the power of z equals the number of time shifts the function z^{-1} is called the **unit delay function** for the function concerned.

When s is purely imaginary it equals $j\omega$. In this special case X is a function of $e^{j\omega T}$ which means that its complex values all lie on the unit circle, and ω is defined by the angle of the vector with the real axis. While ω varies from 0 to infinity the X values are repeated every 2π radians.

When s is purely real it equals σ and $z = e^{\sigma \cdot T}$. At positive values of σ the z values are real and >1. If σ is negative the z values exist between 0 and 1. The vector has a radius $r = e^{\sigma \cdot T}$ and an angle $\varphi = \omega T$. Visualizing this in the complex plane presents the **z-plane** (see fig. IV.1).

As a consequence, a particular point on the unit circle in the z-plane will correspond to an infinite number of positions on the imaginary axis in the s-plane, all 2π radians apart. This circular property stems from the sampled nature of the time signal which implies repeating the spectrum and its mirror at $2\pi/T$ intervals (as described in section 17 about sampling). As in the case of all sampled signals, the spectral information is contained in the range from 0 to the Nyquist frequency, which is covered by the angle range 0 to π in the z-plane. The range π to 2π in the lower half forms the mirror of the upper half. The polar coordinates of the complex z values being radius $r = e^{\sigma T}$ and angle ωT, it follows that the rectangular coordinates are r cos (ωT) for the real value and r sin (ωT) for the imaginary value (see fig. IV.1).

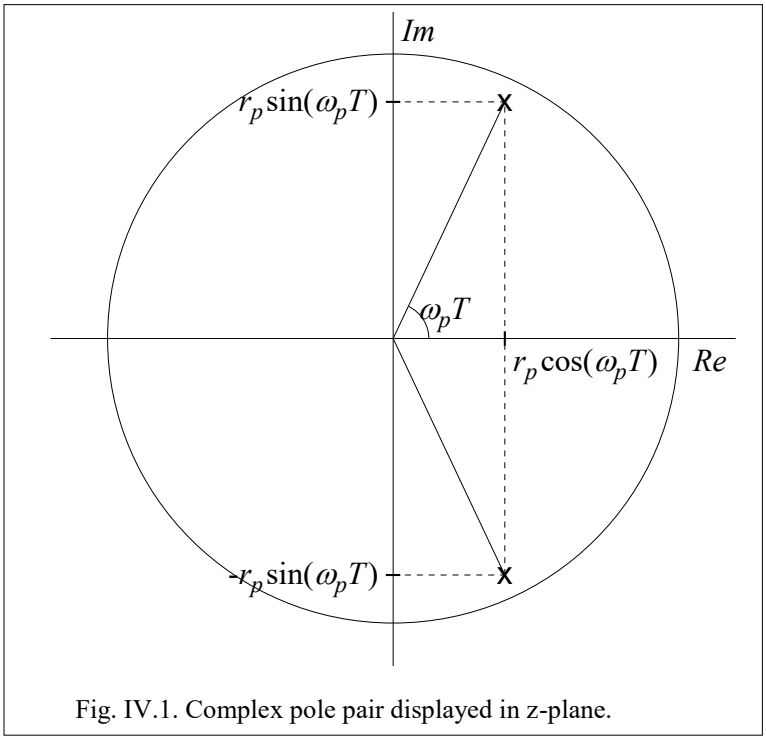


Fig. IV.1. Complex pole pair displayed in z-plane.

For a filter to be stable the poles must lie at the left side of the s-plane (as explained in appendix III), which means *inside the unit circle* of the z-plane.

How can we use the z-transform to compute the recursive coefficients of digital filters? In section 18 about digital filters it is shown that a recursive filter that uses a part b of the most recent previous output can be defined by:

$$y_n = x_n + b \cdot y_{n-1} \tag{IV.6}$$

When a *unit pulse* is applied to this filter (then x_0=1 and all succeeding x values are zero) it will be clear that each next output sample value will become the value of its predecessor multiplied with b so that an exponential impulse response will emerge. The relation of two adjacent output samples of this filter can be defined as:

$$y_n = b \cdot y_{n-1} \tag{IV.7}$$

Obviously, the impulse response theoretically runs to infinity as this is an IIR filter. If $b < 1$ the function will decrease exponentially and if $b > 1$ the function will increase exponentially to infinity. If we express the *former* value as a function of the current value we get:

$$y_{n-1} = b^{-1} \cdot y_n \tag{IV.8}$$

Delaying *m* positions means *m* times multiplication by b^{-1} so that:

$$y_{n-m} = b^{-m} \cdot y_n \tag{IV.9}$$

For *b* we can apply any function, even complex ones. If we use the function $z = e^{sT}$ we have transformed the function $x(t)$ into its frequency domain by the z-transform:

$$y_{n-m} = z^{-m} \cdot y_n \tag{IV.10}$$

Recalling the general formula (18.11) for the recursive filter from section 18:

$$y_n = a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2} + ... + b_1 y_{n-1} + b_2 y_{n-2} + ... \tag{IV.11}$$

we can change its subscripts by applying formula IV.10. We can do this for the output array (the *y* values) as well as for the *input array* (the dependence of the filter on the input and the output is completely defined by the different *a* and *b* coefficient values) so that formula IV.11 changes into:

$$y_n - b_1 y_n z^{-1} - b_2 y_n z^{-2} - .... = a_0 x_n + a_1 x_n z^{-1} + a_2 x_n z^{-2} + ... \tag{IV.12}$$

To define the output/input relation of the filter we can manipulate the formula into a different form: $y_n\left(1 - b_1 z^{-1} - b_2 z^{-2} - ...\right) = x_n\left(a_0 + a_1 z^{-1} + a_2 z^{-2} + ...\right)$ so that we can write:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3} + ...}{1 - b_1 z^{-1} - b_2 z^{-2} - b_3 z^{-3} - ...} \tag{IV.13}$$

This is the general equation for the z-transform of a system's transfer function. The functions $Y(z)$ and $X(z)$ are not functions of time (or Δt) but are the *z-transforms* of output and input signals respectively. Just like Fourier transforms of output and input signals must be divided to get the transfer function, the same applies to the z-transforms of the output and the input.

An arbitrary 2nd order filter function, for example, can be defined as:

$$H(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 - b_1 z^{-1} - b_2 z^{-2}} \tag{IV.14}$$

Multiplying numerator and denominator with z^2 changes the powers to positive values:

$$H(z) = \frac{a_0 z^2 + a_1 z + a_2}{z^2 - b_1 z - b_2} \tag{IV.15}$$

Because both the numerator and denominator in this example consist of a 2nd order or *quadratic* equation, this function is called a **biquad**. Obviously, it has two poles and two zeros. Theoretically the number of components (here the power of z) of the z-transform of an arbitrary function can be infinite. In the case of filters, however, the number of components may be very limited because the filter functions in the frequency domain are usually much simpler than spectra of sounds analyzed in practice. The maximum order to calculate filters can even be limited to 2 because higher orders can be built by *cascading* lower order filter sections like biquads. Cascading then can be realized either by multiplying the individual z-transforms of the low order sections and extracting the resulting recursion coefficients, or by repetitive filtering of the signal with all lower order sections in turn.

To calculate the digital filter coefficients from a complex filter function we use the "standard" one pole pair filter of formula II.59 as a simple example. The roots of its denominator were calculated in appendix III: $(j\omega)_{1,2} = -\alpha \pm j\omega_R$. (Its numerator has no roots so there are no zeros.) To map these values into the z-plane we must bear in mind that σ in the z-transform is equal to $-\alpha$ so that $r = e^{-\alpha T}$. Furthermore, the angle in the z-plane $\varphi = \omega_R T$. Starting from its general z-transform we write:

$$H(z) = \frac{a_0 z^2}{z^2 - b_1 z - b_2} = \frac{a_0 z^2}{(z - p_1)(z - p_2)} = \frac{a_0 z^2}{z^2 - (p_1 + p_2)z + p_1 p_2} \qquad \text{(IV.16)}$$

where $p_1 = r[\cos(\omega_R T) + j \cdot \sin(\omega_R T)]$ and $p_2 = r[\cos(\omega_R T) - j \cdot \sin(\omega_R T)]$ so that $b_1 = p_1 + p_2 = 2r \cdot \cos(\omega_R T)$ and $b_2 = p_1 p_2 = r^2$. The numerator has two zeros at z = 0 which is caused by the multiplication of the whole function with z^2. This has no influence on the filtering, except for a simple forward time shift of the filter response. Because the original complex filter function (formula II.59) has ω_R in its numerator, the coefficient $a_0 = \omega_R T$, which scales the initial amplitude of the damped sine impulse response to 1. The final recursion formula then will be:

$$y_n = \omega_R T \cdot x_n + 2r\cos(\omega_R T) \cdot y_{n-1} - r^2 \cdot y_{n-2} \qquad \text{(IV.17)}$$

where $r = e^{-\alpha T}$.

The same algebra can be applied to the biquad's numerator. Setting the scaling factor a_0 to 1 will produce similar roots for the numerator as in the case of the denominator: $d_1 + d_2 = 2r_d \cdot \cos(\omega_d T)$ and $d_1 d_2 = r_d^2$ (where the *d*'s refer to the zeros or *dips*, r_d to the radius of the zero vectors and ω_d to the frequency of the zero). Substituting these values in the numerator yields: $a_1 = -2r_d \cdot \cos(\omega_d T)$ and $a_2 = r_d^2$. The complete set recursive coefficients for the biquad then will be:

$$a_0 = 1 \qquad b_1 = 2r_p \cos(\omega_p T)$$
$$a_1 = -2r_d \cos(\omega_d T) \qquad b_2 = -r_p^2$$
$$a_2 = r_d^2$$

where ω_p is the pole frequency, ω_d the zero frequency, r_p the pole vector radius and r_d the zero vector radius.

The zeros of a function can be placed anywhere in the z-plane, even *on the unit circle*. The system then remains stable: the function is zero for the frequency defined by the angle. As an example, let's add two zeros on the unit circle at the same angles as those of the pole vectors, see fig. IV.2. In the equations of the biquad recursive coefficients we simply substitute 1 for r_d and ω_p for ω_d . The result is a *notch filter* that completely suppresses the specified frequency.

For the filter impulse response to be real, the poles and zeros in the upper half must be

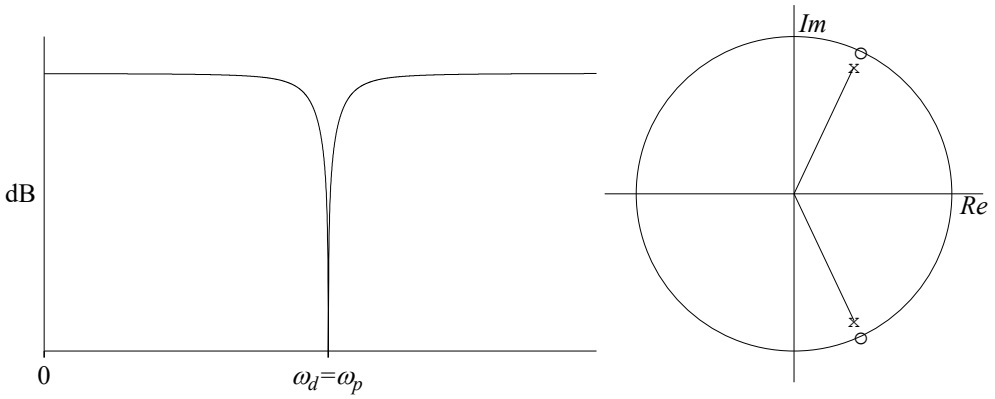


Fig. IV.2. Notch filter realized by adding zeros to a pole pair filter.

mirrored in the lower half of the z-plane, or else be positioned on the real axis.

From the pole-zero positions in the z-plane it is not very difficult to get an impression of the magnitude of the filter function at a specific frequency by using formula III.4 which is repeated here:

$$H(z) = \frac{(z-d_1)\cdot(z-d_2)\ldots(z-d_m)}{(z-p_1)\cdot(z-p_2)\ldots(z-p_n)} \qquad \text{(IV.18)}$$

So, from the position of a frequency on the unit circle all distances to the zeros must be multiplied and the result divided by the product of all distances to the poles. As the poles and zeros in the upper half have their mirrors in the lower half, it suffices to do this for one half of the plane only.

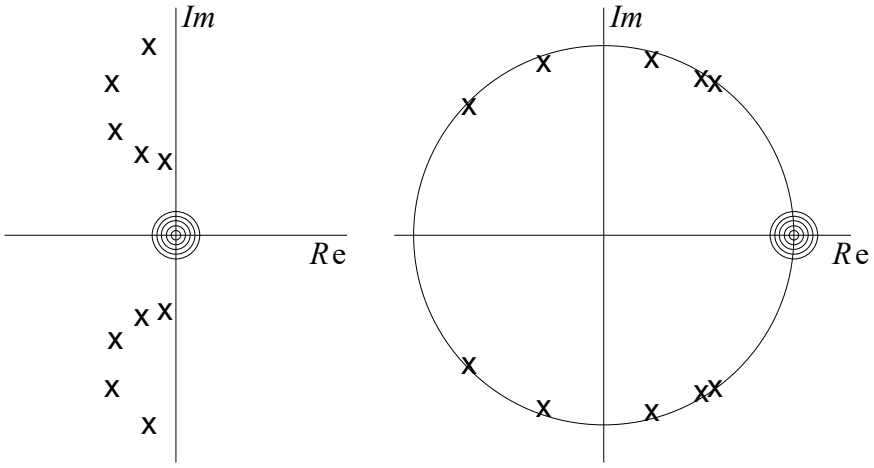


Fig. IV.3. Comparison of poles and zeros in s-plane and z-plane of 10th order Butterworth band-pass filter from appendix III.

For a comparison of the s-plane and the z-plane, fig. IV.3 depicts the poles and zeros of the broad band Butterworth filter from appendix III (fig. III.3) in the s-plane as well as in the z-plane. Often the multiple zeros on the same position are drawn as concentric circles, as can be seen in the figure. You see that the poles in the z-plane are positioned very near the unit circle. This follows from the fact that $r = e^{-\alpha T}$ and the sampling period T in practice is very small so that r will be close to 1. In fact, r is the amplitude decay factor of the damped sine within the short sampling interval. This means that the number precision of the computer used has to be sufficiently high, especially at high sampling frequencies and higher order filters.

In the z-plane the positions of the poles and zeros depend on the sample frequency, as their angles are proportional to T, the sampling period.

This section about the z-transform certainly is not the whole story. Many sophisticated filtering methods using the z-transform (or modified z-transform) have been developed and much more could be said about the subject. The purpose of this appendix, however, is only to explain something about the basic z-transform principles. For more information, there are many books with extensive explanations and math concerning the z-transform.

References

[1] Boersma, Paul (1993): "Accurate short-term analysis of the fundamental frequency and the harmonic-to-noise ratio of a sampled sound." *Proceedings Institute of Phonetic Sciences of the University of Amsterdam,* 17: p 97-110.

[2] Boersma, Paul & David Weenink (2013). *Praat: doing phonetics by computer* (Version 5.3.51), [http://www.fon.hum.uva.nl/praat/] (retrieved 2 June 2013).

[3] Farina, Angelo (2007). "Advancements in impulse response measurements by sine sweeps." *AES 122nd Convention paper*, May 5.8.

[4] Freeman, Larry (2007). "A shorter proof of Euler's formula." *Math Refresh: Review of fundamental math concepts in a straight-forward, accessible way.* [http://mathrefresher.blogspot.com/2007/10/shorter-proof-of-eulers-formula.html] (retrieved October 2012).

[5] Heinzel, G., A. Rüdiger & R. Schilling (2002). "Spectrum and spectral density estimation by the Discrete Fourier Transform (DFT), including a comprehensive list of window functions and some new flat-top windows." *Report from Max-Planck-Institut für Gravitationsphysik Hannover*.

[6] Jong, Nivja H. de & Ton Wempe (2009). "Praat script to detect syllable nuclei and measure speech rate automatically." *Behavior Research Methods*, 41: 385-390.

[7] Lynn, Paul A. (1973). *An Introduction to the Analysis and Processing of Signals.* The MacMillan Press ltd. London.

[8] (07/09/2011). "adult_female_speech.wav". *Open Air Library (Anechoic Audio Database).* [http://www.openairlib.net/anechoicdb/content/acoustics-and-psychoacoustics-book]. (Retrieved April 2018).

[9] (16/06/2015). "ir_church_saint-laurentius_molenbeek_bekkevoort_belgium.wav". *Open Air Library (Impulse Response Database).* [http://www.openairlib.net/auralizationdb/content/saint-lawrence-church-molenbeek-wersbeek-belgium]. (Retrieved April 2018).

[10] Plomp R. & W. J. M. Levelt (1965). "Tonal Consonance and Critical Bandwidth". *Journal of the Acoustical Society of America*,38, 548-560.

[11] Rabiner, L. R and R.W. Schafer (1978). *Digital Processing of Signals*. Englewood Cliffs, New Jersey: Prentice-Hall, Inc.

[12] Sethares, W.A. (1993). "Local consonance and the relation between timbre and scale." *Journal of the Acoustical Society of America* 94 (3), Pt.1: (1218-1228).

[13] Smith, Steven, W. (2003). *Digital Signal Processing: A practical guide for engineers and scientists*. Burlington: NewnesPres Elsevier.

[14] Weenink, David (2013). *Speech Signal Processing with Praat*. [http://www.fon.hum.uva.nl/david/sspbook/sspbook.pdf].

INDEX

E

F

N

O

P

Q

R

S

T

U

V

W

Z